BRUNO BASSO

ANÁLISE SOBRE O *SOFTWARE* DE CÓDIGO ABERTO COMO UMA INOVAÇÃO TECNOLÓGICA

Monografia apresentada como requisito parcial à obtenção do título de bacharel no Curso de Ciências Econômicas, Setor de Ciências Sociais Aplicadas, Universidade Federal do Paraná.

Orientador: Prof. Dr. José Wladimir Freitas da Fonseca

CURITIBA
2008

TERMO DE APROVAÇÃO

BRUNO BASSO

ANÁLISE SOBRE O *SOFTWARE* DE CÓDIGO ABERTO COMO UMA INOVAÇÃO TECNOLÓGICA

Monografia apresentada como requisito parcial à obtenção do título de bacharel no Curso de Ciências Econômicas, Setor de Ciências Sociais Aplicadas, Universidade Federal do Paraná.

Orientador: _____
Prof. José Wladimir Freitas da Fonseca
Departamento de Economia, UFPR

_____
Prof. Othon Juruá Rolim de Souza Reis
Departamento de Economia, UFPR

_____
Prof. Igor Zanoni Constant Carneiro Leão
Departamento de Economia, UFPR

CURITIBA

2008

Os antropólogos reclamam que apesar de conhecerem montanhas de fatos são incapazes de organizá-los numa teoria. Exatamente o oposto, dizem, dos economistas, que conhecem poucos fatos, mas são capazes de reuni-los em múltiplas e belíssimas teorias.

Delfim Neto

SUMÁRIO

RESUMO

O presente trabalho monográfico aborda o fenômeno do *software* livre sob a ótica da inovação tecnológica e como seu modelo de funcionamento se diferencia do *software* proprietário, criando assim um processo de produção diferenciado e aperfeiçoado do padrão vigente. O presente estudo está dividido em dois capítulos. O primeiro tratará do arcabouço teórico para compreender melhor os processos de inovação tecnológica, dos paradigmas tecnológicos e seus pormenores. O segundo capítulo tratará do fenômeno, suas diferenciações e da maneira com que se dá o seu processo de criação e desenvolvimento.

Palavras Chave: inovação tecnológica, paradigma tecnológico, *software* livre.

INTRODUÇÃO

Os processos de inovação tecnológica podem ser entendidos como a melhoria de um processo ou de um produto com o objetivo de cumprir melhor uma função, ou de criar outras, que venham a ter algum impacto no seio da firma. Com a revolução industrial, a partir do século XVIII, houve um avanço tecnológico crescente na indústria mundial, causando assim uma revolução tecnológica sem precedentes na história da humanidade.

Essa revolução, em diversos campos, veio a revolucionar não só a maneira com que as empresas encaravam os processos de produção, como mudaram drasticamente a maneira com que os consumidores passaram a encarar determinados mercados. Um desses setores, ainda mais moderno, é o do *software*, que veio a ter existência somente na segunda metade do século XX, causando uma revolução ainda maior, e exigindo uma interação tecnológica maior dos consumidores.

Dos primeiros computadores comercializados no mundo, até os dias de hoje, houve uma grande mudança no cenário econômico mundial, tendo em vista que este mercado considerado insignificante na década de 1970 tornou-se um dos maiores mercados, movimentando bilhões de dólares ao redor do mundo. Em 1980, poucas empresas, e raras poucas pessoas realmente necessitavam de um computador. Hoje, para que uma pequena empresa possa sobreviver no mercado, há, no mínimo, a necessidade de um microcomputador conectado à internet.

Assim como o preceito da economia é o de "meios escassos para fins infinitos", na informática pode-se fazer um paralelo entre "aplicativos escassos para objetivos infinitos", já que pessoas diferentes precisam de produtos diferentes.

Para criar um *software* comercial, as empresas passaram a criar aplicativos em massa, todos iguais, que atendessem os desejos da maioria. Embora a maioria fosse atendida por essa medida, sempre houveram pessoas à margem desse negócio. Dessa forma, com a universalização de ferramentas para o desenvolvimento de código fonte, houveram aqueles que, à sua própria maneira, revoltaram-se com a indústria e

passaram a, por conta própria, desenvolver produtos que viessem a atender suas necessidades.

Com a popularização da *internet*, e com o livre acesso à sites de *download*, à fóruns e encontros de grupo (Fisl – Fórum Internacional do *Software* Livre), ficou ainda mais simples obter aplicativos feitos sob medida para esses usuários até então marginalizados. Houve, portanto, a criação do modelo livre de desenvolvimento de código fonte.

Dessa forma, surgem os dois principais modos de desenvolvimento de aplicativos através de comunidades livres e organizadas, permitindo a criação de *softwares* personalizados para cada tipo de usuário.

O modelo do *software* livre é um movimento anárquico, social, em que o código fonte deve permanecer aberto, sem que haja qualquer apropriação do trabalho obtido em suas comunidades. Já o modelo de desenvolvimento de código aberto é um movimento ligado à criação de conhecimento, permitindo que seu código fonte seja apropriado e 'fechado' por qualquer pessoa ou empresa.

O objetivo principal deste estudo é analisar o *software* livre sob a ótica da inovação tecnológica, utilizando-se principalmente do modelo evolucionista, para fazer um estudo detalhado desse fenômeno. Em segundo plano, analisar ambos os modelos de desenvolvimento (livre e proprietário) com o propósito de entender qual é o paradigma tecnológico, e qual pode ser considerado a trajetória independente desse paradigma.

No capítulo 1. A TEORIA DA INOVAÇÃO E DO PARADIGMA TECNOLÓGICO será feita uma revisão teórica dos processos de inovação tecnológica, com preponderância à teoria evolucionista, para lançar uma luz atual e madura sobre o tema.

No capítulo 2. OS MODELOS DE DESENVOLVIMENTO DE *SOFTWARE*, SUAS VANTAGENS E DESVANTAGENS NO MERCADO CONCORRENCIAL E AS FORMAS DE DISSEMINAÇÃO DO CONHECIMENTO ADQUIRIDO serão estudadas as formas de criação de *software*, e de que maneira há a difusão do conhecimento gerado pelas firmas que criam, desenvolvem e comercializam seus

produtos. Em segundo plano, serão analisadas as formas com que esses aplicativos são licenciados para os usuários finais, e de que forma há uma distinção entre os atuais modelos de desenvolvimento. Embora existam atualmente três modelos de desenvolvimento do *software*, haverá uma clara distinção entre o modelo proprietário e o modelo livre, uma vez que há mais semelhanças entre o código livre e o código aberto do que há diferença entre esses dois modelos. Para facilitar a compreensão do fenômeno estudado, os modelos livres e abertos serão considerados como um só.

1. A TEORIA DA INOVAÇÃO E DO PARADIGMA TECNOLÓGICO

O objetivo deste capítulo é compreender de que forma ocorre o processo de inovação tendo como fenômeno estudado o *software* livre. Neste sentido a base teórica estabelecida é a teoria evolucionista. Assim, serão estudados os princípios de economia tecnológica apresentados por NELSON & WINTER; as trajetórias tecnológicas desenvolvidas por DOSI; e os setores produtivos tecnológicos de PAVITT.

O estudo se inicia com os modelos de aprendizado de NELSON & WINTER, uma das mais brilhantes teorias do campo tecnológico econômico e que, anos após ter sido apresentada ao mundo, continua apresentando formidáveis ferramentas para compreender a tecnologia econômica.

Em seguida será utilizada a teoria sobre o progresso técnico de DOSI, que irá servir de ligação entre a teoria de NELSON & WINTER e a de PAVITT, que revela as características de cada trajetória tecnológica dependentes da fonte de tecnologia do processo (seja interno ou externo), da dimensão relativa das empresas inovadoras e da intensidade e direção da diversificação tecnológica (tudo isso a partir de diferentes categorias de empresas).

Em seguida será estudada a teoria do paradigma tecnológico de DOSI, bem como suas trajetórias independentes de um modelo de paradigma, que virá a explicar o processo de desenvolvimento do *software* livre como trajetória independente ao modelo de desenvolvimento do *software* proprietário, que é o mais difundido atualmente.

As idéias que servem de arcabouço teórico para o fenômeno estudado são comumente interligados, com um certo grau de interligação entre seus trabalhos, porém, nos tópicos a seguir, para facilitar a colocação de idéias e para simplificar a análise da teoria utilizada, serão, na medida do possível, separados por autor, contendo assim suas principais idéias em um mesmo espaço, para que haja um melhor entendimento da teoria utilizada.

Entre essas três pedras angulares do presente trabalho, estão alguns autores

que também serão utilizados, em menor perspectiva, para um melhor compreendimento da matéria econômica sobre um aspecto totalmente atual, no qual a teoria econômica tem dado pouca importância: o mercado de *software*.

1.1 - O MODELO DE APRENDIZADO DE NELSON & WINTER

A teoria evolucionista é caracterizada como uma vertente da teoria Neo-schumpeteriana, a partir das publicações de NELSON & WINTER (2005), na tentativa de elaboração de um arcabouço teórico capaz de explicar a crescente importância e aceleração do processo de inovação tecnológica na economia. Das teorias de Schumpeter e dos neo-schumpeterianos, origina-se os conceitos de entendimento do sistema econômico em constante movimento e o papel das inovações tecnológicas no mesmo processo.

Os evolucionistas passam a basear o sistema econômico nas teorias biológicas da evolução de cunho Darwinista (a economia não é estática, ela sofre um processo dinâmico, abandonando a visão da mecânica clássica, a qual não leva em conta a tecnologia ou o conhecimento). Surge assim, um sistema que incorpora a aprendizagem, o conhecimento e o papel das inovações para o entendimento do processo de funcionamento da economia. Embora existam inúmeros autores que contribuíram com a teoria evolucionista, serão utilizados conceitos desenvolvidos por NELSON & WINTER, DOSI e PAVITT, sendo os mesmos abrangentes para objeto de análise neste estudo.

No entanto, na teoria evolucionista, a dinâmica econômica é baseada em inovações de produtos, processos e nas formas de organização da produção, onde as inovações não são graduais, podendo causar instabilidade ao sistema econômico. O conceito de maximização não é perfeitamente útil, pois envolve variáveis que podem não ser conhecidas pelo empreendedor. Os evolucionistas adotam o conceito de racionalidade procedural, onde a racionalidade dos agentes é imprevisível, pois é resultado do processo de aprendizagem ao longo das interações com o mercado, com

novas tecnologias e novos processos de inovação. Da mesma forma, a firma tem sua auto-organização como resultado das flutuações do mercado, rejeitando, assim, o seu equilíbrio. ROSENBERG (2006) diz que "o que chamamos de "pesquisa e desenvolvimento" (P&D) constitui um processo de aprendizagem na geração de novas tecnologias", sendo esse conceito de fundamental importância para os grupos de desenvolvimento de *softwares* de código aberto.

Existe uma parte de todo o conhecimento adquirido que tem aplicação direta nas melhorias de produto e/ou processo produtivo, facilitando para a firma o processo de auto-conhecimento e a formação de processos que facilitem a tomada de decisões em qualquer momento, com base em um conhecimento pré-estabelecido do mercado e de suas relações com os consumidores. Considerando o mercado de informática, mais especificamente a relação entre o usuário de computador com as empresas que criam os *softwares* de computador, esse auto-conhecimento e, principalmente, essa facilidade com que a empresa tenha para saber como o mercado irá reagir para suas inovações (considerando, claro, decisões corretas), é vital para o seu próprio crescimento e, em conseqüência, para o crescimento do próprio mercado.

NELSON & WINTER (2005) tratam as rotinas das atividades de determinada organização como sendo as mesmas a maneira mais importante de armazenar o conhecimento. As rotinas de uma organização são análogas às habilidades, que eles chamam de *skills,* de um indivíduo, definindo a seqüência de um comportamento coordenado que é efetivado para o alcance de determinados objetivos. Desta forma, implicam na prática de um comportamento que respeita determinados mecanismos estabelecidos pela empresa ao longo de sua evolução. O conceito de *skill*, assim como o de rotina, envolve programação, conhecimento tácito e envolve escolhas, que tem um caráter automático de comportamento que pode ser observado quando do desenvolvimento do *software* livre.

Assim, as rotinas da empresa afetam tanto atividades operacionais de curto prazo, quanto àquelas referentes às decisões de longo prazo, tais como o investimento e a aplicação de recursos em pesquisa e desenvolvimento. Os autores classificam as

rotinas como sendo de operação, que são usadas no dia-a-dia da organização, de investimento, que são as usadas em projetos ou construção de novas plantas, por exemplo, ou ainda, de transformação, ligadas aos processos de P&D e que dão origem às inovações.

Essas rotinas influenciam diretamente o comportamento das firmas, substituindo a coordenação hierárquica rígida, norteando as decisões dos indivíduos que conhecem seu trabalho, interpretando e respondendo corretamente as mensagens que recebem. Podem ser classificadas como estáticas, de simples repetição das práticas anteriores ou rotinas dinâmicas que permitem incorporar novos conhecimentos. O conhecimento tácito, não codificado, sendo, portanto, mais difícil de ser adquirido ou transferido, é um ativo específico, tido como base da diferenciação competitiva das organizações.

A atividade tecnológica também é constituída de rotinas, pois o processo de inovação é envolvido de grande incerteza e, ciente disto, a firma tem um comportamento mais defensivo. O próprio processo de mudança possui rotinas, limitando as opções técnicas da empresa como forma de amenizar as incertezas do processo. Pode-se dizer então, que os maiores ganhos que a inovação originará a empresa, de certa forma, também dependem das rotinas incorporadas em seu processo inovativo, conforme salientam NELSON & WINTER (2005, p. 134):

> "Como uma primeira aproximação, portanto, pode se esperar que as firmas se comportem no futuro de acordo com as rotinas que empregaram no passado. Isto não implica uma literal identidade de comportamento ao longo do tempo, na medida em que as rotinas podem estar associadas de formas complexas aos sinais do ambiente. Isso implica que é bastante inadequado conceber o comportamento da firma em termos de uma escolha deliberada de um amplo leque de alternativas, que algum observador externo considera serem as oportunidades "disponíveis" para a organização. O leque de alternativas não é amplo, mas estreito e idiossincrático; ele é construído a partir de rotinas da empresa, e a maior parte "deste ato de escolha" também é definida automaticamente por estas rotinas. Isto não significa que as empresas individuais não possam se constituir em sucessos brilhantes por um curto ou longo período; o sucesso e o fracasso dependem do estado do ambiente."

O progresso técnico é limitado por poucas alternativas, sendo assim, as rotinas determinam os procedimentos que devem ser tomados por elas no processo de busca das inovações. São levados em conta ainda, os fatores econômicos e técnicos que

NELSON & WINTER (2005) chamam de trajetória natural. A mudança de um paradigma abre novas oportunidades tecnológicas alterando o processo de busca de inovação por parte das firmas. Porém, nem todas as empresas possuem capacitação suficiente para participar deste processo, neste caso, dependem também da trajetória tecnológica pretérita e do caráter cumulativo do progresso técnico.

O caráter cumulativo é importante no processo de busca de inovação, pois implica em determinado nível de capacitação tecnológica, permitindo à empresa se propor a alcançar determinadas tecnologias, podendo ser desde melhoramentos técnicos, que viabilizam trajetória natural, ou um conjunto de conhecimentos científicos, técnicos e empíricos que constituem uma determinada tecnologia, ao mesmo tempo capacitando a empresa na busca de um padrão tecnológico superior.

Assim, conforme NELSON & WINTER (2005), a atividade tecnológica não gera apenas novos produtos ou processos, mas também uma série de conhecimentos que orbitam ao redor da inovação. Dessa forma, uma das grandes vantagens do *software* livre é a facilidade com que novos desenvolvedores possuem para buscar o conhecimento obtido com grupos em estágios mais avançados de desenvolvimento (comumente em *softwares* com versões já finalizadas), encurtando, e muito, um processo que, de maneira tradicional, ordenaria um tempo além do realmente necessário e a geração de um conhecimento muitas vezes redundante (no sentido de já ser conhecido, mas não divulgado), o que poderia ocasionar em um atraso que, muitas vezes, inviabilizaria a criação de um novo produto ou o encareceria de forma a diminuir sua chance de sucesso no mercado.

O processo de busca é composto por diversos tipos de comportamentos, que vão desde a imitação, constituindo na cópia de um determinado comportamento de uma firma pelos concorrentes e co-participantes de um mesmo tipo de atividade e ainda, os comportamentos intra e inter-industriais, onde as possibilidades técnicas da firma são determinadas dentro ou fora da mesma, respectivamente.

Conforme NELSON & WINTER (2005), são os mecanismos de seleção que vão determinar a mudança técnica, definindo, assim, o êxito de uma inovação. A firma

inovadora, objetivando a realização de lucros, atua com racionalidade limitada, utilizando-se de rotinas e mecanismos de busca, adotando estratégias que serão sancionadas por mecanismos de seleção, tanto mercantis quanto não-mercantis.

O caráter dinâmico dos processos de busca e seleção fica explicito, pois a inovação atua como fator gerador de mudanças constantes, utilizando-se das inovações técnicas passadas na criação de outras presentes e futuras. Desta forma, evidencia-se a mudança tecnológica como um fenômeno endógeno produzido pelo próprio processo competitivo, onde o mercado funciona como um fornecedor de retornos de conhecimento ao processo de geração, selecionando os desenvolvimentos mais prováveis, podendo obter um sucesso maior no mercado.

O que se pode observar no objeto de análise é que os diversos grupos que desenvolvem o *software* (nesse momento, independente do modelo que será utilizado para o seu desenvolvimento, uma vez que o mercado é, de modo geral, indiferente quanto ao seu modelo de desenvolvimento) se utilizam do próprio mercado e das suas próprias evoluções nos seus produtos para gerar produtos ainda melhores. Basta ver como eram os *softwares* criados e desenvolvidos na década de 1980 e como diferem dos atuais, tanto em sua forma (desde a interação entre o usuário e o computador, até mesmo a evolução visual desse período) quanto à sua essência (*softwares* com funções que sequer eram imaginadas antigamente. Inclusive, *softwares* que sequer eram imaginados).

Quanto à diferenciação entre as empresas, NELSON & WINTER (2005), trata-na como sendo uma situação normal da economia capitalista, pois as estratégias bem sucedidas geram organizações bem sucedidas, as vencedoras. Por outro lado, as outras que fracassaram, serão perdedoras. Sendo assim, as vencedoras poderão ter margens de rentabilidade acima da média do setor em que atuam, permitindo um diferencial de crescimento, proporcionando as economias de escala e o aumento de participação no mercado. À medida que o processo de mudança técnica condiciona a estrutura de mercado, a estrutura de mercado influi no processo de busca e seleção das inovações.

De fato, a diferenciação entre os diversos grupos de desenvolvimento de *software* (novamente, sem diferenciar os modelos de desenvolvimento) acabam causando uma clara diferenciação entre os produtos no mercado, gerando uma rentabilidade maior para as empresas mais competentes, as vencedoras. Como exemplo, é importante citar os *softwares* para *internet Netscape* e *Internet Explorer*, que, tendo batalhado na segunda metade da década de 1990[1], acabou gerando a hegemonia da *Microsoft*, com o *Internet Explorer* presente em aproximadamente 92% dos computadores com acesso à internet em 2001[2]. Mesmo que esse monopólio tenha sido, aparentemente, nulo (uma vez que o *software* era distribuído gratuitamente), isso ajudou com que a *Microsoft* obtivesse ganhos financeiros através de outras áreas, com a utilização do mesmo *software*. Só o fato de que quase todos os computadores do mundo tinham apenas um navegador, fazia com que a *Microsoft* estivesse em evidência cada vez que um determinado endereço na *internet* fosse visitado, uma vez que o *software* que dava acesso a esse endereço tinha o logotipo da empresa que o desenvolveu.

Assim, conforme NELSON & WINTER (2005) descreve, uma mudança técnica através dessa forma de P&D faz com que as firmas (ou os grupos de desenvolvimento de *software*) satisfaçam os seguintes pressupostos:

Satisfação: A criação de mudanças técnicas deve ser obtida, através da busca pela satisfação, seja da empresa ou do consumidor, e não somente por uma política de desenvolvimento de idéias. Assim, as respostas dadas pelos usuários de programas de computadores constituem um dos principais meios de desenvolvimento do código fonte, buscando assim a satisfação da empresa (seja o lucro propriamente dito, ou um "lucro social", tendo em vista o aprendizado e o benefício gerado por essa evolução) e a satisfação do usuário final, uma vez que as melhorias dos produtos gera uma satisfação maior daqueles que irão fazer uso desse programa.

Imitação: Um grupo de desenvolvimento pode usar soluções utilizadas por outros grupos para obter um resultado melhor em seu próprio *software*. Esse processo

---

1 Período conhecido como "Guerra dos Navegadores", que durou de 1995 a 1999

2 http://pt.wikipedia.org/wiki/Imagem:Browser_Wars.svg

de integração entre diferentes grupos tem auxiliado o desenvolvimento do *software* livre, uma vez que respostas para determinados problemas devem ser adaptadas (desde que haja a devida autorização do grupo que desenvolveu a solução inicialmente) e não desenvolvida, poupando tempo e recursos (sejam financeiros ou humanos), facilitando que determinado projeto possa evoluir muito mais rapidamente.

Assim, conforme NELSON & WINTER (2005, p. 344):

> "De uma perspectiva evolucionária, o crescimento econômico em qualquer economia, desenvolvida ou menos desenvolvida, seria considerado um processo de desequilíbrio que envolve uma combinação de firmas que empregam diferentes 'safras' de tecnologias. Essas tecnologias se modificam ao longo do tempo. (...) A qualquer tempo, as diferenças entre países podem ser explicadas pelas diferenças das combinações de tecnologias utilizadas, assim como pelas proporções de fatores."

Dessa forma, o processo de inovação tecnologia (nesse caso, o processo de criação de aplicativos para computadores), passa a ser importante não só localmente, como também globalmente, podendo diferenciar os países que possuem grupos organizados na criação de aplicativos. Nesse aspecto o Brasil possui local de destaque entre os grupos de desenvolvimento de *software* livre, sendo um dos países com maior aceitação de soluções desenvolvidas dessa forma. Os grupos de desenvolvimento que se reúnem em Porto Alegre, Rio Grande do Sul, possuem certo grau de dinamismo que não são encontrados com freqüência em outros países.

1.2 PARADIGMAS E TRAJETÓRIAS TECNOLÓGICAS SEGUNDO O MODELO DE DOSI

O processo inovativo é caracterizado pela incerteza, altos custos e longo prazo de maturação, sendo a incerteza um dos elementos centrais do conceito de inovação. DOSI (1990) descreve a inovação como processos de aprendizado e descobrimento sobre novos produtos, novas formas de organização econômica anteriormente inexploradas.

Sobre as estratégias adotadas pelas empresas, DOSI (1984) comenta que elas

estão limitadas pela estrutura do próprio setor e do regime tecnológico presente na mesma. Ambos irão definir até onde a empresa pode agir, Assim é possível acompanhar o surgimento da necessidade de transformação das tecnologias através de novos paradigmas tecnológicos.

O autor refere-se à existência de três grandes sistemas: o científico, o tecnológico e o econômico, e focaliza nesta distinção a análise das relações entre o sistema tecnológico e as variáveis econômicas. A tecnologia engloba não somente os elementos materiais, mas também o conhecimento e a experiência adquirida ao longo do tempo. Assim, a empresa parte de conhecimentos já adquiridos para buscar melhorias e desenvolvimento, suas possibilidades dependem do que já se fez no passado, por isso o progresso técnico é considerado cumulativo. A tecnologia também depende dos mecanismos de busca e aprendizagem para melhoria da eficiência e desenvolvimento do processo produtivo.

DOSI (1984) apresenta conceitos de paradigmas e trajetórias tecnológicas que oferecem uma referência de hierarquização da importância dos diferentes âmbitos no processo de inovação, definindo assim, um paradigma tecnológico como padrão de soluções para problemas tecnológicos, baseando-se em princípios das ciências naturais e em materiais tecnológicos.

A direção do progresso técnico aparece como solução dos problemas e necessidades que se apresentam no decorrer do tempo dentro de um paradigma, enfocando os esforços na exploração das oportunidades tecnológicas que este oferece e, exercendo um poderoso efeito de exclusão com respeito às outras possibilidades alternativas. De acordo com o exposto, pode-se definir como sendo a trajetória tecnológica, o padrão de solução normal dos problemas dentro de um paradigma tecnológico. O seu progresso é o resultado das melhorias dos *trade-offs* entre as dimensões que o configuram: campo de aplicação, tecnologia material, propriedades físico-químicos, dimensão tecnológica e econômica.

Um novo paradigma tecnológico surge das novas possibilidades oferecidas pela ciência, porém, depende também das estruturas econômicas e institucionais.

Segundo DOSI (1982), a configuração de um novo paradigma é estabelecida por um amplo conjunto de possibilidades de direção, até colocar-se os novos produtos na linha de montagem, ocorrem várias etapas de refinamento através das determinações econômicas e sociais, que caracterizam o novo paradigma.

Assim um novo paradigma corresponde em primeiro lugar a um modelo de resolução de certos problemas econômicos e/ou tecnológicos selecionados, fundado sobre princípios altamente selecionados, derivados das ciências naturais e das tecnologias materiais, conjuntamente com regras específicas, concebidas para adquirir novos conhecimentos e os preservar, DOSI (1982, 1988). Em outros termos, um paradigma tecnológico permite determinar ao mesmo tempo, o campo da pesquisa, os problemas apresentados, os procedimentos a utilizar e as tarefas a efetuar para resolver esses problemas.

No caso do processo de inovação, ou mesmo da própria inovação, será considerado como paradigma se ele respeitar as três condições a seguir, conforme DEPRET *et* HAMDOUCH (2000):

a) Primeiramente ele deve aparecer mais fecundo, mais simples, mais coerente, mais preciso, sobretudo mais eficaz para resolver os problemas não resolvidos pelos métodos mais ou menos racionalizados do paradigma dominante, conforme HERAN (1998);

b) Em segundo lugar, as novas potencialidades devem induzir uma reconfiguração estratégica e organizacional radical;

c) Em terceiro lugar, ele deve induzir as firmas do setor a adotá-lo sob pena de desaparecer.

No que tange a primeira condição, considerando que o *software* livre cumpre as mesmas funções que o proprietário, mas que contém um número menor de linhas de código[3], e resolve problemas técnicos das empresas com um potencial maior[4] que os

---

[3] O *Kernel* do Linux (2.6.0) possui 5.929.913 linhas de código, e o *Microsoft Windows Vista* possui aproximadamente 50.000.000 de linhas de código. Fontes: http://tecnologia.uol.com.br/ultnot/2007/01/29/ult4213u21.jhtm e http://linux-

produzidos por código proprietário, o *software* livre atende, e bem, esse primeiro aspecto da inovação tecnológica.

É importante citar também que as opções que existem para substituir os aplicativos proprietários existentes no mercado são variadas, e muitas vezes cumprem a mesma função com modos de funcionamento mais eficientes e com menor uso de recursos[5] do sistema.

Além dessa melhoria em relação ao desempenho, os *softwares* de código aberto possuem uma facilidade maior em relação ao uso. Funções de difícil acesso ao sistema *Windows*, por exemplo, estão habilitadas com facilidade no *Linux*. A interface de uso do sistema *Linux* com utilização da área de trabalho proveniente do *Kde*[6] possui ferramentas que somente os desenvolvedores da *Microsoft* possuem, como a opção de alterar, radicalmente, o ambiente de trabalho, além de proporcionar uma interação maior com os *softwares* presentes no computador, como o *K-Office*, a versão integrada do *OpenOffice* com o *Kde*, opção inexistente nas versões do sistema operacional *Windows* com o *software Microsoft Office*, ambos desenvolvidos pela mesma empresa. Opção essa inexistente até para empresas que contratem os serviços da *Microsoft* e obtenham certa exclusividade no desenvolvimento de certas ferramentas e opções de *software* e/ou serviços.

b) Em segundo lugar, as novas potencialidades devem induzir uma reconfiguração estratégica e organizacional radical;

A estrutura organizacional das empresas que desenvolvem *softwares* sofre uma variação radical quanto ao modo de funcionamento dos grupos responsáveis pelo desenvolvimento e criação do código fonte. As empresas que se utilizam do modelo de *software* proprietário para o desenvolvimento, por relações legais de vinculação empregatícia, possuem uma hierarquia que não permite uma mudança radical sem que

---

fedora.org/portal/mcdules/newbb/viewtopic.php?topic_id=79&forun=35

[4] Considerando que a empresa possui mais opções para configurar e até desenvolver certas ferramentas

[5] A instalação do *Microsoft Office* 2007 utiliza aproximadamente 2GB (Gigabytes) de instalação no disco rígido, enquanto a instalação do *BrOffice* 2.4 utiliza aproximadamente 1/7 desse total, ou seja, 300 MB (Megabytes)

[6] "O KDE é um ambiente de desktop moderno para sistema UNIX", conforme: http://twiki.softwarelivre.org/bin/view/KdeBR/OQueEKde visita em 07/06/2008

exista algum modo de processo administrativo demorado e muitas vezes dispendioso. Há também uma questão física a respeito dessas empresas, uma vez que havendo um vínculo empregatício entre os programadores responsáveis por modificações no código fonte, há uma necessidade de se manter esses funcionários em escritórios ou em ambientes de trabalho comuns entre os programadores. Mesmo empresas que possuam diversos escritórios, até mesmo em países diferentes (por exemplo, a *Microsoft* possui escritórios em 102 países[7]) necessitam de um espaço físico comum para facilitar a comunicação entre os diversos programadores responsáveis pela criação e evolução do código fonte.

Enquanto isso, os grupos que se reúnem para desenvolver *software* de código aberto (ou *software* livre) possuem um modelo hierárquico que permite uma modificação rápida e, muitas vezes, sem custo. Uma vez que raramente há um vínculo legalizado entre o(s) mantenedor (es) e os colaboradores, a modificação pode ser feita sem qualquer limitação quanto à formação legal e/ou financeira dos profissionais que fazem parte desse desenvolvimento. Além dessa facilidade, os colaboradores dos projetos de *software* livre podem trabalhar em casa, sem a necessidade de que todos se localizem em um mesmo espaço. De fato, projetos grandes inviabilizariam ou dificultariam muito essa obrigação de manter os programadores em um mesmo espaço comum.

c) Em terceiro lugar, ele deve induzir as firmas do setor a adotá-lo sob pena de desaparecer.

Nos últimos anos a adoção do *software* livre nas grandes empresas do Brasil tem aumentado. Conforme pesquisa do Instituto Sem Fronteiras (ISF), aproximadamente 73% das grandes empresas brasileiras utiliza de alguma solução com base em *software* livre[8], por terem um poder financeiro maior para evitar o uso de *software* pirata, além de contarem com um conhecimento maior que possibilite a manutenção e o uso eficiente dessas ferramentas desenvolvidas em grupos de *software*

---

[7] http://pt wikipedia.org/wiki/Microsoft, visita em 08/06/2008

[8] http://idgnow.uol.com.br/computacao_corporativa/2008/02/27/software-livre-e-usado-em-73-das-grandes-empresas-no-brasil/paginador/pagina_2 , visita em 07/06/2008

livre.

Conforme a notícia vinculada no parágrafo anterior, "os sistemas operacionais baseados em *software* livre se adéquam às especificações e expectativas técnicas das áreas em que há grande volume de transações e processamento de dados, bem como armazenamento, segurança, interoperabilidade e disponibilidade são, portanto, essenciais".

Além disso, 48% das empresas que utilizam o *software* livre usam-no em operações críticas, ou seja, operações onde o volume e o processamento de dados se dão de forma urgente sem admitir margem de erro, exigindo um sistema robusto e, principalmente, confiável.

1.3 O PROGRESSO TÉCNICO EM ROSENBERG

Segundo ROSENBERG (2006), há uma dificuldade da teoria econômica em definir exatamente o que é o progresso técnico, uma vez que não há apenas uma, mas várias, definições acerca desse assunto. A melhor definição sobre o progresso técnico, segundo ROSENBERG (2006) é do "progresso técnico compreender certos tipos de conhecimento que tornam possível produzir a partir de uma quantidade de recursos, (1) um maior volume de produto ou (2) um produto qualitativamente superior".

Conforme nota-se durante o início do desenvolvimento do *software* livre em seus primórdios (para facilitar a compreensão desse assunto, mais exatamente quando o sistema operacional *Linux* foi inicialmente desenvolvido, em meados de 1992), até hoje, houve um desenvolvimento contínuo de melhorias no seu código fonte inicial, gerando um produto muito mais robusto, completo e eficiente.

Depois de iniciado o processo de progresso técnico, ROSENBERG (2006) conclui que quem estuda esse fenômeno tem negligenciado a questão de "quem fez primeiro?", sendo essa questão essencial para os fenômenos inventivos da história.

Assim, ROSENBERG (2006, p.44) diz que:

> "O processo de difusão, via de regra, depende de uma seqüência de melhoramentos nas características de desempenho de uma invenção, de sua modificação e adaptação graduais para adequar-se às necessidades ou demandas específicas de vários nichos de mercado e da disponibilidade e introdução de outros insumos complementares que tornam mais útil uma invenção original."

Dessa forma, a difusão do conhecimento desse progresso técnico é de fundamental importância para o sucesso obtido pelos grupos de desenvolvimento de *software* livre nos últimos anos, tendo como resultado o surgimento de programas com qualidade suficiente para competir (de igual para igual, em determinados casos) com os *softwares* desenvolvidos pelas empresas que até então detinham o monopólio (caso da *Microsoft*, com seus *softwares Windows, Office* e *Internet Explorer*) desses mercados.

Especificamente, em se tratando de operações críticas (operações bancárias, extração de petróleo, por exemplo), alguns *softwares* livres acabaram com o monopólio existente, sendo considerados como opções tão, ou mais qualificadas que os modelos fechados[9]. Dessa forma, aplicações que, dez anos atrás, não eram largamente conhecidos, acabaram se desenvolvendo com o auxílio do conhecimento gerado pelo seu uso por profissionais da área (nesse caso, do mercado bancário) e, com diversas e freqüentes evoluções, tanto pequenas quanto revolucionárias, acabaram por criar um *software* (ou um modelo) que facilitou, e muito, tanto o acesso, quanto a presença desses aplicativos em ambientes que, até então, eram de exclusividade de *softwares* de código fechado.

Após ter dado existência ao progresso tecnológico de determinado produto, e após a aceitação desse produto pelo mercado, há o que ROSENBERG (2006) chama de "aprendizado pelo uso", ou seja, o uso contínuo de determinado produto causa uma melhoria no conhecimento existente sobre aquele produto, facilitando a criação de novos processos de inovação ou um aceleramento desse processo, causando assim um aprendizado mais eficiente e uma competitividade maior dessa firma, com taxas de

---

[9] No próximo capítulo haverá uma melhor explanação a esse respeito

lucro maiores e uma facilidade maior em desenvolver determinados processos inovativos.

1.4 OS SETORES PRODUTIVOS EM PAVITT

PAVITT (1984) revela que o processo inovativo da firma considera as mudanças técnicas dependentes de certos aspectos que são: (a) a fonte de tecnologia do processo, sendo internas ou externas, (b) o peso relativo dessas inovações e das inovações dos produtos, (c) a dimensão das empresas que geram essas inovações e (d) a intensidade dessas inovações, ou seja, qual o impacto que essa diversificação tecnológica causa nesse mercado.

No que se refere às inovações de processos de criação de *softwares*, a mudança do sistema de produção do código fonte do modelo proprietário para o livre causou uma completa revolução na maneira com que as empresas devem lidar com esse modo de funcionamento atualizado. Esse novo modelo de criação do código fonte veio também a influenciar o modo como as empresas que se utilizam do modelo proprietário de desenvolvimento, uma vez que certas soluções utilizadas por essas empresas foram baseadas em soluções apresentadas por grupos que desenvolvem *software* livre.

PAVITT descreve os modos de analisar diferentes companhias de acordo com modelos pré-estabelecidos para gerar um conceito dessas empresas e seu impacto em seu mercado no que se refere às suas fontes, natureza e impacto de suas inovações tecnológicas, considerando assim quatro grupos:

1) Setores dominados por fornecedores: As inovações tecnológicas nesses mercados são menores e mais impactantes que outros mercados, tendo como principal fonte de inovação empresas de outros setores. "Se trata, portanto, de uma trajetória tecnológica baseada em reduções de custos, entre uma escolha entre as técnicas oferecidas por outros setores". PAVITT (1984).

As principais inovações geradas nesse setor são por processos informais de aprendizado, tanto interno (dentro da empresa) como externo (no próprio setor), através do simples 'aprendizado pelo uso', existindo dessa forma um alto grau de interdependência entre o produtor e o consumidor, uma vez que o usuário final do produto irá, de diversas maneiras, dar um retorno ao produtor em relação a alterações necessárias nos produtos, quanto o produtor irá buscar melhorias nesses produtos tendo por base o que obtiver de resultado no mercado. Assim, essa relação entre esses dois elementos econômicos irá gerar a dinamização necessária para que as empresas desse setor produtivo sejam capazes de gerar inovações tecnológicas e para que os produtos sofram modificações necessárias para que continuem, à sua própria maneira, atraentes para que haja uma demanda desses produtos.

2) Setores com alto grau de economias de escala: As principais atividades produtivas das empresas desse setor se darão de forma a existir uma grande dependência de capital, com a produção de itens com um alto grau de complexidade.

Os produtos gerados por esse setor geralmente possuem sazonalidade, podendo ser reconhecidos pelo processo de produção em massa, com uma enorme importância das economias de escala, tendo como exemplo as empresas que trabalham na produção de automóveis. A tecnologia dessas empresas é gerada através de processos altamente evoluídos de pesquisa e desenvolvimento, tendo, geralmente, seus setores de P&D desenvolvidos e com o uso de recursos financeiros abundantes para seu funcionamento.

As tecnologias criadas por esses setores podem ser tanto de processos, com o barateamento ou evolução em seus parques produtivos, como de produtos, com mudanças (às vezes drásticas) em seus produtos finais. Similar ao setor anterior, as empresas desse grupo são também dependentes do 'aprendizado pelo uso', uma vez que seus produtos finais estão sempre sob análise, mesmo após estarem disponíveis no mercado, sofrendo assim mudanças constantes em suas formas e graus de utilização. A diferenciação deste setor com o anterior se dá na total dependência de um setor de pesquisa e desenvolvimento forte tanto na geração de inovações tecnológicas quanto

na sua difusão.

Os principais exemplos desse setor, além da indústria automobilística é a siderurgia, metalurgia e papel.

3) Fornecedores especializados: É o setor que mais contribui para as inovações tecnológicas apresentadas no setor 1, uma vez que suas inovações tecnológicas são aplicadas principalmente em produtos, que são destinados a outros setores. Geralmente são empresas de porte pequeno, tendo um alto conhecimento para criar e produzir equipamentos.

As empresas de engenharia mecânica geralmente possuem esse perfil, tendo um alto grau de conhecimento em processos mecânicos, facilitando a existência de inovação tecnológica.

4) Setores baseados em ciência: As firmas desse setor estão ligadas as trajetórias dos paradigmas tecnológicos existentes e no progresso das ciências que venham influenciar essas trajetórias, em vista principalmente da importância dos setores de P&D dessas empresas.

Em se tratando das empresas que produzem *software*, incluindo as comunidades que criam aplicativos com código livre, muitas vezes não há um setor propriamente dito de pesquisa e desenvolvimento, mas esse processo de busca e aprendizado ocorre normalmente com o uso do aplicativo, e assim as melhorias acabam sofrendo importantes modificações sem que haja uma formalização desses setores de pesquisa.

Porém, mesmo que não haja essa ligação formal, os grupos que desenvolvem *software* (tanto proprietário quanto livre) estão totalmente ligados em oportunidades tecnológicas, uma vez que as possibilidades de apropriação de conhecimento através desse processo mais simplificado de pesquisa e desenvolvimento, pela estrutura dinâmica dos seus processos criativos.

Não há, nesse setor, um modelo simples de difusão de conhecimento, nem mesmo da criação desse desenvolvimento, diferentemente dos outros setores, que

possuem ferramentas mais 'simples' de análise.

Dessa forma, PAVITT (1984) contribui para o arcabouço teórico utilizado no presente estudo, uma vez que os setores descritos anteriormente, e suas formas de difusão do conhecimento adquirido dão uma ferramenta de análise de mercado e, em última instância, uma visão clara das empresas que produzem aplicativos para computadores, facilitando a análise do fenômeno que será estudado mais aprofundadamente no próximo capítulo.

Entre NELSON & WINTER passando por DOSI, PAVITT, ROSENBERG, DEPRET e outros, as visões destes modelos possuem pontos de convergências na teoria evolucionista ao considerar o processo de inovação como o processo que se constrói a partir da acumulação de conhecimento no seio da firma e em setores específicos da economia como é o caso de PAVITT. Considerando também as mudanças técnicas que o *software* sofreu e nos últimos anos, desde o seu método de criação, quanto seu modo (tanto interno quanto externo) de funcionamento, as teorias apresentadas por esses autores servem, com perfeição, para a análise do fenômeno estudado. As recentes mudanças notadas no modelo de criação de aplicativos para computadores, tal como a evolução dos editores de texto, por exemplo, comprovam que essas teorias são ideais para explicar o fenômeno estudado.

No próximo capítulo será analisado o *software* livre como um fenômeno econômico revestido de inovação técnica de acordo com a teoria evolucionista até o momento estudada.

2. OS MODELOS DE DESENVOLVIMENTO DE *SOFTWARE*, SUAS VANTAGENS E DESVANTAGENS NO MERCADO CONCORRENCIAL E AS FORMAS DE DIFUSÃO DO CONHECIMENTO ADQUIRIDO

O objetivo do capítulo é estudar, em um primeiro momento, o que é *software*, e em um segundo momento, como ele é desenvolvido tanto na ótica das comunidades de código aberto quanto nas empresas que se utilizam do código proprietário, para em um terceiro momento será estudado como se dá a disseminação do conhecimento nas comunidades de criação do *software* de código aberto, e como essa disseminação ocorre nas empresas que utilizam o código proprietário.

Em 1944 foi construído o MARK 1, nas dependências da Universidade de Harvard[10], considerado como o primeiro computador eletromecânico do mundo, tendo sido o início da era da informática. Em 1975 os estudantes William (Bill) Gates e Paul Allen criaram um dos primeiros *softwares* da história e alguns anos depois fundaram a *Microsoft*.

A partir do início da década de 80, os computadores passaram a ser produzidos em larga escala e chegaram ao mercado doméstico nos Estados Unidos, sendo incorporados em outros mercados com o passar de poucos anos. Quando os primeiros computadores foram vendidos sem um sistema operacional, os compradores passaram a ser também entusiastas de códigos de programação, uma vez que, para cumprir as funções que esses usuários pretendiam nesses equipamentos, era necessário gerar o seu próprio código fonte, já que não existia qualquer fornecedor desses códigos.

A partir da década de 90, o valor necessário para se adquirir o *software* já ultrapassava o valor do *hardware* necessário para executá-lo. Com isso o desenvolvimento de aplicativos para computadores, seja o próprio sistema operacional ou seus programas tornou-se mais lucrativo para as empresas do que a fabricação do próprio computador.

Com isso surgiram gigantes no ramo do *software*, onde se destacam a *Microsoft* e a *Apple*, com seus logotipos e programas conhecidos e difundidos

---

[10] http://www.widesoft.com.br/users/virtual/parte1.htm

mundialmente, e com seus preços elevados para o usuário final.

Em 15 de maio de 2008, os *softwares* necessários para o funcionamento de um microcomputador poderiam ser adquiridos a um preço de venda de aproximadamente R$ 1.977,00. Considerando a versão mais simples do sistema operacional *Windows Vista*, com preço de R$ 589,00[11], a versão padrão (*standart)* do *Microsoft* Office 2007, por R$ 1.299,00[12], e uma versão de um *software* Anti Vírus, por R$ 89,00[13], sem considerar outras opções de segurança para um microcomputador de uso doméstico ou até para uso em um pequeno escritório. Se forem consideradas opções para compra de *softwares* específicos para engenharia (*AutoCAD),* edição de imagem (*photoshop)* ou até mesmo edição de áudio (*Sony SoundForge),* somente o valor do *software* pode ultrapassar, em muito, o valor de um microcomputador com o mínimo da capacidade necessária para se executar esses mesmos aplicativos.

Por outro lado, as versões de *softwares* geradas através do desenvolvimento com o código aberto são encontradas de graça, sem a obrigação da assinatura de um contrato de manutenção ou uma licença de uso, tanto para o mercado corporativo quanto doméstico, com as mesmas funções e qualidades dos aplicativos pagos, sendo considerados como opções equivalentes àqueles que só são obtidos após o processo de compra, tendo como opção aos *softwares* citados acima o *Gimp* (edição de imagem), o *OpenOffice* ou sua versão brasileira, o *BrOffice* (suíte de escritório), *Qcad* (engenharia) ou *Audacity* (edição de som).

Assim, é necessário que, antes que sejam explicadas as formas de criação do desenvolvimento do *software*, seja ele aberto ou fechado, é necessário que haja um aprofundamento a respeito de "o que é *software*?"

2.1 – *SOFTWARE*

O objetivo deste tópico é estudar o que é *software* e quais são os componentes

---

11 http://www.americanas.com.br/prod/595490/Catalogo?chave=Expo_Home_Info

12 http://www.americanas.com.br/prod/595511/Catalogo?chave=Expo_Home_Info

13 http://www.americanas.com.br/prod/522957/Catalogo?i=1&chave=Expo_Home_Info

básicos que o formam, através do uso de um algoritmo[14] estruturado, incluindo a compilação de uma linguagem de programação, que irá gerar o *software* final, ao que o código foi proposto e desenvolvido.

Segundo KING (1989), os programas para microcomputadores se dividem em três tipos, os *softwares* de sistema operacional, os *firmwares* e os *softwares* aplicativos.

O *software* de sistema operacional (ou operativo) é o responsável por administrar as funções básicas de um microcomputador e o prepara para que o mesmo cumpra o objetivo que tenha sido proposto ao seu uso, seja em ambiente doméstico ou corporativo. "Sem programas o computador é apenas uma máquina inerte, uma coisa inútil". (KING, 1989). O sistema operacional cumpre o papel de iniciar o funcionamento e oferecer aos *softwares* aplicativos uma plataforma onde estes possam cumprir as funções definidas e programadas durante o seu processo de desenvolvimento.

Como exemplo de *softwares* de sistema operacional, pode-se citar as diversas versões da família *Windows*, da *Microsoft* como modelo de *software* proprietário e as diversas distribuições do *Linux* como modelo de código aberto. A relação de *softwares*, tanto proprietários quanto livres, para os mais diversos usos de um microcomputador, é imensa. Desde sistemas básicos de segurança, como aplicativos antivírus, passando por visualizadores (tanto de vídeo, como o *Windows Media Player*, áudio, como o *Winamp*, arquivos de texto, como o *Acrobat Reader*), ou editores de arquivos (*Microsoft Word*, *Microsoft Excel*, por exemplo), é impossível citar os mais variados exemplos de *softwares* disponíveis no mercado. Para evitar que sejam cometidas injustiças e erros de análise, todos os *softwares*, seja para o uso que tiverem, serão denominados apenas como *software*, aplicativo ou programa, tendo em vista que o modelo de desenvolvimento é, sob qualquer aspecto, o mesmo, seja um sistema operacional, seja um editor de texto. A diferenciação se dará no mercado, diferenciando a faixa de consumo, o valor de venda ou outros itens que não são, nesse

---

[14] Um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações (FARRER, 1999)

momento, relevantes.

É importante definir também como as diversas peças de hardware se comunicam com os diversos aplicativos presentes em um microcomputador. Assim, é necessária a seguinte definição de *Firmware* que, conforme KING (1989) é um "tipo operacional de *software* de sistemas operacionais, que é suprido juntamente com o computador. Seu propósito é adaptar o computador a um determinado conjunto de requisitos de um usuário ou de um grupo de usuários". Em outras palavras, os microcomputadores são formados por diversas peças de *hardware*, que irão prover aos *softwares* o poder de processamento e desempenho para cumprir seus objetivos. O *software firmware* é o responsável pelo correto funcionamento dessas diversas peças e do reconhecimento do poder e do tipo da peça em sua integração com o sistema operacional.

O *software driver* é produzido pelo fabricante de determinado item de *hardware*, e, embora seu desenvolvimento se dê de maneira semelhante aos sistemas operacionais e aplicativos, não serão estudados casos de *drivers* neste trabalho, uma vez que o desenvolvimento se dá apenas pela empresa que fabrica determinado *hardware*. Para que esses itens de hardware sejam reconhecidos pelo sistema operacional e, portanto, ofereça ao usuário todas as opções possíveis, se faz necessário o desenvolvimento de um *driver* de dispositivo, ou seja, uma função específica de *software* que irá efetuar a ligação efetiva entre o sistema operacional e o *hardware* do computador.

Embora o *software firmware* e os *drivers* não sejam focos de análise do presente estudo, os mesmos entrarão em algum momento na análise do *software*, considerando que o seu desenvolvimento é similar aos *softwares* tradicionais podendo ser, portanto, desenvolvidos através dos modelos estudados.

O *software* aplicativo é o programa que irá cumprir as diversas funções aos qual o microcomputador pode ser programado. Qualquer função que seja utilizada em um microcomputador será executada por um *software* aplicativo, e este terá algum grau de integração com o sistema operacional, para melhor utilizar os recursos de *hardware* disponível no computador.

A construção e o desenvolvimento dos três tipos de *software* descritos acima se dão da mesma forma, sendo diferenciado somente o objetivo final do programa. O objetivo do presente estudo é analisar apenas a forma com que o *software* é desenvolvido, e, portanto, não será feita qualquer análise do objetivo final do *software,* seja ele o sistema operacional, *firmware* ou *driver* de dispositivo.

Há, portanto, um material físico no microcomputador (*hardware)*, que, para cumprir suas funções, necessita de um *firmware* para que as diversas placas cumpram suas funções. E para que o sistema operacional e os diversos aplicativos se comuniquem com essas placas, existe o *driver*, que é o responsável pela interligação entre os *firmwares* e o sistema operacional/aplicativos. Importante lembrar ainda que os *drivers* de dispositivos podem ser criados pela comunidade, mas os *firmwares* não, uma vez que é de responsabilidade do fabricante de determinada peça de *hardware* a criação desse *software*.

Com a definição do que é *software*, será analisado então como ele é criado, quais são os métodos de desenvolvimento e sua integração com o usuário final ou com os usuários-programadores que venham a ter acesso ao mesmo bem como as diferenças entre as duas formas de desenvolvimento consideradas no presente estudo.

Segundo FARRER (1999), para que seja criado o *software*, é necessário o uso de algoritmos estruturados e linguagens de programação para se obter o código fonte do aplicativo, e a partir desse código, o *software* propriamente dito, utilizado pelo usuário final.

Algoritmo é definido como sendo "um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações" (FARRER, 1999), sendo geralmente destinado a resolver determinado problema através de um padrão de comportamento que deve ser obedecido para que o *software* possa iniciar um determinado processo e finalizá-lo com a obtenção do resultado a que se propõe.

Com o passar dos anos e através de vários processos da inovação tecnológica, os computadores aumentaram sua capacidade de processamento revelando uma verdadeira mudança de trajetória tecnológica no sentido de DOSI. Os algoritmos simples utilizados na década de 50, quando os computadores passaram a ser

produzidos em escala industrial, são muito complexos e estão além do limite da compreensão humana, embora ainda sejam criados pelo ser humano. Com isso se dá a necessidade do desenvolvimento estruturado dos algoritmos, que é uma forma de sistematização do algoritmo para ajudar na resolução de grandes e complexos problemas relacionados ao desenvolvimento do código fonte (FARRER, 1999).

Para que este algoritmo seja armazenado na memória de um computador, ele deve ser escrito em uma linguagem que o computador possa "entender", para que o *software* cumpra as funções propostas pela equipe de desenvolvimento do código fonte. Para isso é utilizada uma linguagem de programação, ou seja, a linguagem do computador, totalmente numérica, expressa por letras e números para facilitar a compreensão do ser humano e facilitar o seu desenvolvimento. A primeira linguagem simbólica é conhecida também como *Assembler* ou Linguagem Montadora, que nada mais é que a "tradução" de determinados caracteres em uma linguagem de programação em linguagem de computador (FARRER, 1989).

Segundo KING (1989, p. 3):

> "As funções comerciais, matemáticas, científicas, industriais, e até sociológicas e legais, envolvendo manipulação de dados, são representadas por conjuntos de instruções codificadas em notações conhecidas como linguagem de programação de computadores e alimentadas na memória do computador no momento apropriado de forma a permitir que o computador execute a manipulação quando necessário."

O sucesso do *Assembler* provocou o desenvolvimento de novas linguagens em que as notações matemáticas eram desenvolvidas se utilizando de "linguagem humana", usando-se a língua inglesa, por sua predominância, e deixando ao próprio computador a incumbência de traduzir essa linguagem para o computador através de um *software* compilador (FARRER, 1999).

Compilador, portanto, é um *software* responsável pela tradução da linguagem de programação, seja qual for, para a linguagem do computador. Além disso, para o desenvolvimento do código fonte, é necessário também o uso de um *software* interpretador, ou seja, que interprete os comandos e execute as instruções que estão nele contido (FARRER, 1999).

O passo mais importante no desenvolvimento do código fonte que irá gerar o *software* é a criação de um algoritmo adequado e lógico, que irá gerar um código estruturado para que o *software* cumpra as funções pelas quais foi desenvolvido de forma rápida e eficiente, sem que haja falhas de programação que venham a causar um comportamento não esperado que venha a causar a perda de dados ou até mesmo eventuais danos ao *hardware*.

Conforme foi visto anteriormente, o *software*, comumente chamado de programa de computador, aplicativo ou *software* aplicativo, é uma seqüência lógica de instruções a serem seguidas e/ou executadas com o objetivo de obter dele um determinado resultado no menor espaço de tempo, com menor custo, e maior eficiência possível.

O conjunto de instruções que forma o *software* é também conhecido como código fonte, que após ser compilado de acordo com uma determinada linguagem de programação, gerará o *software* a ser utilizado.

Os *softwares* atualmente são desenvolvidos para diversos dispositivos, desde os tradicionais microcomputadores, *notebooks*, aparelhos celulares, etc. Tendo o objetivo de facilitar o compreendimento do fenômeno, serão analizados aqui apenas os *softwares* criados e desenvolvidos para uso em micro computadores, sem distinção entre o mercado corporativo ou doméstico, desconsiderando, portanto, aqueles desenvolvidos para qualquer outro tipo de dispositivo.

É necessário explicitar a possibilidade de qualquer pessoa gerar o *software* desde que possua o código fonte original, mas que, utilizando-se apenas do *software*, ou seja, o aplicativo final, não é possível executar um processo de engenharia reversa, não sendo possível obter o código fonte original.

2.2 – OS MODELOS DE DESENVOLVIMENTO DE *SOFTWARE*

Após ter identificado o que é *software* e a compreensão de que, independente do modelo utilizado em seu desenvolvimento, o produto final é gerado através de um código fonte. O que será discutido nesse tópico são as principais formas de

desenvolvimento desse código fonte, suas principais vantagens e desvantagens utilizando-se de um dos modelos.

É importante citar que, conforme estudado no capítulo anterior, nesse ponto passa a existir uma trajetória tecnológica no processo de desenvolvimento do *software*, e também no próprio *software*, que fará com que exista certo grau de diferenciação entre os diversos grupos de desenvolvimento, bem como entre os diversos modelos do mesmo. Conforme NELSON & WINTER, essa diferenciação se dará de acordo com a sua colocação nesse processo inovativo, diferenciando os diversos grupos entre os vencedores (ou aqueles que terão um alto grau de diferenciação e alta taxa de inovação) e os perdedores (aqueles que, por qualquer motivo, não irão gerar inovações tecnológicas importantes). Assim, os diversos grupos existentes, em sua maioria, devem possuir um determinado grau de inovação, no sentido de que alguns desses grupos estão desenvolvendo seus projetos já há alguns anos. O *kernel* estável do sistema operacional *Linux* está em desenvolvimento desde 1992, tendo incluído novas funções e novos modos de funcionamento durante os anos.

De fato, o sistema operacional *Linux* não está só presente no mercado de microcomputadores, sendo encontrado até em aparelhos celulares, sem citar outros casos igualmente importantes para as comunidades de *software* livres, que são encontrados nos mais diversos aparelhos eletrônicos.

JACKSON *apud* KING (1989) diz que "o projeto de *software* é uma atividade difícil. Para bem desempenhá-la é preciso muito senso prático, sólida formação técnica e muito bom humor para aqueles dias em que tudo dá errado".

Os modelos estudados aqui serão o de desenvolvimento de código aberto, ou *Open Source*, onde o código fonte que irá gerar o *software* é disponibilizado junto com o aplicativo, possibilitando que todos, desde usuários, programadores e até concorrentes, possam ler, modificar ou se utilizar dele e também o modelo de desenvolvimento conhecido como "proprietário", onde apenas a empresa ou grupo que o desenvolve tem acesso ao código fonte original, não podendo haver, portanto, qualquer modificação ou melhoria por quaisquer programadores que não sejam ligados diretamente à empresa desenvolvedora desse código.

É importante citar que estas não são as duas únicas formas de desenvolvimento de *software*, mas serão as únicas a serem estudadas por serem predominantes e também para buscar analisar como o desenvolvimento de *softwares* de código livre pode ser considerado uma inovação tecnológica em comparação ao modelo proprietário que é mais amplamente utilizado no mercado.

É importante frisar que no presente estudo o termo desenvolvimento será usado para designar o processo de criação do código fonte inicial[15], bem como suas alterações posteriores para sua melhoria/alteração visando atingir determinado objetivo, enquanto será utilizado o termo disseminação para a forma como o *software* é distribuído entre os usuários (tanto finais quanto programadores), seja para seu uso ou para melhorias no código fonte.

Em outras palavras, desenvolvimento é o processo no qual o código fonte é criado, mantido e melhorado, enquanto disseminação é a forma com que o código fonte atinge o mercado, seja na forma simples (o próprio código fonte, para verificação, estudo ou modificações comunitárias) seja na forma trabalhada (o *software* final).

2.2.1 – Os grupos de desenvolvimento de *software* de código fonte aberto

O objetivo desde item é estudar como surgiu o *software* de código aberto. Também serão demonstradas as formas com que se dá a distribuição de conhecimento desse modelo de desenvolvimento para compreender melhor a teoria apresentada no capítulo anterior. Discutir de que forma esse modelo de desenvolvimento pode ser considerado uma inovação tecnológica no mercado e como ele pode beneficiar usuários e empresas a baixar custo mantendo sua produtividade no que se refere ao uso de ferramentas de informática.

O desenvolvimento "anárquico" de um código fonte é dado através de grupos de criação de *software* aberto, onde todos os usuários têm acesso direto ao contido no

---

[15] Supondo que o código fonte original é desenvolvido através de linguagem estratificada, ocasionando um desenvolvimento em linhas de programação, independente do tempo de existência desse código, ou de já ter sido interpretado/compilado ou não

código, tanto para seu próprio conhecimento quanto para melhorias ou modificações. Esses grupos, muitas vezes formado por ex-funcionários de empresas que se utilizam do código proprietário, se formaram com a ideologia de criar um modelo de desenvolvimento independente que pudesse gerar algum tipo de inclusão social para os programadores e desenvolvedores que não trabalhassem em empresas que se "apropriam de seu próprio código" deixando ao mercado apenas o *software* final, ou seja, o programa de computador propriamente dito.

Em 1984, Richard Stallman criou o Projeto GNU[16], tendo o objetivo de desenvolver, em comunidade e com código aberto, um sistema operacional totalmente livre, onde qualquer pessoa teria o direito de utilizar o sistema sem pagar por compra ou direito de uso, e ainda tendo a liberdade de modificá-lo de acordo com suas necessidades. Em 1991, na Finlândia, Linus Benedict Torvalds criou o *Kernel*[17] do projeto GNU[18], e o apresentou às diversas comunidades de *software* para que fossem feitas modificações e melhorias. Surgiu daí o Linux (mistura de Linus e Unix, o que explica o porquê do acrônimo GNU conter o Unix), o primeiro e, até hoje, maior e mais conhecido *software* livre disponível (WIKIPEDIA, PROJETO GNU).

O projeto GNU é desenvolvido pela FSF (*Free Software Foundation*), entidade sem fins lucrativos, e tem por objetivo a eliminação completa de restrições ao uso e modificação dos códigos fonte de programas de computador.

"*Software* Livre (*Free Software*) é o software disponível com a permissão para qualquer um usá-lo, copiá-lo, e distribuí-lo, seja na sua forma original ou com modificações, seja gratuitamente ou com custo. Em especial, a possibilidade de modificações implica em que o código fonte esteja disponível." (PORTAL *SOFTWARE* LIVRE)

---

16 Stallman escolheu o GNU como mascote para o projeto, uma vez que o indivíduo, na manada, é fraco, enquanto o grupo, unido, se fortalece à medida que novos membros são incorporados. Além disso, GNU é um acrônimo para "GNU Não é *Unix*", já que o objetivo do grupo era que o *software* fosse compatível com o *Unix*, mas não utilizasse seu código fonte

17 Em tradução livre, cerne.

18 Richard Stallman estava trabalhando no *Hurd*, que seria o *Kernel* do projeto GNU, porém, o *kernel* desenvolvido por Torvalds acabou sendo incorporado ao GNU/*LINUX*, como o projeto é conhecido atualmente, embora essa nomenclatura ainda seja motivo de discussão entre seus usuários.

2.2.1.1 – Forma de organização para o desenvolvimento do *software* livre

Para melhor compreender como o código fonte é desenvolvido, se faz necessária o estudo da forma com que os grupos de desenvolvimento se reúnem para desenvolver determinado *software.*

Conforme BERKUS, as comunidades que desenvolvem *software* de código aberto (também chamados de *Open Source*) são divididas em grupos de acordo com o tamanho e a finalidade do grupo desenvolvedor. As comunidades são classificadas da seguinte forma:

Solo: Os projetos possuem apenas um desenvolvedor, e em alguns casos dois. São os responsáveis por alterações do código fonte original, e o grupo é criado com o objetivo de dar algum destaque ao profissional que irá manter o grupo. Em alguns casos, são programadores de grupos maiores que abandonaram o projeto em que trabalhavam por alguma divergência com grupo anterior.

Os projetos "solo" possuem geralmente mantenedores envolvidos que dão boas respostas à comunidade. Porém, caso haja qualquer tipo de problema, por haver uma única pessoa responsável por manter o projeto, o mesmo pode sofrer grandes atrasos e até pode ser cancelado por falta de interesse e/ou o profissional responsável não disponha de tempo para manter o projeto em funcionamento.

Para participar do grupo, basta que o usuário programador modifique o código fonte e encaminhe essas modificações ao mantenedor, que irá testar e decidir se as incluirá ou não no código fonte original. Não há a necessidade do usuário programador se cadastrar ou manter qualquer vínculo com o mantenedor do projeto.

Como exemplo, podem ser citados os aplicativos Joe (editor de texto para sistema operacional *Linux*) e o *iBookShelf* (Um catálogo pessoal de livros).

Monarquia: São definidos como os projetos "solo" que formam uma grande comunidade e passam a exigir que mais pessoas tenham acesso à inclusão de melhorias ou modificações no código fonte. O líder do projeto ainda é o único responsável por definir quais modificações serão incluídas no código (por isso, monarquia, ou seja, há um único responsável pelo projeto todo, o "rei", que irá tomar as decisões que mais

afetem o desenvolvimento do *software*, e "ministros", ou seja, outras pessoas, com poderes delegados pelo mantenedor, que terão o objetivo de auxiliar o desenvolvimento do aplicativo, mas que não terão o poder total para tomar as decisões). Em caso de sobrecarga a esse profissional, há a indicação de um substituto imediato que tomará essas decisões para que o projeto não fique por muito tempo parado.

A diferença entre o projeto monarquia e o projeto solo é, além do tamanho, a inclusão de modificações no código fonte são testadas e discutidas em grupo, e não somente pelo mantenedor, como é o caso dos projetos solo, o que implica em aplicativos mais bem acabados.

Para participar do projeto, basta se aproximar dos programadores líderes ou algum dos seus colaboradores e enviar as modificações do código fonte, que serão analisadas, e a decisão dessas modificações serem incluídas ou não será tomada rapidamente. Dessa forma, os projetos que se utilizam dessa hierarquia para obter resultado possuem um acesso simplificado aos mantenedores do código fonte original e a evolução dos softwares se dá de forma mais eficiente, rápida e efetiva do que as outras hierarquias existentes

Com o crescimento do grupo, é muito comum ocorrer divergência entre os membros e o mantenedor, causando assim algum grau de ruptura, o que causa a saída de algum colaborador que, com alguma freqüência, irá criar um projeto solo.

Como exemplo pode ser citado o que é considerado o maior projeto de *software* livre atualmente, o sistema operacional *Linux*.

Comunidade: São geralmente grupos que se assemelham aos projetos monarquia em tamanho, porém as decisões a respeito de que alterações serão feitas no código fonte original são tomadas em grupos através de votação e não por um único indivíduo. Quando há uma nova contribuição ao código fonte original, toda a comunidade analisará, bem como testará para certificar a qualidade ou necessidade de tal contribuição, e a inclusão será decidida através de votação.

Esse método de hierarquia significa que as modificações e melhorias ao código fonte original levarão mais tempo para serem analisadas que as modificações

feitas aos códigos fonte que se utilizem de outros modos de organização, mas que, em contrapartida, terão uma análise mais efetiva de suas modificações, significando um *software* mais bem acabado e com uma quantidade ainda menor de falhas, sejam elas críticas (falhas de segurança) ou simples (falhas que comprometem o funcionamento do aplicativo).

Comumente a votação é feita por meritocracia, ou seja, as decisões são tomadas por merecimento. Muitas vezes o mérito é definido pelo tempo do programador no grupo, pela importância das contribuições já feitas e pela eloqüência do mesmo no grupo. É um processo altamente politizado, em que novos membros são mais duramente avaliados e suas contribuições são testadas vigorosamente.

Normalmente há um cadastro central de usuários para facilitar o controle, embora não haja qualquer processo de ingresso no grupo, bastando geralmente o preenchimento de alguns dados para facilitar o acesso do novo membro ao grupo, bem como sua participação e comunicação no grupo de discussão da comunidade.

O suporte aos usuários dos *softwares* criados por esse tipo de grupo é geralmente feito através das listas de discussão mantidas pelo próprio grupo, em que participam todos os colaboradores, independente da importância e do tempo dentro da equipe. Por ser um grupo com uma estrutura central, porém sem um mantenedor definido, a conseqüência é de que geralmente não há uma estratégia detalhada.

Pela dificuldade política em se manter os projetos de comunidade, existem poucos exemplos conhecidos para citar, porém os sistemas operacionais *Debian* e o *FreeBSD* são os exemplos mais divulgados.

Corporativo: São projetos em que o código fonte original não tenha sido criado por uma comunidade livre e sim por uma empresa que se utilizou do modelo do *software* proprietário. A empresa desenvolvedora do código fonte original liberou o acesso do mesmo para receber contribuições públicas. Geralmente o grupo que desenvolverá o código é formado por empregados da empresa, que continuarão a trabalhar nela e irão desenvolver o código em paralelo.

A estratégia para o desenvolvimento do código fonte será tomado pela empresa dona do código original, geralmente pelo seu próprio departamento de

marketing. As decisões a respeito de abrir o código são tomadas porque o código fonte é antigo e, portanto, não comercializável, não gerando para a empresa um custo baixo na "reciclagem" do código. A decisão pode ser também para facilitar a distribuição do *software* de pequenas empresas ao mercado.

Para contribuir com modificações ou melhorias ao código fonte, geralmente é necessário um processo formal de avaliação, e os novos colaboradores não participam do processo de decisão estratégico para o desenvolvimento do código fonte, o que explica porque esses grupos geralmente possuem poucos colaboradores diretos.

O suporte aos usuários desses *softwares* é dado através de grupos de discussão e fóruns públicos na *internet*, geralmente por usuários mais experientes do projeto e colaboradores que a empresa permite que usem mais tempo para participar. Para obter suporte da empresa é necessário assinar um contrato formal de serviço ou obter uma licença comercial.

O *OpenOffice* é o melhor exemplo atualmente dos projetos corporativos, com seus principais mantenedores sendo funcionários da empresa *Sun Microsystems*, que é, legalmente, a proprietária do código fonte original do *StarOffice*, que é o *software* que deu origem ao *OpenOffice*. O sistema operacional *Red Hat Fedora*, da empresa americana *Red Hat* pode ser incluído nesse modelo.

Fundação: São grupos não comerciais com uma estrutura administrativa próxima a de uma empresa comercial, com gerentes, diretores e uma cadeia de comando definida e funcional. É um modelo criado geralmente para o desenvolvimento do código fonte de um *software* utilizado por grandes corporações, que se utilizará de mecanismos legais para proteger seus interesses e assegurar algum tipo de poder de decisão no desenvolvimento estratégico do *software*.

As fundações se originam quando há a necessidade de uma estrutura legal e formal com a necessidade de contratar funcionários para o desenvolvimento de determinado *software*. Pode ser também o caso de empresas que abrem o código fonte de um *software* "proprietário" com o fim de protegê-lo.

O ingresso a essas equipes de desenvolvimento se dá através de uma avaliação do candidato, incluindo aí a assinatura de algum documento para que a fundação tenha

uma garantia de direitos.

Os novos colaboradores não possuem qualquer direito no comando do grupo, sendo apenas usuários programadores com permissão para participar do grupo. A fundação mantenedora do projeto irá definir então em que parte do projeto esses novos usuários programadores serão incluídos. Há inclusive a possibilidade da fundação decidir pela criação de subgrupos para o desenvolvimento de uma parte específica do *software*, que poderá ser administrado como projetos monárquicos ou de comunidade individualmente.

O suporte dado aos usuários de *softwares* criados por fundações se dá através de grupos de discussão e fóruns públicos na *internet*. O suporte pago pode ser encontrado em algumas fundações que se comportem, nesse aspecto, como grupos corporativos.

O *Apache* (sistema operacional para servidores), o *Gnome* (ambiente de *Desktop* para sistemas *Unix*), o *Mozilla Firefox* (navegador) e o *Mozilla Thunderbird* (gerenciador de *e-mails*) são os aplicativos mais conhecidos que se utilizam de fundações para o seu desenvolvimento.

Os diversos grupos formados para a criação de *softwares* de código aberto são classificados de uma das formas estudadas acima podendo também ser classificados em mais de um grupo de acordo com sua estrutura mista. A estrutura desses grupos não é fixa, podendo sofrer modificação a qualquer momento de acordo com suas posições estratégicas e como crescimento ou decrescimento do tamanho do grupo.

É importante notar que o modelo de comercialização ou distribuição do *software* de código aberto não implica necessariamente na distribuição gratuita do produto, e sim na distribuição gratuita do código fonte. Desde que sejam respeitadas as regras da *Gnu Public Licence* (GPL), ou do *Open Source Iniciative* (OSI, (apresentadas no tópico a seguir), o grupo de desenvolvimento ou o mantenedor do projeto tem total liberdade para cobrar o valor que julgar justo pelo uso do *software*, apropriando novas modificações no código fonte original, incluindo aquelas criadas por programadores independentes e/ou sem qualquer vínculo empregatício com a empresa ou grupo criador do projeto, desde que essas modificações tenham sido

enviadas para que fossem integradas ao código fonte, respeitando assim qualquer lei de direito autoral vigente.

As definições do *software* livre se aplicam também aos *softwares* de código aberto, incluindo aí a obediência aos quatro tipos de liberdade definidas na *Gnu Public Licence* e do Open *Source Iniciative*, porém os desenvolvedores que se utilizam do modelo de desenvolvimento do *software* de código aberto não condenam o modelo de desenvolvimento de *software* proprietário. Tecnicamente todo *software* de código aberto é também um *software* de código livre.

Assim, antes que seja feita uma explicação sobre as formas de licenciamento dos aplicativos desenvolvidos através de código aberto, se faz necessária uma melhor ligação entre o assunto discutido nesse tópico com o modelo apresentado no capítulo anterior, que trata das teorias desenvolvidas por ROSENBERG. Nesse sentido é possível verificar esse progresso técnico, embora seja ligado ao desenvolvimento de um produto (ou ao processo de produção), pode-se fazer uma ligação entre o progresso de um determinado *software* com o progresso do grupo que desenvolveu determinado aplicativo denotando a importância da base teórica evolucionista.

Assim, conforme foi discutido, as comunidades de desenvolvimento surgem de um processo totalmente solitário (muitas vezes levado por apenas um único indivíduo) e, com o sucesso dessa "comunidade" [19], há uma evolução natural desse grupo para que chegue até um grupo de grande alcance e participação de centenas, milhares ou até milhões de pessoas mudando dessa forma o paradigma tecnológico como proposto por DOSI.

---

[19] Entre aspas com o sentido de definir comunidade como o grupo que participa não só da produção, mas também do uso e do seu desenvolvimento através desse uso. O fato de um único programador criar, manter e divulgar determinado projeto não significa que não haja uma comunidade dando suporte ao seu trabalho, mesmo que não dando a esse desenvolvedor o seu devido crédito

2.2.1.2 - Forma de licenciamento dos *softwares* de código aberto

Os diversos grupos de desenvolvimento dos aplicativos de código aberto, após criar e definir qual vai ser o nicho de mercado ocupado por seu projeto, e após o completo desenvolvimento desse aplicativo, irá disponibilizar que esse aplicativo, seja comercialmente, ou para obtenção gratuita através de *download* público, irão enquadrar esse projeto para o uso necessário do usuário final, que irá obedecer certas, e necessárias regras, respeitando o devido direito autoral desse projeto e que, de qualquer modo, seja dado crédito para os criadores, mantenedores e colaboradores desse projeto.

Neste tópico, portanto, serão analisados os requisitos para que determinado código fonte considerado como *open source* seja comercializado e utilizado por aqueles que dele quiserem fazer uso e também para um entendimento mais completo desse modelo de desenvolvimento.

O *software* de código aberto está de acordo com as regras do Open *Source Iniciative*, ou Iniciativa do Código Aberto, que é composto de dez critérios:

1) **Redistribuição Gratuita**: A licença não pode restringir qualquer parte do *software* e não pode haver qualquer exigência do criador original do projeto que se refira a pagamento de *royalties* ou qualquer cobrança de taxa sobre o uso do código fonte;

2) **Código Fonte**: O *software* final deve conter em sua documentação o código fonte original do projeto e o mesmo deve ser de acesso irrestrito a qualquer um que tenha a intenção de usar ou modificar o *software*;

3) **Trabalhos derivados**: A licença de distribuição do *software* deve permitir modificações ou trabalhos derivados do código fonte original, e também permitir que estes trabalhos derivados sejam também distribuídos pelos mesmos termos da licença do código fonte original;

4) **Integridade ao código fonte original**: A licença pode restringir o uso do código fonte original de ser redistribuído após modificação apenas se a licença original permitir a distribuição de complementos do *software* original com o propósito de

modificar o código original;

5) **Não discriminação de grupos ou pessoas**: A licença de distribuição do *software* original não pode discriminar, de qualquer forma, determinados grupos ou pessoas de se utilizar ou modificar o código fonte original;

6) **Não discriminação contra outras atividades**: Não pode haver qualquer discriminação quanto aos usos possíveis de determinado *software*, seja para o mercado corporativo ou para pesquisa genética ou atmosférica, por exemplo;

7) **Distribuição da licença**: Os direitos de uso, modificação e distribuição do código fonte original devem ser aplicadas a quaisquer modificação e redistribuição do código fonte modificado, sem a necessidade de uma licença adicional para esses novos projetos;

8) **A licença não deve ser específica a um produto**: Caso um determinado grupo faça uso de apenas uma parte do código fonte original, o novo projeto também estará de acordo com a licença original do produto que originou o novo projeto;

9) **A licença não pode restringir o uso por outros *softwares***: A licença não pode impor restrição a quaisquer *softwares* distribuídos juntamente com o *software* original;

10) **A licença não pode ser neutra quanto à tecnologia**: O *software* e o código fonte original devem ser distribuídos sobre qualquer forma de tecnologia ou meio de distribuição, seja por *download* direto, meios magnéticos (cd-rom, etc) ou quaisquer outras formas de distribuição existentes.

Curiosamente, esse modelo de desenvolvimento tem sido usado além do âmbito da informática. Em 2006, um grupo de estudantes da universidade de Copenhagem, Dinamarca[20], utilizou o conceito de *software* livre e desenvolveu a primeira receita de 'cerveja livre', utilizando os mesmos moldes do desenvolvimento de um código fonte criado para o mercado de informática (receita) para a obtenção de um determinado tipo de *software* (cerveja) desenvolvido em comunidade. Qualquer grupo que faça uso dessa receita (ou desse código fonte) pode produzir sua própria cerveja e efetuar qualquer modificação na receita original, e não há qualquer obrigação

---

[20] Copenhagen IT University, como visto em http://www.freebeer.org/

de pagamento de *royalties*, tendo como única obrigação a divulgação da receita original, com o devido crédito ao grupo que criou esse projeto.

Um dos motivos principais na diferenciação entre os projetos de *software* livre e *software* de código aberto é que este último não discrimina modificações criadas para empresas que se utilizem do modelo de desenvolvimento do código proprietário para a distribuição ou venda de um determinado tipo de *software*, ou até mesmo a produção e/ou comercialização de qualquer outro produto, seja no ramo dos microcomputadores ou não.

Outro motivo que diferencia os modos de licenciamento do *software* é que a *Gnu Public Licence* geralmente é classificada mais como uma licença política e social, e não apenas como uma licença de uso de *software*, sendo considerada também como um movimento social com o objetivo de ir contra esferas de controle de mercado e aprisionamento do processo de conhecimento presente no modelo de *software* proprietário.

As licenças para o desenvolvimento utilizando o modelo do código livre não são consideradas válidas somente para o mercado de *software*, mas podendo ser estendidas para qualquer tipo de aplicação e todos os tipos de tecnologia que dela possam fazer uso.

2.2.1.3 – A diferença entre *software* de código aberto e *software* livre

Para facilitar o entendimento do processo de desenvolvimento e buscar explicar as vantagens e desvantagens em se produzir *softwares* se utilizando do modelo de código aberto, é necessário o estudo da divergência entre o *software* livre e o *open source* para definir qual é o modelo utilizado por ambas as formas de desenvolvimento e buscar semelhanças que possam prover uma análise mais ampla.

É importante frisar que as semelhanças entre os modelos permitem analisar de forma similar o comportamento dos diversos grupos de desenvolvimento, havendo apenas a necessidade de pequenas modificações no estudo da distribuição e disseminação tanto do *software* gerado quanto do código fonte.

Para que um determinado *software* seja classificado como um projeto livre ou de código aberto, é necessário estar de acordo com os quatro tipos de liberdade descritos na *GNU PUBLIC LICENCE*, que são a liberdade de permitir que outros grupos ou pessoas possam; (a) executar o *software* para atingir qualquer objetivo, (b) estudar como ele funciona e modificá-lo de acordo com suas necessidades ou vontades, (c) distribuir o *software* gratuitamente ou cobrando por ele, para promover uma difusão maior e também auxiliar os usuários que porventura possam vir a utilizá-lo, e (d) aperfeiçoar e modificá-lo com o objetivo de evoluí-lo, desde que o novo código fonte seja distribuído à comunidade para que todos possam se beneficiar de tais mudanças e tenham a possibilidade de aplicar novas modificações quando acharem necessárias.

O *software* livre vai além da questão financeira ou técnica, sendo definido mais como uma ideologia ou motivação social, calcada na liberdade em se utilizar dos aplicativos gratuitamente e na colaboração para melhor desenvolvê-lo. (NUNES, 2003).

A liberdade de executar o *software* indica a liberdade de utilização do mesmo, seja para o usuário doméstico ou corporativo, sem qualquer limitação de uso ou quantidade de cópias obtidas. A redistribuição do código fonte é obrigatória e sem ela o processo de disseminação do conhecimento não ocorre, uma vez que esse processo passa necessariamente pela disponibilização pública do código fonte, a ser estudada adiante.

É importante demonstrar a diferença entre os modelos, para que não haja uma confusão ao estudar os modos pelos quais esse conhecimento é gerado e distribuído, e embora o conhecimento seja gerado pela comunidade de desenvolvimento, os sistemas gerados livremente não podem ser utilizados por empresas privadas sem que estas abram o código resultante novamente para a comunidade. Isso difere para os códigos gerados abertamente, pois as empresas privadas que façam uso do código tem o direito de apropriar modificações, desde que respeitem ao direito de propriedade e dêem o devido crédito.

É necessário também citar a diferença de uso da palavra livre e aberta para não

gerar controvérsias com o entendimento do presente estudo. Projetos gerados livremente fazem uso da *Gnu Public Licence* sendo considerados portanto como *software* livre, e os gerados abertamente fazem uso da *Open Source Iniciative*, considerados como sendo *software* de código aberto.

Para finalizar, os *softwares* de código aberto são geralmente classificados também como *softwares* livres, porém sua forma de licenciamento é regida pelo Open *Source Iniciative* e não pela *Gnu Public Licence*.

Para facilitar a análise e o compreendimento do fenômeno estudado, serão utilizados principalmente modelos de desenvolvimento de *software* de código aberto, pois os mesmos também são considerados como *software* livre, podendo servir de exemplo ao mesmo tempo à ambas definições. E como o objetivo principal do presente estudo é analisar a forma de desenvolvimento e não suas formas de licenciamento, sob a ótica de como é criado o código fonte, é possível considerar os fenômenos como sendo vertentes diferentes de um mesmo estudo de caso.

Para uma melhor compreensão do fenômeno, embora as duas formas 'livres' de desenvolver e manter o *software* tenham suas particularidades e, portanto, sejam diferentes, mesmo que minimamente, no presente estudo será considerada a diferença entre apenas dois sistemas, o sistema chamado "proprietário", ou seja, com o código fonte 'preso' à empresa que o criou, e o sistema chamado "livre", com o código distribuído livremente para modificação e melhorias em comunidade.

A diferenciação em apenas dois modelos (proprietário e "livre"), considerando tanto o *software* de código aberto quanto o *software* de código livre como um modelo só (exemplificado tanto como o *software OpenOffice*, com seu código livre e aberto, mas legalmente pertencente à *SUN MICROSYSTEM*, quanto pelo sistema operacional *Linux*, com seu código independente de corporações), e o modelo proprietário (como a família de sistemas operacionais *Windows*, da *Microsoft*) tem o objetivo de simplificar a análise e estudar de forma mais concisa o movimento anárquico e buscar uma compreensão mais justa e direta do movimento de libertação do código fonte como forma de inovação tecnológica, independente das pequenas particularidades que esses dois processos (tanto o modelo "livre", quanto o modelo "aberto") possuem,

considerando as suas semelhanças e principalmente suas diferenças compartilhadas em relação ao sistema mais amplamente utilizado, o "sistema proprietário".

2.2.1.4 – Disseminação do conhecimento gerado em comunidades

O objetivo deste tópico é analisar de que forma é gerado e transmitido o conhecimento adquirido entre as diversas comunidades de criação de *software* e quais as vantagens e desvantagens desse modelo.

A teoria apresentada por NELSON & WINTER no capítulo anterior se torna evidente no progresso gerado pelos grupos de desenvolvimento de aplicativos de código aberto, uma vez que existe um conhecimento gerado através dessa inovação, uma manutenção e ampliação de conhecimento gerado através do desenvolvimento dessa inovação e, como citado por ROSENBERG, há um grande e evidente 'aprendizado pelo uso' daqueles que, por suas necessidades particulares, fazem uso de aplicativos desenvolvidos dessa forma.

Um dos melhores exemplos do aprendizado pelo uso é que os grupos desenvolvedores de *software* de código aberto se reúnem em eventos conhecidos como FISL (ou Fórum Internacional do *Software* Livre), que ocorrem em diversas cidades do mundo, e têm como objetivo oferecer a esses grupos um local onde possam discutir sobre o movimento do *software* livre, bem como gerar discussões sobre das vantagens, desvantagens, políticas e rumos dessas comunidades.

Durante o evento FISL 8.0 (Oitavo Fórum Internacional do Software Livre), ocorrido entre os dias 12 à 14 de Abril de 2007 em Porto Alegre, o economista Paul Singer, Secretário Nacional de Economia Solidária, vinculado ao Ministério do Trabalho e Emprego do Brasil, disse que "a idéia do *Software* Livre foi a melhor invenção para controlar o movimento de privatização do conhecimento".

Essa citação demonstra que por trás do simples desenvolvimento de um *software* há toda uma ideologia social a respeito de como o conhecimento é gerado e perpetuado na sociedade, iniciando com a equipe que desenvolveu determinado projeto ou até todos aqueles que possam, de alguma forma, se beneficiar desse conhecimento.

Todo o conhecimento obtido através de comunidades de *software* de código aberto é adquirido com o desenvolvimento do aplicativo e oferecido para todos aqueles que tenham interesse em se utilizar de tal conhecimento tenham uma noção básica de programação de computadores e que tenham o código fonte original.

Para a redistribuição do código fonte não é necessária a permissão ou autorização do proprietário do *software* uma vez que no projeto inicial, o proprietário autorizou o uso e alteração de acordo com a *Gnu Public Licence*. É importante frisar que, mesmo não havendo necessidade da autorização do proprietário, no novo código fonte se faz necessário o reconhecimento da origem do *software* bem como os que participaram do desenvolvimento até aquele ponto.

O *software livre* é desenvolvido através de comunidades constituídas na maioria das vezes, e embora seja um desenvolvimento público onde todos tenham acesso ao código fonte que o originou, isso não significa que o mesmo não esteja protegido por leis de direitos autorais, ou *copyright*. Para o caso do *software livre*, há o conceito de *copyleft*[21], que se baseia em uma propagação de direitos, ou seja, ao se utilizar de um código fonte que esteja de acordo com a *GPL*, o programador é obrigado a transferir o novo código fonte de acordo com as regras já estabelecidas, e não apropriá-lo para seu próprio benefício. Em outras palavras, o programador é obrigado a estender o direito de acesso ao código fonte para todos os que dele quiserem fazer uso. (WIKIPEDIA, *Software* Livre)

"*Software* Livre não é de domínio público, apesar de essa ser uma interpretação compreensível do conceito de "liberdade"". (NUNES). Programas de domínio público são aqueles em que o criador não possui mais os direitos autorais do código fonte e da forma de licenciamento do código.

---

[21] Popularizado por Richard Stallman ao ser incluído em 1988 à *GPL*. Tem o objetivo de prevenir que sejam colocadas restrições sobre os direitos autorais originais, mantendo sua liberdade. O símbolo utilizado é o © invertido.(http://pt.wikipedia.org/wiki/Copyleft)

NUNES (2007) define a eficiência do *software* livre da seguinte forma:

> O software livre tem, comprovadamente, menos *bugs*[22] do que o software proprietário, e esses *bugs*, quando identificados, levam um tempo muito menor para serem corrigidos. Isso se deve ao fato de o código fonte da aplicação estar disponível para muito mais programadores ao redor do mundo. É como se a qualidade do programa estivesse sendo constantemente "auditada" por uma enorme equipe de programadores.

Além disso, "A grande força do software livre está no potencial de cooperação para depuração coletiva, capaz de neutralizar pressões mercadológicas e políticas e melhor dominar complexidades.". (WIKIPEDIA, Código Aberto)

Para citar um exemplo, conforme notícia do jornal inglês *The Inquirer* com data de 30 de abril de 2007[23], o médico Francês Michel Xhaard desenvolveu *drivers* de dispositivo para 235 modelos de *webcams* para o sistema operacional *Linux* sem que tivesse recebido qualquer tipo de remuneração ou incentivo que não fosse o seu próprio bem estar. Essas contribuições foram posteriormente incorporadas ao sistema operacional *Linux* e solucionaram problemas relacionados ao desenvolvimento desses *drivers*, incluindo aí o custo que as empresas que se utilizam do sistema operacional iriam despender para que elas próprias desenvolvessem um sistema de reconhecimento desse *hardware* específico.

O exemplo acima demonstra que uma modificação no código fonte original do projeto, ou parte dele, bem como em seus *drivers* de dispositivos, podem ser criados, modificados e incorporados no sistema para o qual foram desenvolvidos, sem custo para os grupos ou empresas que fazem uso de *software* de código aberto, causando uma evolução natural e constante de seu modo interno de funcionamento.

Conforme pesquisa da Universidade de Maastricht, Holanda[24], o Brasil é hoje o país que possui o maior número de programadores trabalhando em projetos *Open Source*, tendo concluído que trabalhar com *software* livre oferece a oportunidade de maiores salários, sendo mais valorizada na Argentina, África do Sul e Brasil.

---

22 Falhas de programação, causando erro em determinada função do *software*.

23 http://www.theinquirer.net/en/inquirer/news/2007/04/30/one-man-writes-linux-drivers-for-235-usb-webcams

24 http://wnews.uol.com.br/site/noticias/materia_especial.php?id_secao=17&id_conteudo=404&id_coluna=9

2.2.2 – O modelo de desenvolvimento do *software* de código fonte proprietário

Neste tópico será analisado de que forma é desenvolvido o *software* proprietário, utilizando como exemplo a *Microsoft*, por ser atualmente a maior empresa do gênero no mercado mundial.

O termo *software* proprietário foi cunhado a partir do aumento da popularidade do *software* livre, uma vez que até então este era o modelo predominante no mercado e o termo indica que o *software* é de propriedade do desenvolvedor original[25], porém o código fonte, por ser "proprietário" pertence somente à empresa criadora do projeto ou à que adquirir os direitos sobre o código fonte original.

O modelo de desenvolvimento do *software* proprietário se difere do modelo do *software* de código aberto por não haver qualquer abertura do seu código fonte. De fato, as empresas que se utilizam do código aberto se assemelham a uma produção de um bem industrial tradicional, no sentido de que qualquer tipo de segredo ou vantagem tecnológica ali existente não é conhecida do público em geral e de seus concorrentes.

Qualquer vantagem obtida através de um processo de inovação tecnológica seja barateamento de custo por um processo de produção ou pela criação de um novo produto fica restrita aos departamentos internos da empresa, e não são difundidas ao mercado.

Para que seja possível uma alteração no código fonte original é necessário que o mesmo seja aberto pela empresa ou passe a domínio público. No Brasil a lei que trata do direito autoral para o mercado de *software* é a lei nº 9.609/98 (também chamada de "Lei dos *Softwares*". Conforme o artigo nº 45 da lei nº 9.610/98, só será considerado domínio as obras às quais decorreu o prazo de proteção aos direitos patrimoniais[26], obras de autores falecidos que não tenham deixado sucessores ou ainda obras de

---

[25] Tal qual o ***software*** de código aberto, em que o grupo desenvolvedor também é dono do código original, porém o mesmo é disponibilizado à sociedade.

[26] O prazo de proteção do ***software*** é de cinqüenta anos, contados a partir de 1º de janeiro do ano subseqüente ao da sua publicação ou, na ausência desta, da sua criação (**artigo 2º, § 2º, da Lei nº 9.609/98**).

autores desconhecidos[27]. A lei dos direitos autorais, de nº 9.610/98 também pode ser utilizada, mas de forma mais ampla.

Em *softwares* obtidos de países estrangeiros, via *download*, é utilizado o artigo 2º da lei nº 9.610/98 que diz: "Aplica-se o disposto nesta lei aos nacionais ou pessoas domiciliadas em país que assume aos brasileiros ou pessoas domiciliadas no Brasil a reciprocidade na proteção aos direitos autorais ou equivalentes".

A forma de licenciamento do *software* proprietário é usualmente conhecida como EULA (*End User Licence Agreement*[28]), que transfere ao usuário o poder apenas para executar o aplicativo, mas não para qualquer modificação ou melhoria, tanto para o código fonte quanto para o *software* em si.

Após a criação e desenvolvimento do código fonte original, o mesmo é comercializado ou distribuído na forma de um arquivo executável, mas sem que os usuários do programa tenham qualquer acesso ao código fonte original. O *software* é considerado como pronto e a partir desse ponto são desenvolvidos apenas *softwares* de correção (Chamados de *Patchs*[29]), que tem a função de incluir alguma alteração no *software* original, mas passa a ser um programa dependente, ou seja, sem a existência do aplicativo para o qual foi desenvolvido, não tem qualquer função prática. De maneira vulgar, é apenas um remendo no código fonte original, que têm sua existência garantida por ineficiência, falha ou imprecisão do código fonte original. O desenvolvimento contínuo portanto é praticamente inexistente e os programadores que participaram do projeto inicial são deslocados para outros projetos ou, mais comumente, em desenvolver uma nova versão do *software*.

2.2.2.1 – Disseminação do conhecimento gerado pelo *software* proprietário

Neste tópico serão discutidas as maneiras com que o *software* proprietário gera e administra o conhecimento, e quais são as vantagens e desvantagens desse modelo.

O código fonte de um *software* proprietário pertence unicamente à empresa

---

27 Conforme consta no site http://www.ufsm.br/direito/artigos/informatica/software.htm

28 Licença de uso para o usuário final, em tradução livre.

29 Correção às falhas da programação do código fonte original.

que o criou, sendo objeto de estudo apenas àqueles que possuam um vínculo empregatício com a empresa e a permissão de poder ler ou modificar o mesmo. Todo o conhecimento adquirido dessa forma é obtido, mantido e repassado apenas para as novas gerações de funcionários da empresa e, portanto, o número de pessoas que poderão contribuir diretamente é naturalmente limitado.

Usuários finais dos aplicativos têm a liberdade de utilizar o *software*, mas não de incluir por conta própria qualquer função ou modificação ao código fonte original. Para isso é necessária a comunicação com a empresa desenvolvedora apenas com o objetivo de fazer uma simples sugestão que, quando aprovada, será estudada e incluída no código fonte por funcionários da empresa e quem fez a sugestão não será remunerado de qualquer forma, sendo em valores monetários ou no reconhecimento por parte da empresa da participação do usuário.

A estrutura administrativa é rígida e bem definida, e geralmente os programadores não possuem acesso a todo o código fonte, mas apenas a uma pequena parcela que representa a função do sistema em que estão desenvolvendo. Dessa forma, caso um desses funcionários possa de qualquer forma "vazar" essa informação, ela será imperfeita e incompleta. Dessa forma é mantida a segurança à segredos na criação do código fonte proprietário.

Não há um evento onde os grupos e empresas que utilizam o código proprietário discutam sobre seus códigos fonte e obtenham quaisquer contribuições de outros grupos com o objetivo de aprimorar o *software* original, bem como alterar qualquer falha durante a execução de um projeto. Existem feiras internacionais de exposição de produtos, onde são expostos somente os produtos finalizados, e onde eventualmente são feitos lançamentos de novas tecnologias, mas que não oferecem ambientes de discussão para as empresas que fazem uso desse modelo de desenvolvimento.

Vale frisar que desenvolvedores de código proprietário, com destaque à *Microsoft*, enviam funcionários qualificados para sua representação nos Fóruns Internacionais de *Software* Livre com o objetivo de gerar discussão a respeito das dicotomias entre ambos os modelos. A própria *Microsoft* tem utilizado soluções de

softwares de código aberto para a melhoria de seus próprios produtos[30], e além de fazer uso da tecnologia criada em comunidades, irá contribuir com o código fonte original, colaborando assim com o desenvolvimento desse *software*. Conforme Kyril Faenov, diretor de computação de alta performance da *Microsoft* em 2005, o desenvolvimento de uma solução própria, ao invés de utilizar uma já criada através dessa comunidade de *software* livre levaria muito tempo e custaria muito caro.

Pode-se concluir que há, portanto, um "engessamento" do processo de conhecimento do *software* proprietário no que se refere aos benefícios "sociais" do desenvolvimento do código fonte, havendo somente o ganho financeiro com o desenvolvimento de novas versões do aplicativo com novas funções, e exigindo um *hardware* mais eficiente, na maioria das vezes, mais caro.

Embora seja lógico que, com o advento de computadores mais potentes e poderosos, a indústria do *software* irá criar programas que se utilizem melhor dessa capacidade de processamento, é possível afirmar que nem sempre existirá o uso eficiente de um *hardware*, por assim dizer, mais modesto. Existem certas versões do sistema operacional *Linux*, com destaque para o *Linux Slackware* que, mesmo com o desenvolvimento de microprocessadores mais poderosos, têm todas as suas funções executadas sem perda de eficiência, em microprocessadores mais antigos.

Conforme consta no *site* do desenvolvedor[31] do *Linux Slackware*, o *hardware* mínimo para instalar e executar está muito abaixo daquele necessário para executar o *Microsoft Windows Vista*, por exemplo, o que mostra um aproveitamento significativo dos recursos presentes em um microcomputador.

Com um número reduzido de programadores, o desenvolvimento de correções às falhas do *software* original é mais lento e, sem dúvida, mais limitado. Durante o ano de 2006 o *software Internet Explorer*, desenvolvido pela empresa americana *Microsoft* apresentou falhas de segurança durante 284 dos 365 dias do ano[32] enquanto o concorrente de código aberto *Firefox*, desenvolvido pela Fundação *Mozzila*,

---

30 http://idgnow.uol.com.br/computacao_corporativa/2005/09/19/idgnoticia.2006-03-12.6599387025/

31 http://www.slackware.com/install/sysreq.php

32 http://blog.washingtonpost.com/securityfix/2007/01/internet_explorer_unsafe_for_2.html

apresentou falha durante apenas 9 dos 365 dias. Isso significa que as falhas localizadas em *software* de código aberto possui um tempo de trabalho maior e uma capacidade melhor pra identificar, corrigir e testar novas versões dos aplicativos, sem as falhas localizadas anteriormente.

Objetivou-se neste capítulo estudar as formas de desenvolvimento predominantes no mercado, bem como as formas utilizadas para o licenciamento direto de seus produtos para o usuário final, tendo como idéia geral a fundamentação teórica baseada em NELSON & WINTER, DOSI e PAVITT, para buscar uma análise a respeito dos modelos de criação de *software* e compreender de que forma o modelo de criação do *software* livre pode ser considerado como uma trajetória independente do paradigma tecnológico atual, o *software* proprietário.

CONCLUSÃO

No decorrer do presente estudo foi efetuada uma análise do fenômeno *software*, e mais especificamente sobre o modelo de desenvolvimento de código livre, utilizando teorias de diversos autores evolucionistas, tendo como destaque NELSON & WINTER, DOSI, ROSENBERG e PAVITT, para lançar luz numa matéria econômica que tem recebido pouca atenção dos teóricos econômicos atualmente.

A preocupação maior na análise do fenômeno estudado foi de que forma a geração de conhecimento, e sua posterior difusão do conhecimento como um processo social, mesmo que tenha o lucro puro e simples como objetivo, e como esses processos de criação, influenciam no surgimento de paradigmas tecnológicos suas trajetórias independentes que venham a desafiar o modelo atualmente em uso.

Tendo sido estudadas as formas do atual paradigma tecnológico, o modelo de desenvolvimento proprietário, e após uma análise mais profunda no modo como esse modelo foi modificado com o passar dos anos para uma trajetória independente que veio, ao seu tempo e ao seu modo, mudar quase drasticamente o modelo de uso anterior, percebeu-se, claramente, que houve a necessidade de uma adequação na maneira como as empresas tradicionais criam, desenvolvem e negociam o *software*.

Embora os primeiros computadores comerciais fossem vendidos sem qualquer sistema operacional, e os usuários que adquiriram esses equipamentos tivessem a necessidade de desenvolver seus próprios aplicativos, e os dissemina quase de "mão em mão", podendo ser considerados como o surgimento do processo de criação do *software* livre, considerados, portanto, anteriores ao modelo de desenvolvimento do código proprietário, o modo de desenvolver aplicativos com o uso de comunidades surgiu tão somente em 1992, quando o sistema *Linux* surgiu na Finlândia.

Com a popularização da *internet*, houve um crescimento dos aplicativos criados livremente pelo mercado amplo que os desenvolvedores tinham acesso. Qualquer computador com acesso à *internet* era um alvo em potencial para que houvesse um novo aplicativo, ou novas funções para programas já existentes, fazendo assim com que não só os aplicativos livres se desenvolvessem, como também o

processo de criação desses aplicativos.

Enquanto esse processo de popularização não acontecia, os atuais gigantes do *software*, tais como *Microsoft, Computer Associates, Apple, IBM*, entre outras, desenvolviam o modelo mais utilizado de criação de código fonte atualmente, o modelo proprietário.

Dessa forma houve a criação e o desenvolvimento do atual paradigma tecnológico, o *software* proprietário. Com o embate entre esses modelos, e com a inclusão de alguns desses gigantes do modelo proprietário em algum grau no modelo de *software* livre, houve não só a criação desse novo paradigma tecnológico, como houve também uma quebra dele, claramente criando uma trajetória independente no mesmo mercado, o *software* livre.

Surgiram nesse mercado algumas empresas que são, hoje, também consideradas como gigantes do *software*, como a americana *Red Hat,* a israelense *Mirabilis* (desenvolvedora do comunicador instantâneo *ICQ)*, a brasileira *Mandriva*, desenvolvedora do *Mandriva Linux*, entre outras, todas elas utilizando o modelo livre de criação, tendo seu conhecimento acumulado disponível no mercado, tanto para usuários finais quanto outros desenvolvedores.

No momento em que sumariou-se as principais conclusões deste estudo monográfico, a empresa *Microsoft*, através do diretor do *Open Source Software Lab da Microsoft*, Sam Ramji, divulgou um *webcast*[33] com uma análise de como o movimento do código livre influenciou no desenvolvimento do *Microsoft Windows Server* 2008[34].

Assim, concluí-se que, com a utilização ampla da teoria evolucionista, o *software* livre e sua repercussão no mercado atual pode ser considerado como uma trajetória independente do atual paradigma tecnológico, podendo ser também entendida como um modelo de desenvolvimento em amadurecimento, fazendo com que haja uma adequação do mercado e de outros desenvolvedores.

---

33 Transmissões de vídeos via *internet*, com aproximadamente 45 minutos de duração

34 A transmissão foi feita no Brasil, no dia 08 de agosto de 2008. O artigo foi divulgado inicialmente através do endereço http://port25.technet.com/archive/2008/02/27/opening-windows-server-2008.aspx

REFERÊNCIAS

BERKUS, J.. 5 types. Disponível em: <http://www.powerpostgresql.com/5_types>. Acessado em 12 mai 2008.

DEPRET M. H.; HAMDOUCH, A. **Les Societés de Biotecnologie sont-elles menaces?** Biotecnologie et Finances. 2000.

DOSI, G. PAVITT, K. SOETE, L. **The economics of technical change and international trade.** Hertfordshire, Inglaterra. Harvester Wheatsheaf. 1990.

DOSI G.; FREEMAN C.; NELSON R.; SOETE L.; SILVENBERG, G. **Technical Change an Economic Theory.** Pinter, Londres, 1988.

DOSI, G. **Techincal Change and Industrial Transformation.** St. Martin's Press. New York, 1984.

DOSI, G. **Technological Paradigms and Technological Trajectory:** A **suggested interpretation of the determinants and directions of technological change.** Research Policy, Vol. 11, nº 3, pp. 147-162, 1982.

FARRER, H. BECHER, C. G. FARIA, E. C. MATOS, H. F. SANTOS, M. A. MAIA, M. L. **Programação Estruturada de Computadores:** Algoritmos Estruturados. Terceira edição. Belo Horizonte. LTC - Livros Técnicos e Científicos. 1999.

FISL – Fórum Internacional de *Software* Livre. Disponível em <http://fisl.softwarelivre.org/8.0/www/>. Acessado em 14 maio 2008.

FRIEDMAN, THOMAS L. **O Mundo é plano:** Uma breve história do século XXI. Rio de Janeiro. Editora Objetiva, 2005.

FREE SOFTWARE FOUNDATION, disponível em: <http://www.fsf.org/>. Acessado em 05 mai 2008.

GNU: GNU IS NOT UNIX. Disponível em: <http://www.gnu.org/home.pt.html>, Acessado em 15 mai 2008.

GNU PUBLIC LICENSE; disponível em: < http://gplv3.fsf.org/gpl-draft-2007-03-28.html>. Acessado em 05 mai 2008.

HERAN, F. **De L'organisation Fonctionnelle à L'organisation integer.** Working Paper, Université de Lille-I, 1998.

JACKSON, A. M. ***System Development***. Englewood Cliffs. Prentice Hall, 1983

KING, D. **Criação de *Software***: Técnicas Eficientes. Rio de Janeiro. Campus, 1989.

NELSON. RICHARD R. WINTER. SIDNEY G. **Uma teoria evolucionária da mudança econômica**. Campinas. Editora da Unicamp, 2008.

NELSON, R. R, WINTER, S. G. **An Evolutionary Theory of Economic Change.** Cambrige, Harward University Press, 1982.

NUNES, L. L. V. N. K. Aspectos Sociais do Uso do Software Livre. Disponível em: <https://twiki.im.ufba.br/pub/MAT159/Tema5/SoftwareLivre.pdf>. Acessado em 15 mai 2008.

PAVITT, K. ***Sectoral patterns of technological change:*** *Towards of taxonomy theory, research policy,* pp 343-373.

PORTAL, O QUE É *SOFTWARE* LIVRE. Disponível em <http://www.softwarelivre.gov.br/SwLivre/>. Acessado em 08 mai 2008.

OPEN SOURCE INICIATIVE. Disponível em: < http://www.opensource.org/docs/osd>. Acessado em 08 mai 2008.

ROSENBERG. NATHAN. **Por dentro da caixa preta**. Campinas. Editora da Unicamp, 2006.

SUPER INTERESSANTE: Dito e Feito. São Paulo. Ano 8, nº 11. Novembro 1992. Editora Abril,

THE INQUIRER, disponível em: <http://www.theinquirer.net/default.aspx?article=39291>. Acessado em 13 jun 2008.

WIKIPEDIA, a enciclopédia livre. Software Livre. disponível em: <http://pt.wikipedia.org/wiki/Software_livre>. Acessado em 12 mai 2008.

WIKIPEDIA, a enciclopédia livre. Código Aberto. Disponível em: <http://pt.wikipedia.org/wiki/C%C3%B3digo_aberto>. Acessado em 08 mai 2008.