



**Faculdade de Ciências e Letras de Araraquara  
Departamento de Economia**

**Desenvolvimento de Software Livre/Código Aberto no Brasil**

**Nome do Aluno:** Felipe Augusto Assad de Oliveira

**Nome do Orientador:** Prof. Dr. Eduardo Strachman

**Nome do Examinador:** Prof. Dr. Rogério Gomes

Araraquara, dezembro de 2011

FELIPE AUGUSTO ASSAD DE OLIVEIRA

DESENVOLVIMENTO DE SOFTWARE LIVRE/CÓDIGO ABERTO NO  
BRASIL

Monografia apresentada como exigência parcial  
para a conclusão do curso de Ciências Econômicas  
da Universidade Paulista “Júlio de Mesquita Filho”,  
sob a orientação do Prof. Dr. Eduardo Strachman

Araraquara, dezembro de 2011

## Resumo

O presente trabalho propõe-se a discutir os determinantes do processo de desenvolvimento de software livre, seus efeitos na indústria de software e as ações do governo em relação a estas iniciativas

Impulsionados, principalmente, pelos impactos dos avanços nas tecnologias de informação e comunicação e por inovações de processos relacionados a estes progressos, projetos de desenvolvimento de Software Livre/Código Aberto vêm crescendo em ritmo cada vez mais acelerado e tornando-se alvo de estratégias de empresas e políticas públicas. A partir desta constatação, o trabalho procura relacionar o arcabouço teórico que visa explicar estas iniciativas com dados da indústria brasileira de software e verificar como este modelo de produção pode modificar e agregar às suas feições. Foi realizado também um balanço das intervenções do governo brasileiro relacionadas a software livre/código aberto.

**PALAVRAS CHAVE:** software; software livre; FOSS; copyright; copyleft; Brasil; inovação; tecnologia aberta

## **Agradecimentos**

Em primeiro lugar agradeço à família. Principalmente à minha mãe, Cilane Assad de Souza, a pessoa que me ensinou a importância de lutar pelas coisas importantes, pelo exemplo de perseverança e pelo carinho. Agradeço ao meu irmão Daniel Augusto Assad de Oliveira pela fonte de inspiração, desde o traçar dos objetivos até o tema desta monografia e pelos valiosos exemplos durante todos estes anos. Agradeço a Neusa Mendonça Assad de Souza por todas as oportunidades oferecidas, pela fé e por todo o carinho nos melhores e nos mais difíceis momentos.

Ainda, em relação à família, agradeço à república Smurféticos - a cada um dos seus moradores, passados e presentes - por toda a amizade, apoio e troca de experiências durante estes que, por causa de vocês, foram os melhores anos de minha vida!

Agradeço ao Professor Eduardo Strachman por acreditar neste tema, pela ajuda concedida durante todo o processo e pelas valiosas aulas ministradas.

Finalmente, agradeço à Universidade Estadual Paulista. Somos resultado das experiências que vivemos e das pessoas que conhecemos, portanto, o mais sincero agradecimento por tamanha oportunidade de crescimento pessoal e acadêmico.

## Conteúdo

<b>Introdução</b> .....	<b>1</b>
<b>1. Economia do Desenvolvimento de Software Livre/Código Aberto</b> .....	<b>4</b>
1.1. Definição de Software e Software Livre/Código Aberto.....	4
1.2. Desenvolvimento de Software Livre .....	8
1.3. Motivações Para a Participação de Indivíduos.....	11
1.4. Governança de Projetos.....	14
1.5. O Mercado de Software: Caracterização dos Segmentos.....	15
1.6. Interação Entre Empresas e Software Livre.....	17
1.7. Inovação.....	21
<b>2. O Mercado de Software Brasileiro e a Influência do Software Livre</b> .....	<b>25</b>
2.1. Tipologia de Segmentos e Feições do Mercado Brasileiro .....	25
2.2. Desenvolvimento Socialmente Eficiente de Software Livre.....	30
2.3. Influência de Software Livre nos Segmentos de Mercado .....	33
<b>3. Intervenção Pública Para Software Livre/Código Aberto</b> .....	<b>41</b>
3.1. Intervenção Estatal: Argumentação .....	41
3.2. O Arcabouço Institucional da Indústria Brasileira de Software.....	46
3.3. Análise da Ação Governamental.....	49
<b>4. Conclusão</b> .....	<b>53</b>
<b>5. Bibliografia</b> .....	<b>54</b>

## Índice de Ilustrações

Figura 1- <i>Evolução: Projetos de Software Livre/Código Aberto criados</i> .....	2
Figura 2 – Esquematização da Disposição de Membros de um Projeto SLCA.....	14
Figura 3 – Síntese e Tipologia Das Estratégias Adotadas .....	19
Figura 4 – Curva de Demanda Por Serviços de TI.....	21
Figura 5 – Segmentação da Indústria Brasileira de Software.....	25
Figura 6 – Segmentação da Indústria de Software – Empresas Nacionais.....	26
Figura 7 – Segmentação da Indústria de Software – Empresas Estrangeiras .....	27

## Introdução

Nos últimos anos foi possível observar vários avanços nas tecnologias de informação e comunicação que possibilitaram reduzir distâncias e aumentar as possibilidades de interação. Assim, as ampliadas possibilidades de comunicação incentivam a difusão de conhecimento e inovações. Inovações das mais variadas, que vão de grandes redes sociais até modelos de negócio baseados na web, são resultado direto destas mudanças estruturais. A indústria de software é possivelmente uma das que mais diretamente baseia-se neste arranjo estrutura e também é uma das mais pervasivas. Os produtos deste segmento são utilizados como insumos pelos mais diversos setores da economia.

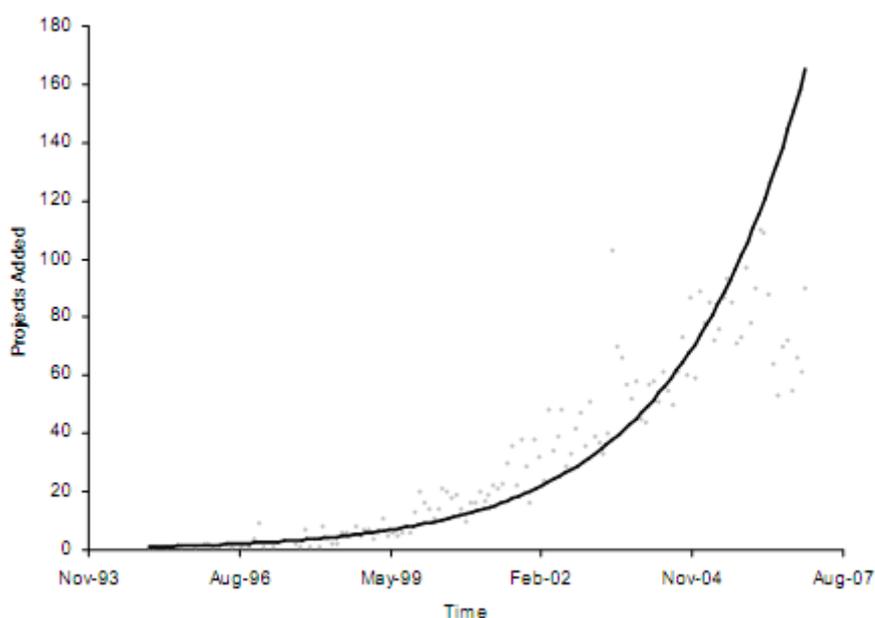
Estes produtos são produzidos das mais diversas formas sob os mais diversos arranjos legais no tocante a direitos de uso e transmissão. Notadamente, o processo aqui estudado baseia-se na colaboração. Tapscott e Williams (2006) estudaram estes processos produtivos baseados em “colaboração em massa” e os caracterizaram como:

Novas infra-estruturas colaborativas de baixo custo - desde a telefonia grátis via internet até softwares de código aberto ou plataformas globais de terceirização - permitem que milhares de indivíduos e pequenos produtores criem conjuntamente produtos, acessem mercados e encantem os clientes de uma maneira que apenas as grandes empresas podiam fazer no passado (TAPSCOTT e WILLIAMS, 2006, p. 10)

Os autores ainda argumentam com base no artigo de Coase (1937) chamado “A Natureza da Firma”, na qual o mesmo teorizou sobre os custos de transação para explicar a colaboração em massa. Os principais fatores, tais como custos com coordenação, pesquisa e contratação como razões importantes para a organização vertical de empresas. Atualmente, o avanço das tecnologias de comunicação permitiu uma grande diminuição destes custos de modo que todo o processo produtivo pode ser repensado de modo a aproveitar o melhor dessas vantagens. A lei ainda continua válida, porém, agora, consegue explicar como esta nova estrutura configura um cenário onde empresas, governos e até indivíduos atuam na busca por melhores condições ao efetuar compras ou organizar processos cada vez mais complexos (Tapscott e Williams, 2006, p.52)

Neste contexto, iniciativas voltadas para a produção de software livre vêm crescendo cada vez mais. Longe de ser uma novidade criada a partir da conjuntura atual, estas iniciativas datam desde a década de 80 - a criação da Free Software Foundation data de 1985 e a primeira versão do núcleo Linux data de 1991. Porém, conforme observou Riehle (2008), o total de projetos baseados em código aberto assim como o total de código adicionado vêm crescendo em taxas exponenciais ao longo do tempo.

**Figura 1- Evolução: Projetos de Software Livre/Código Aberto criados**



Fonte: DESHPANDE e RIEHLE (2008), p. 7

Deste modo, a presente monografia procura realizar um estudo sobre o desenvolvimento de software livre no Brasil. Percebe-se o desenvolvimento de software livre como a materialização de esforços coletivos de indivíduos que se organizam espontaneamente em comunidades com regras e ideologias próprias. Imbuídas principalmente das noções de liberdade quanto à participação no desenvolvimento, uso e distribuição de suas criações. Estas comunidades conseguem produzir com sucesso software de qualidade e competitivo. A lógica destas, a princípio, diferencia-se da lógica do mercado na qual se situa, a saber, os softwares comerciais e os serviços de TI (tecnologia da informação). Desta maneira, iniciativas deste tipo encerram um enorme potencial de inovação, redução de custos, quebra de dependência tecnológica, descoberta de novos mercados e ampliação

dos já existentes além de possíveis mudanças no comportamento dos agentes que os compõem.

Visto que há grande dependência da tecnologia externa nesse segmento e dada sua influência sobre outros segmentos da economia, torna-se relevante estudar meios de incentivar a inovação neste segmento, o uso destes softwares livres e a diminuição de envios de renda ao exterior por pagamentos de softwares comerciais ou TI.

Neste sentido, é possível entender não só o efeito e a postura do mercado frente a estas, mas também a preocupação do governo com este assunto e sua postura frente à questão. Ao assumir papel de incentivador do desenvolvimento de software livre e adotá-lo como recurso estratégico para suas ações nas gestões da TI, do conhecimento e do governo eletrônico, suas ações e medidas alinham-se assim, com suas diretrizes de soberania tecnológica, transparência, aumento do bem-estar e do valor agregado.

O primeiro capítulo busca fazer uma revisão teórica acerca dos conceitos econômicos e técnicos relacionados ao desenvolvimento de software livre/código aberto. No segundo capítulo, foi realizada uma caracterização da indústria brasileira de software, o surgimento de modelos de negócio baseados em software livre e como e quando cada modelo de negócio é mais eficiente do ponto de vista da indústria como um todo. No terceiro e último capítulo, os conceitos teóricos da intervenção governamental visando estas iniciativas são expostos para que, em seguida, fosse feita uma análise das ações do governo brasileiro.

## **1. Economia do Desenvolvimento de Software Livre/Código Aberto**

Os determinantes do processo de desenvolvimento de Software Livre/Código Aberto são dados por motivações diferentes daquelas que determinam a produção de Software “normal”, pago, como a conhecemos. Basicamente, a única e principal diferença deste modo de produção seria disponibilizar gratuitamente o código fonte do que foi produzido. Porém, as características deste tipo de produto e o mercado configurado ao seu redor não só implicam que esta ação afeta todos os produtores de software, mas também a maneira como indivíduos, empresas e governos podem lidar com este tipo de aplicação.

Portanto, o entendimento adequado destes determinantes do processo de desenvolvimento de Software Livre/Código Aberto é de central importância para entender como se dá e como se pode expandir o mercado baseado em Software Livre/Código Aberto.

Este capítulo procura caracterizar o que é o software enquanto bem, procurando estabelecer como sua produção é vista do ponto de vista econômico para depois explicitar as linhas gerais e a interpretação econômica de como o software livre/código aberto é desenvolvido. A seguir, será exposta a tipologia do mercado de software e algumas concepções teóricas acerca de seu funcionamento a fim de que, enfim, seja possível perceber como a entrada desta variante do modelo tradicional afeta o mercado.

### **1.1. Definição de Software e Software Livre/Código Aberto**

Primeiramente, a fim de possibilitar a compreensão da natureza do desenvolvimento de software livre/código aberto é necessário revisar brevemente os conceitos de código fonte e direitos autorais.

O código fonte é o conjunto de instruções lógicas padronizadas em determinada linguagem de programação, as quais são compiladas pelo computador (*hardware*), isto é, traduzidas para sua linguagem, de modo que este seja capaz de realizar determinadas operações. O código fonte, então, é a essência do software, o que define o funcionamento de um computador frente uma tarefa. Como presente no

“Estudo sobre o software livre comissionado pelo Instituto Nacional da Tecnologia da Informação (ITI) FGV”.

“código fonte” é o que permite o entendimento do código-objeto ligando o computador ao programador. É ele também que permite o acesso e o estudo do conhecimento incorporado na construção do software. “Acima de tudo, é o acesso ao código fonte que permite que modificações possam ser feitas no programa” (FALCÃO, LEMOS, et al, 2005, p. 4)

Para Pressman, o software é tido como um não-objeto, uma não-coisa. “As soluções em software podem satisfazer imediatamente as necessidades dos indivíduos, como bens finais de consumo ou, indiretamente, como “meio de produção” (PRESSMAN, 2002). Sua produção em escala não necessita de insumos e sua forma de apropriação é a propriedade intelectual, ou seja, patentes e correlatos. Conforme assinala o autor, a análise econômica de atividades baseadas em software baseia-se no seu processo produtivo e nas estruturas de mercados que se formam por conseqüência. Logo, temos que:

i. o software é planejado e desenvolvido (“engineered”,) mas não é manufaturado, fazendo com que os custos estejam concentrados na fase de engenharia ou design; ii. o software não sofre desgaste ou depreciação física, e as falhas no seu funcionamento não são resultado de desgaste, mas derivam de deficiências em seu design; iii. a despeito da evolução recente nas metodologias de desenvolvimento, o software é ainda majoritariamente produzido manualmente (PRESSMAN, 2001, pp.6-8).

Tem-se um bem cujo processo produtivo é intensivo em mão-de-obra e com relativo grande peso na etapa de *design*. Depois de criado o produto e primeira unidade, cópias integrais podem ser feitas subseqüentemente e o código fonte – que é a substância do software pode ser reutilizado para outras aplicações – tido como reuso de conhecimento a um preço próximo de zero por aquele agente que detém sua propriedade, reflexo, portanto, de sua característica de bem imaterial. Sua reprodução não necessita de um processo fabril – no caso da realização do *download* de determinada aplicação, por exemplo, de modo que seu custo marginal (de reprodução) é quase nulo. Tem-se, então, uma predominância dos custos fixos ligados à etapa da concepção.

Assim, a escala é essencial para as empresas deste setor. Situações de oligopólio ou até monopólio não são raras em vários segmentos. Isso acontece pois as principais barreiras à entrada estão ligadas aos custos de concepção e ao *market share* das outras empresas. Ou seja, o grau de adoção de um programa é essencial para seu sucesso e, com isso, a principal estratégia de empresas situa-se em conseguir estabelecer um sistema de auto-reforço, de causação cumulativa. A

estratégia para chegar ao ponto mais alto do mercado é de suma importância para empresas e se dá principalmente pela descoberta de novos segmentos (vantagens de pioneirismo) ou da imposição de padrões dominantes.

“Para muitos produtos em software o valor depende não apenas de suas características intrínsecas, mas se eleva com o número de usuários da mesma solução compatível” (MESSERSCHMITT & SZYPERSKI, 2000, p.8).

Portanto, não necessariamente, a adoção de um software depende apenas de este ser o melhor ou o pior do ponto de vista técnico, mas sim do número de usuários que o adotam, de modo que surgem – e ganham importância - efeitos de rede. Estes são muitas vezes externos ao desenvolvimento da aplicação em questão. Existe um custo relacionado ao tempo despendido de aprendizado para um programa e à questão da interoperabilidade, a capacidade de um programa lidar com diferentes tipos de arquivo. Caso existam dois programas concorrentes para a realização de uma mesma tarefa e que gerem arquivos incompatíveis entre si, haveria não só uma concorrência entre estas aplicações, mas sim pela hegemonia do formato de arquivo aceito e das operações para realizar as várias funções do software. Portanto, para evitar custos referentes à adequação de formatos e/ou aprendizado de diferentes aplicações, os usuários tendem a adotar poucas ou apenas uma única aplicação.

A partilha dos arquivos gerados, portanto, cria efeitos de rede. Com isso, temos que:

“Quando alguém se integra à rede, ela fica melhor e maior, beneficiando a si próprio e aos demais integrantes. Esse efeito é auto-alimentado na medida em que a expansão da base de usuários faz com que um número maior de pessoas acredite valer a pena adotar o sistema, gerando um ciclo virtuoso de crescimento. (...) A economia de rede ergue enormes barreiras à entrada, consolidando padrões de facto (GUTIERREZ & ALEXANDRE, 2004, p.30).

Além da questão dos efeitos de redes e sistemas de auto-reforço, há de explicitar a característica de pervasividade do software, ou seja, a difusão e a infiltração destes em diversos setores da economia. Uma grande parcela destes setores depende do desenvolvimento do complexo eletrônico e de sua posterior sinergia com o setor microeletrônico. É importante considerar o crescimento das atividades intensivas em conhecimento. Com isso *“a proporção de trabalho que*

*simplesmente* manuseia bens tangíveis, ao longo do processo produtivo, tem cada vez mais se tornando menos significativa do que a proporção do trabalho responsável pela produção, distribuição e processamento do conhecimento” (CASSIOLATO, 1999, p. 172). As tecnologias relacionadas à informática estão relacionadas ao fluxo de informações e conhecimento caracterizando um novo padrão “tecno-produtivo”, conforme assinala (Roselino 2006). Athereye define a indústria do software como:

“mais do que uma indústria qualquer – produz um bem intermediário central na nova economia digital. O seu papel é análogo ao papel desempenhado pelo setor de bens de capital numa economia de base metal-mecânica” (Athereye 2005, p.7).

O software é visto como um bem intermediário, sendo necessário para a utilização de diversos equipamentos, de máquinas de lavar a computadores de bordo de locomotivas até boa parte daqueles equipamentos eletrônicos existentes no setor financeiro. Essa característica é denominada transversalidade (Roselino, 2006, p. 12). Assim, o software passa a ser uma ferramenta essencial para a competitividade de empresas e países, em vários setores de suas economias, atraindo atenção de formuladores de política e mesmo de indivíduos que percebem ganhos de bem-estar em alguma parte de seu dia-a-dia.

Sobre o código fonte de um software, quando este não faz parte do seu conteúdo, adjetiva-se este software como de “código fechado”. A consequência disso é que não é possível compreender na totalidade a lógica de seu funcionamento, nem as interações entre os recursos ou realizar modificações em sua estrutura. No Brasil, esta característica é assegurada através do conteúdo disposto na lei número 9.610 de 1998 que regula os direitos autorais e de reprodução, conhecidos como *copyrights*.

Por outro lado, quando o código fonte é disponível para visualização, alteração e até posterior distribuição, adjetiva-se a aplicação como de “código aberto”. Para que este mantenha estas características – de acordo com o(s) desejo(s) de seu(s) criador(es) - é distribuído com uma licença especial a ele anexada, o “*copyleft*”. A lógica de licenças “*copyleft*”, não é exatamente a oposta do arcabouço legislativo contido na noção de “copyright”. Nas palavras de Moniz e Cerdeira:

”o copyleft, surgido nos EUA, nada mais é do que o próprio instituto do copyright em que o autor libera, desde o licenciamento primeiro, os direitos

de uso, reprodução, distribuição e, eventualmente, de alteração de sua obra a qualquer interessado. Não traz, de fato, alterações substanciais nos princípios clássicos salvo o de, por meio de apropriado contrato de licença, permitir tais liberdades. (MONIZ, CERDEIRA, et al, Eficiências Tecnológica, Econômica e Social, 2004, p. 68).

Este tipo de licença pode ser útil para prevenir “seqüestro comercial”, ou seja, que algum tipo de instituição apodere-se dos direitos do software, registrando-o sob determinado tipo de patente, por exemplo. Podemos separar diferentes arranjos de licenças “*copyleft*”, em três categorias principais, as quais determinam o caráter do software código aberto.

- **Recíprocas totais**, que exigem que o código fonte do software e todas as adições a este sejam totalmente disponíveis. A mais conhecida licença desta categoria é a licença GPL, criada pela Free Software Foundation (FSF), entidade sem fins lucrativos que objetiva a disseminação do software livre. Chama-se de “software livre”, portanto, toda aplicação que se encaixa nesta categoria.

- **Recíprocas parciais**, as quais são um pouco mais flexíveis e permitem que um software livre seja anexado ou tenha a ele anexados componentes não livres. Este tipo de arranjo pode ser exemplificado pelo conteúdo disposto nas licenças LGPL (também elaborada pela FSF) ou EPL (EclipsePublicLicense).

- **Permissivas**, que praticamente não impõem nenhum tipo de restrição e permitem inclusive que seja feito um novo licenciamento com condições diferentes como, por exemplo, redistribuí-lo em termos não livres. Exemplo deste tipo de licença é a BSD (Berkeley Software Distribution).

Portanto, para um software, o que determina as qualidades de livre ou proprietário é o tipo de arranjo de licenças sobre os quais este está embasado e não – ao contrário do senso comum – se é cobrada uma taxa sobre a distribuição. Deste modo, pode haver software com código fechado ou aberto distribuídos gratuitamente ou não. Tudo depende da licença sob a qual é distribuído, que é elaborada de acordo com o modo como este é produzido, o qual, por sua vez, é resultado da interação social dos programadores.

## 1.2. Desenvolvimento de Software Livre

Raymond (1999) faz oposição entre os modos de produção de software proprietário e Software Livre/Código Aberto (doravante denominado SLCA) como catedral e bazar, respectivamente. A metáfora do bazar, diz respeito ao sistema de

inovação envolvendo vários programadores (desenvolvedores) e caracterizado pela ausência de uma unidade centralizada que dita as decisões

Generalizando, para funcionar, a direção do desenvolvimento de SLCA visa: integração dos usuários no desenvolvimento do código; auto-seleção dos programadores para realização das tarefas que melhor se ajustam a suas habilidades; design coincidente com a resolução de *bugs*; atualização pouco espaçada temporalmente (Raymond 1999).

Estas características assinalam o fato de que o software desenvolvido nestes moldes é composto por várias partes pequenas e independentes, combinadas de forma coerente e que realizam tarefas complexas. Estas pequenas partes são denominadas “módulos” e a sistematização deste arranjo de *design* é denominada “arquitetura modular”. Isto também assegura que qualquer mudança em um dos módulos terá nula ou limitada influência sobre o restante da aplicação, de modo a permitir que os programadores trabalhem sem medo de interferência. Estas características possibilitam maior agilidade e qualidade do processo e assinalam para uma inovação de processo em comparação ao modelo de produção do software proprietário o qual é desenvolvido como um todo e de maneira hierarquizada, com interação entre todos os recursos como veremos a seguir.

A arquitetura modular implica em um acréscimo em velocidade e qualidade do desenvolvimento enquanto dificulta a gestão de projetos já que poderia causar um desperdício de esforços, caso dois ou mais programadores se propusessem a realizar a mesma tarefa. Portanto, um projeto deve ser constantemente atualizado para mitigar estes efeitos. Conforme assinalam Kogut e Methiu (2001) faz-se necessário que os processos de *design* e resolução de *bugs* sejam sincronizados, aliados à integração dos usuários na produção do código fonte, para assim contribuírem para um melhor aproveitamento do potencial criativo. A arquitetura modular possibilita que o ritmo de produção do SLCA seja dado de acordo com a performance do membro mais produtivo da cadeia de produção. Em contraste, o modelo do software proprietário é caracterizado pelo design e subsequente resolução de *bugs*, de modo que tem seu ritmo ditado de acordo com a performance do membro menos produtivo.

O desenvolvimento de SLCA é descrito por Von Hippel e Von Krogh (2003) como um modelo privado-coletivo de inovação, o qual depende da integração de usuários no desenvolvimento da aplicação. Um indivíduo só participará de um projeto se os benefícios de contribuir para o bem público forem maiores do que os custos desta ação. Os resultados do trabalho podem ser apropriados privadamente uma vez que qualquer inovação feita é revelada. Uma importante ação para diminuição de esforços – e, conseqüentemente, de custos é o reuso de conhecimento nos diferentes estágios de produção, no sentido de uma mesma solução servir à resolução de múltiplos problemas.

O código fonte é o conhecimento acerca do funcionamento de um software tanto para indivíduos quanto para a própria máquina que vai executar uma tarefa, portanto, desenvolver software é também uma atividade de resolução de problemas. Indivíduos de fora do projeto que se engajam em comportamentos *free-rider* lograriam acesso ao código fonte, mas estariam privados dos conhecimentos tácitos de outros membros do projeto e não conseguiriam auferir os mesmos benefícios destes. Assim, os autores consideram o resultado do processo de inovação como um bem público, porém não puro, uma vez que é necessário fazer parte do grupo para ter acesso ao bem em sua completude.

Enquanto seria crível, em modelos coletivos de desenvolvimento que aumentos no número de participantes aumentariam custos de vigilância contra comportamento *free-rider*, temos que, neste caso, cada novo participante é um potencial desenvolvedor ou sinalizador de novos *bugs*. Benkler (2001) aponta que neste modelo de inovação privado-coletivo, cada pessoa possui conhecimentos inatos e pode ser um especialista em determinada área, e se, caso optasse por deixar seu posto, levaria consigo seu *know-how*, enquanto que no modelo de inovação privado-coletiva, cada membro da comunidade opta por contribuir na área que melhor conhece. Argumento que se apóia na característica de construção modular do SLCA. As contribuições individuais devem ser suficientemente pequenas de modo que os ganhos proporcionados (pelas contribuições) sejam maiores que os gastos de coordenação de agentes esparsos.

### 1.3. Motivações Para a Participação de Indivíduos

A participação de indivíduos em um projeto do qual podem não receber nenhum benefício material a priori parece contradizer o conceito do “homem econômico”, racional e maximizador, de conseguir vantagens para si. Porém, existem motivos que pesam no cálculo de custo/benefício que um indivíduo realiza para decidir se participa ou não desse tipo de atividade.

Com a expansão e notoriedade dessas aplicações, surgem motivações ligadas ao prestígio destes desenvolvimentos e/ou simplesmente ao potencial de aprendizado deles decorrente ou, ainda, ao fato de pertencer a um determinado grupo social, etc. Podemos dividir as motivações em duas categorias, de acordo com Rossi (2004):

- **Extrínseca:** Referente aos benefícios trazidos imediatamente mais os benefícios futuros da participação em um projeto - em geral, compensação monetária, reputação, aprendizado, aperfeiçoamento de performance, satisfação de uma necessidade de aplicação ainda não existente ou encontrada, etc.

- **Intrínseca:** Valoração em si mesma (“*per se*”): Referente à ação de um indivíduo motivado por satisfação, diversão ou desafio proporcionado pela ação ao invés da recompensa material que ela pode oferecer. Mais além, cita-se Lindenberg (2001): participação motivada por benefícios associados à participação/sujeição a normas sociais.

#### **Motivações Extrínsecas:**

O artigo de Lerner e Tirole (2002) foca no papel das motivações extrínsecas. Um indivíduo só participará do desenvolvimento de um software livre se houver benefício líquido, dada a soma dos benefícios presentes, como criação de solução para algum problema por eles encontrado, mais os benefícios futuros da participação em um projeto comercial ou SLCA, como, por exemplo, a existência de um sinal de incentivo como uma gratificação associada a um aumento de reputação ou ofertas de trabalho, melhoria da performance, etc.

Basicamente, existem dois aspectos nos quais se baseiam as motivações extrínsecas: um sinal de incentivo, no sentido de obtenção de notoriedade ou

oportunidades melhores de trabalho (conhecimento, etc) em projetos não SLCA e/ou a obtenção de uma solução para um problema (a criação de uma ferramenta que permita fazer algo que não era possível anteriormente).

Quando o agente realiza o cálculo de custo/benefício associado à sua participação em um destes projetos, leva em conta os dois aspectos acima mencionados. Ademais, enfrenta custos referentes à criação de reputação, os quais podem ser dirimidos dada a possibilidade de o programador (desenvolvedor) já estar familiarizado com a linguagem usada no projeto em questão por conta de uso prévio em alguma atividade. Do lado dos ganhos, a “publicidade” do código fonte reflete de forma mais direta o trabalho realizado por cada “contribuinte” e seu talento, o que favorece uma maior fluidez de informações a respeito de competências dos desenvolvedores.

Esta busca por reputação não é a única variável que explica a participação em um projeto de SLCA, pois esta não é capaz de explicar por que um “desenvolvedor de ponta” participaria de um projeto. Afinal, nem todos os participantes de um projeto serão líderes. Logo, o grau de exposição do trabalho de cada um depende diretamente das decisões tomadas pelos líderes do projeto, de modo que este trabalho depende de uma alocação de poder decisório a respeito de quais linhas de código fonte serão adicionadas à versão final do programa. Outro aspecto é que do número total de participantes da maioria destes projetos, apenas uma pequena porção destes insere linhas de código fonte. A maior parte testa e aponta erros de programação. Participação esta que dificilmente seria explicada pela busca por reputação.

Portanto, a segunda razão para participação é a satisfação de alguma necessidade que não era possível previamente. Estas invenções feitas pelos próprios usuários de um programa estão aliadas a baixos custos de sua divulgação proporcionados pela internet. A revelação de inovação só ocorreria se os ganhos gerados fossem maiores que os custos de disponibilizar a inovação. Estes são pressionados para baixo por efeitos de rede que multiplicam a disponibilização da informação, de modo a justificar também a atualização de programas. A heterogeneidade da crescente massa de usuários, que leva ao crescimento das comunidades, pode conduzir a diferentes avaliações deste custo de oportunidade.

Deste modo, quanto mais usuários fazem parte de um grupo, maiores as chances de ocorrerem adições úteis ao código fonte e mais facilmente estes progressos estariam disponíveis. Tem-se uma situação que favorece a revelação gratuita de detalhes de inovações feitas por estes usuários.

### **Motivações Intrínsecas**

Dentro desta categoria, encaixam-se as motivações que vão do prazer de exercer a atividade de programação - motivos socialmente orientados como a participação/sujeição a normas sociais (e suas obrigações), razões ideologicamente fundamentadas, altruísmo, etc.

O ambiente de ação dos programadores parcialmente favorece o experimentalismo conferido pela autonomia na programação e, por consequência, uma oportunidade de exercitar a criatividade, tudo isto possibilitado pelo custo praticamente nulo de comunicação entre os diferentes indivíduos que participam de determinada comunidade.

Assim, a satisfação individual obtida da participação em projetos de SLCA certamente tem papel importante, porém um aspecto altruísta de cada um também faz parte da ação destes. Raymond (1999) caracterizou este como um ambiente no qual a escassez econômica não é uma questão a ser atacada e que *“seu status social é determinado pelo que você doa ao invés do que você tem”*. Estas características contribuem para a formação de uma identidade dos colaboradores com a comunidade da qual fazem parte, notadamente, uma cultura de doação. Assim, pode surgir uma generalização de reciprocidade assegurada pelo arranjo de licenças que cobrem determinado projeto de uma comunidade (assegurando que estas não seriam expropriadas) e pela identificação com a comunidade e suas idéias.

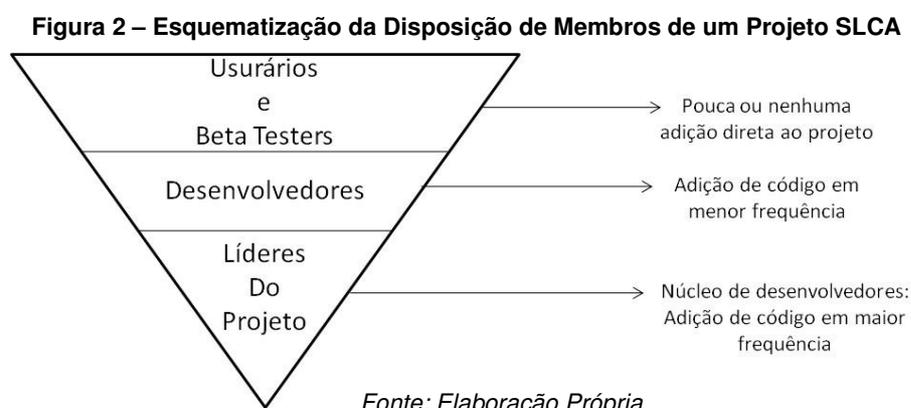
É difícil apontar qual o tipo de motivações que predominam na participação de projetos SLCA. Diversos autores observam a predominância deste ou aquele tipo em diferentes projetos. Porém, argumenta-se aqui que estas categorias são complementares. A maneira pela qual uma comunidade lida com a dinâmica destas

interações determinará sua longevidade. Frank e Jungwirth (2002) percebem que a modelação de uma estrutura institucional que concilia “investidores” - que consideram exclusivamente uma decisão de custo-benefício – com “doadores” - que agem por força de idéias – caracteriza o desenvolvimento de SLCA. Esta modelação institucional é fruto dos esforços empreendidos na parte de governança dos Projetos.

#### 1.4. Governança de Projetos

Dentre vários autores que realizaram pesquisas empíricas acerca de como são coordenados os projetos de SLCA, Hunt e Johnson (2002) descobriram que as atividades realizadas são altamente concentradas em um núcleo de desenvolvedores.

Estes são seguidos de uma quantidade maior de pessoas que resolvem *bugs* seguidos de uma quantidade maior que apontam estes *bugs* ou sugerem outras alterações, conforme o diagrama abaixo, no qual, basicamente, a maior parcela da adição de código fonte é feita pelo menor grupo:



Logo, estas comunidades agregam grupos heterogêneos de desenvolvedores que realizam diferentes tarefas. Coordenar estas interações pode ser uma tarefa razoavelmente complexa e custosa, de acordo com o tamanho da equipe com que cada projeto conta. Projetos como Linux ou Chromium (navegador e sistema operacional do Google) têm donos efetivos, porém também são nomeados usuários para ajudar na coordenação. Por outro lado pode haver espécies de processos de eleição de comitês para a coordenação de projetos, como no caso do projeto Apache até o ano de 1996.

Aspectos de liderança e autoridade são essenciais para entender como a colaboração entre membros esparsos pode lograr sucesso. McGowan (2002) chama atenção para uma importante característica do processo de desenvolvimento - o arranjo de licenças não exclui nenhuma pessoa de ter acesso ao código de um programa, logo, um membro que fosse excluído de uma comunidade ainda poderia ter acesso a este e, em último caso, realizar “*forking*”, que consiste na criação de projetos paralelos que visam realizar a mesma função caso o projeto anterior não satisfizesse os padrões ou expectativas da comunidade. Porém, o principal problema em ser excluído destas comunidades, seria ser privado do acesso ao trabalho de outros programadores, conforme a hipótese do modelo de inovação privado-coletivo. Ou seja, ser impossibilitado de usufruir do capital humano – conhecimento tácito presente nelas, dificultando estas ações de *forking*.

Deste modo, há razões fortes pelas quais os membros das comunidades aceitariam sujeitar-se à hierarquia e regras impostas pelo meio, independente de sua identificação ou não com os ideais da comunidade, mesmo que este fator também venha a influir na decisão de cada um.

### **1.5. O Mercado de Software: Caracterização dos Segmentos**

Para a elaboração deste trabalho e discussão acerca do mercado de software, será utilizada uma tipologia semelhante à estabelecida por Roselino (2006) e pela ABES (2010). Estas abrangem distintos modelos de negócios tais como:

“o conjunto de empresas (públicas ou privadas), voltadas primordialmente ao desenvolvimento e comercialização de soluções em software, na forma de serviços, software desenvolvido sob encomenda, ou software comercializado como produto acabado” (Roselino 2006, p.34).

Dentro da categoria de serviços de software, pode-se dividi-la em atividades de baixo ou alto valor agregado. As atividades de baixo valor agregado:

“são caracterizadas pela inexistência de padrões tecnológicos relevantes, bem como o conteúdo pouco intensivo em conhecimentos específicos, resultando em menores possibilidades de diferenciação dos produtos/serviços, e no predomínio da concorrência baseada em preço” (AMICCI, 2004, p. 135).

Esta categoria abarca serviços relativos à manutenção de sites de internet, alimentação e manuseio de bancos de dados pouco complexos. O fato de este setor não ser capaz de impor padrões e ter sua concorrência baseada em preços assinala

para a sua dependência de soluções já existentes. Este setor, então, toma custos com software e hardware como pouco ou não variáveis – pelo fato do tamanho das empresas e por estas não exercerem nenhuma influência sobre estes de modo que a mão-de-obra é o fator que mais oscilaria em termos de preço.

Já as atividades de alto valor agregado:

“são atividades que abrangem o design de alto nível [como] os projetos e a modelagem da arquitetura de soluções em aplicações de software, assim com de bancos de dados complexos. Estas atividades envolvem, portanto, um conjunto mais complexo de funções, bem como o domínio de processos mais intensamente tecnológicos. Existe grande heterogeneidade de empresas voltadas para este segmento, dado o número de aplicações que encontra em diferentes setores da economia” (ROSELINO 2006, p. 38).

A maior parte deste setor é voltada para a fabricação de software sob encomenda que seja capaz de dar cabo de atividades complexas e, portanto, assume-se mais riscos ao se entrar em determinados projetos, dado o elevado grau de incerteza quanto à qualidade e funcionalidade de seus programas, quando comparados a atividades de baixo valor – por exemplo, neste setor, contratos celebrados geralmente possuem cláusulas para indenização do contratante caso a solução não funcione ou não seja satisfatória. Pois estas soluções não tomam somente padrões já existentes e tecnologias já difundidas. Dadas as especificidades da demanda por aplicações - muitas vezes, processos importantes e muito específicos para empresas, as empresas que ofertam estes serviços devem começar desde o estágio inicial de concepção, passando pela codificação, testes e, finalmente, produto final. Com efeito, a questão de escala torna-se mais importante, tomada como vantagem competitiva. Primeiramente pela credibilidade que uma gama variada de clientes pode passar e, em segundo lugar, pelas oportunidades de reuso de conhecimento para aplicações diferentes economizando esforços de programação.

O software produto, ou software pacote é aquele tipo de aplicação que contém um conjunto de recursos para satisfazer um determinado público alvo. Aquela parcela que não é contemplada por nenhum software pacote procuraria, então, solução em serviços de software de baixo ou alto valor agregado. O desenvolvimento de software produto (ou software pacote) é aquele em que as

características acima mencionadas do software manifestam-se com maior intensidade. Neste sentido:

“Software pacote é uma aplicação preparada previamente que serve a um conjunto amplo de clientes. (...) Neste segmento, a competitividade é definida pela capacidade de desenvolvimento técnico e de comercialização de produtos em massa. É alto o investimento necessário para desenvolver e lançar o produto e o retorno depende de sua aceitação pelo mercado” (MELO & CASTELLO BRANCO, 1997, p.2).

Neste setor, acontece uma interação entre o produtor e o potencial demandante, pois o software pacote é produzido pensando-se em nichos razoavelmente amplos, a fim de diluir os custos de produção. Com isto, a adoção de padrões, os efeitos de rede e os ganhos de escala são de central importância para uma estratégia bem sucedida. O potencial de inovações disruptivas passa a se tornar o principal fator de perturbação deste sistema integrando-se assim à atividade das empresas, que passam a investir parcelas crescentes em P&D.

#### **1.6. Interação Entre Empresas e Software Livre**

A existência de SLCA pode surtir efeitos distintos daqueles causados por software proprietário no âmbito do mercado de software como um todo, o que, por sua vez, requer uma estratégia de aproximação diferenciada por parte de seus agentes integrantes.

“While incumbents had to adapt their strategies to this newly emerging competition, OSS-based firms had to develop viable business models enabling them to generate profits.” (BITZER e SCHRÖDER, 2006, p. 2).

Um importante aspecto a ser considerado sobre o SLCA é que este é um tipo de bem que pode ter funções iguais às do software proprietário de modo a poder competir com ou complementar este. Sua inserção em mercados e as barreiras à entrada que estes enfrentariam seriam diferentes daquelas enfrentadas por softwares proprietários. Ademais, seu processo produtivo e organizacional diferenciado encerra potencial para produtos e serviços inovadores de modo a favorecer a descoberta de novos mercados e complementar a abrangência dos já existentes.

Dahlander e Magnusson (2005) investigaram a emergência de empresas que baseiam suas existências na utilização de recursos provenientes das comunidades a fim de se apropriarem e criarem valor a partir desses produtos. Boa parte do

conhecimento necessário para fazer softwares assim como boas partes destes, não são desenvolvidos dentro de empresas, mas sim, em comunidades que coexistem com empresas. Há diversas maneiras pela quais empresas constroem modelos de negócios sustentáveis.

As comunidades seguem diretrizes de funcionamento diferentes das firmas e não existem acordos contratuais entre as partes. As comunidades não estão inseridas em nenhuma forma de hierarquia das empresas. Logo, os insumos fundamentais - leia-se o código fonte aplicável possibilitado por capital humano não são de sua propriedade. West (2003) reconhece que empresas podem disponibilizar o código fonte de suas aplicações visando aumentar sua adoção – procurando cultivar ganhos pelos efeitos de rede – e acelerar o desenvolvimento tecnológico. Deste modo, as motivações de empresas são predominantemente econômicas, diferentemente daquelas das comunidades, fazendo com que a interação entre as duas partes possa ser problemática.

O direcionamento estratégico das comunidades é uma das maiores questões com as quais empresas deparam-se. Se as duas partes têm objetivos contrastantes, será difícil obter ganhos com o produto do trabalho das comunidades. O meio termo a estes são comunidades gerenciadas por empresas. Exemplo deste caso são os projetos Chromium que incluem o navegador da web Google Chrome (que experimentou aumento de 375% em seu market share agosto de 2009 e julho de 2011, saindo de 2,84% para 13,49%) e o sistema operacional Google Chrome OS (um dos primeiros sistemas operacionais baseados em computação em nuvem). Os autores argumentam que com uma imposição muito forte a respeito das estratégias do projeto, pode ser difícil gerar os incentivos necessários para que desenvolvedores juntem-se à comunidade. Por outro lado, com pouco controle, os recursos obtidos podem ser muito escassos e até improdutivos no caso de objetivos diferentes entre as partes.

Quanto às relações entre empresas e comunidades os autores as separam em três categorias: Simbióticas (empresa e comunidade ganham), comensalistas (empresa ganha e comunidade é indiferente) e parasitárias (empresas ganham e comunidades perdem) conforme tabela abaixo:

**Figura 3 – Síntese e Tipologia Das Estratégias Adotadas**

Synthesis and typology of approaches			
	Symbiotic (firm gains–community gains)	Commensalistic (firm gains–community indifferent)	Parasitic (firm gains–community loses)
Nature of relationship	Giving something to the community, often through a firm-established community	Search for useful input from the community	Search for useful input without obeying norms, values and rules
Possibility of influencing community	High	Low	None
Managerial challenges	Respect norms and values Obey licenses Resource consumption of developing community	Respect norms and values Obey licenses Getting acceptance of the community for using its resources in commercial applications	Avoiding direct conflicts
Operational means of subtle control	Attracting developers Aligning different interests Resolving ambiguity about control and ownership Devoting personnel to work in communities Creating and maintaining reputation Fringe benefits Interaction tools Selling development tasks	Devoting personnel to work in communities	

As estratégias simbióticas (empresa e comunidade ganham) são àquelas em que empresas desenvolvem a si mesmas e as comunidades. Neste molde, o processo de decisão é feito por ambas as partes (membros da comunidade e da empresa). Para que este molde seja possível, a empresa deve ser uma força ativa e ser vista como necessária dentro da comunidade, de modo que sua participação seja valorizada. Conseqüentemente esta poderia exercer influência na decisão quanto às estratégias da comunidade, seja tanto por disponibilizar código fonte desenvolvido internamente ou fornecer uma estrutura que permite a comunicação ágil entre os membros ou ainda outras funções. Exemplo desta estratégia, foi a adotada pela empresa MySQL (vendida para Sun Microsystems a qual foi comparada em janeiro de 2010) que auto intitula-se como “*a mais famosa base de dados de código aberto do mundo*” e hoje adota uma estrutura dual de licenças. Uma delas é uma versão livre e gratuita e outra é uma corporativa com restrições ao código e vendida como software tradicional.

Paralelamente, estratégias comensalistas - conforme o conceito, em que agentes beneficiam-se da existência com outra entidade sem prejudicá-la – seriam aquelas em que a empresa beneficia-se de recursos desenvolvidos nas comunidades sem deixar de participar ativamente do desenvolvimento do recurso do qual se utiliza.

Por outro lado, uma empresa pode adotar uma estratégia parasitária de modo a procurar recursos em diversas comunidades sem contribuir em nada para seu desenvolvimento.

A linha que separa o comensalismo e o parasitismo é tênue de acordo com o volume e relevância do trabalho promovido pela empresa dentro da comunidade, conferindo um caráter de dinamicidade quanto à interação das duas partes ao longo do tempo. Com efeito, a empresa pode ser vista como um agente nocivo e ter seu acesso à comunidade restrito de modo que esta não seria uma estratégia utilizada abertamente por nenhuma empresa, ou pelo menos por longos períodos.

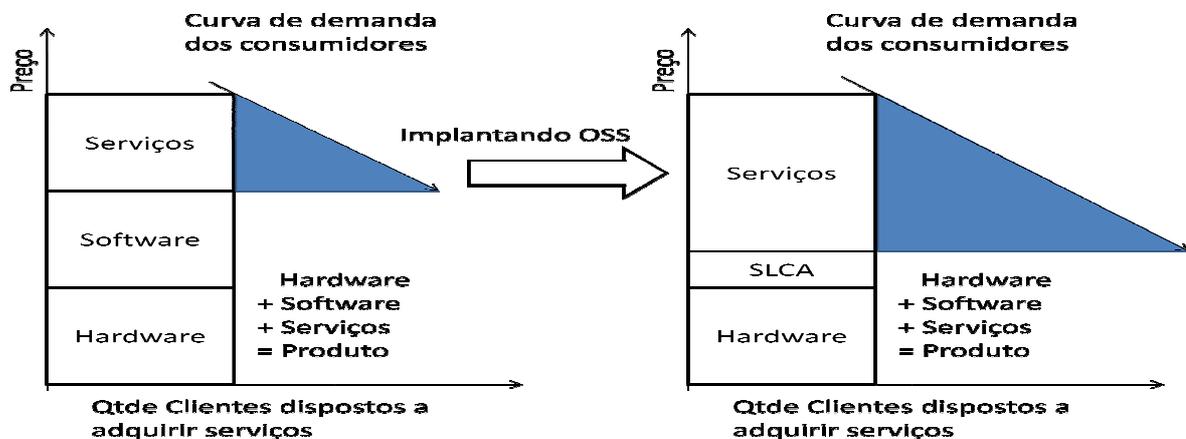
O modelo de negócios a ser seguido por uma empresa, apesar de promissor, encontra diversos empecilhos, pois deve ser capaz de atrair desenvolvedores alinhados com seus valores, transformar recursos de comunidades em soluções lucrativas e garantir sua aceitação ao longo do tempo. Ademais, disponibilizar trabalho e estrutura para estas comunidades, como forma de aumentar a atratividade de sua atividade e notoriedade, pode ser uma estratégia deveras custosa, pois o resultado do trabalho fica disponível para potenciais concorrentes e a incerteza quanto aos rumos do projeto é grande nos estágios iniciais.

Conforme Riehle (2007) esquematiza em seu artigo, para as empresas de TI, que são aquelas que comercializam “soluções” para empresas de modo a realizar a conjunção de hardware, software e serviços os quais compreendem a consultoria, montagem e treinamento em tal solução, seus custos são compostos por gasto com hardware e software— sobre os quais, geralmente, a empresa não possui direitos e deve pagar royalties/licenças de uso – e serviços.

Logo, com a adoção de SLCA, uma empresa de TI, pode diminuir seus custos com software e obter uma maior flexibilidade de preços, de modo a expandir a oferta do componente do fator do qual têm total propriedade, o qual corresponde pela maior parcela deste tipo de atividade, o fator serviços.

As curvas de demanda e oferta de serviços de TI podem ser descritas abaixo:

Figura 4 – Curva de Demanda Por Serviços de TI



Fonte: Riehle (2007), p. 4

Esta diferença de custos permite que a um mesmo volume de serviços prestados, o lucro obtido aumente, ou seja, o custo fixo diminui e permite a expansão do negócio até o ponto em que o lucro de serviços seja igual a zero.

## 1.7. Inovação

Bitzer e Schröder (2005) propõem-se a debater a questão da competição em indústrias intensivas em conhecimentos. Primeiro, os autores procuram entender o processo de produção e gestão do SLCA e a atuação das forças pró e contra inovação. Os autores procuraram evidenciar como a entrada de SLCA no mercado está associada à inovação.

A disponibilidade do código fonte para todos e da fácil comunicação entre os membros das comunidades possibilita uma fácil identificação de propostas de incrementos e mudanças do SLCA. Com efeito, isto possibilita uma classificação objetiva de módulos que competem entre si para entrarem no código fonte e determinar quais são os melhores. Assim, os autores só consideram como um progresso para o SLCA, uma ação que resulta em inovação conforme pode ser visualizado na tabela abaixo:

### **Características Pró Inovação do SLCA**

*Exército de programadores:* Grande número de programadores, comunidades abertas, ações de suporte e consciência coletiva de problemas.

*Difusão de conhecimentos:* O código fonte contém tanto o produto quanto a forma como este é produzido, grandes oportunidades de conhecimento

*Efeito motivação:* alta motivação dos desenvolvedores, cultura de doação, motivações intrínsecas

*Cooperação sem limites:* ausência de conflitos comerciais, sistema de inovação aberta e arquitetura modular

*Efeito usuário-desenvolvedor:* Conhecimento detalhado acerca das melhorias necessárias e a inovação que é conduzida próxima ao usuário

*Forking:* risco de forking

*Efeito Custos-zero:* não há gastos de P&D, investimentos de pesquisa a serem recuperados e disposição a abonar projetos que não levam a lugar nenhum

*Fonte:* J. Bitzer, P.J.H.Schröder / (2005) p. 12

O grande número de programadores é resultado do código livre gratuito. Universidades utilizam de SLCA com fins pedagógicos para aumentar o contato de seus alunos com código fonte. Na prática, maior número de desenvolvedores de modo a melhorar a percepção e reparo de bugs. Ademais, a difusão de conhecimento pode ser maximizada (Raymond 2000b). Com isto, teoricamente, qualquer um pode aprender a ser um desenvolvedor de SLCA.

O efeito motivação resulta em uma atratividade dos projetos, possibilita e incentiva os membros a realizarem um bom trabalho para que consigam reputação e aprendizado. Motivações estas, que são institucionalizadas e garantidas dado o arranjo de licenças de cada projeto. A cooperação existente nestes projetos por conta do fato que os membros envolvidos geralmente não têm interesses comerciais com o software desenvolvido. Com isto, aliado à característica de que mudanças em um determinado módulo não resultam acionam mudanças em outros módulos, tem-se que conhecimentos de programação complementares podem ser explorados.

O efeito do usuário desenvolvedor é de grande vantagem, pois estes têm grande conhecimento do funcionamento do software. Mesmo que estes sejam duas

pessoas diferentes, o trabalho do desenvolvedor é publicado, de modo que este pode ser contactado diretamente. O risco constante de o projeto não vingar ou não ser páreo às expectativas de seus desenvolvedores pode resultar em *forking*, o que pode acabar a ser uma motivação para o empenho dos desenvolvedores.

Essa ameaça influencia o processo decisório que pesa mais os aspectos tecnológicos na decisão dos rumos dos projetos e agiliza a inserção de inovações ao projeto. Ademais, dado que o custo de desenvolvimento é zero, pode-se esperar que soluções tecnologicamente superiores, mesmo que sejam menos viáveis economicamente, sejam levadas em consideração no processo decisório.

Por outro lado, estas características que funcionam como forças positivas também têm seu lado negativo além de outros aspectos que podem servir de empecilho à inovação, conforme os autores assinalam

#### **Características Anti Inovação do SLCA**

*Redundância:* esforços despendidos em vão, feitos repetidamente

*Processo não direcionado:* processo de desenvolvimento sem direção clara, itens faltando, inovação incompleta

*Forking:* divisão da base de dados, problemas de incompatibilidade, fragmentação dos efeitos de rede

*Aperto da oferta:* divisão das habilidades de programação em muitos projetos

*Problema de reuso do código:* reuso extensivo do código, difícil de localizar alterações ulteriores

*Fonte:* J. Bitzer, P.J.H.Schröder / (2005) p. 16

Um dos principais problemas encontrados é o desperdício de esforços. De fato, a independência de módulos e a liberdade com que contam os desenvolvedores possibilitam uma enorme quantidade de esforços que podem ser redundantes, ainda mais devido à ausência de uma direção centralizadora de todas as atividades. Ademais, não existe uma garantia concreta de que o SLCA forneça suporte adequado para o maior número possível de hardware/software.

Com relação ao forking, caso o risco se concretize, os desenvolvedores ficarão mais esparsos e podem aumentar os problemas de interoperabilidade de modo a aumentar os custos associados ao uso de tal solução e desacelerar os

efeitos de rede. Ainda, devido à abundância de código disponível, o lado negativo é que linhas ou até módulos inteiros podem ser copiados, de modo que código ruim seja replicado. Logo, uma vez que um módulo é incrementado, isto só é feito para um projeto, os demais projetos que o replicaram não teriam estes módulos atualizados.

## 2. O Mercado de Software Brasileiro e a Influência do Software Livre

Uma vez compreendidas as características do modelo de produção do SLCA, os diversos agentes interessados em utilizá-lo e como este pode inserir-se no mercado de software. Um mesmo tipo de software pode ser adotado de maneiras diferentes em determinadas regiões, de modo que se faz necessário, ainda, entender as especificidades locais aonde este se insere.

Nesta linha de pensamento, o capítulo seguinte procura analisar o mercado brasileiro de software em termos de divisão de segmentos, participação na geração de renda, pessoal empregado, participação de empresas estrangeiras e desempenho exportador. Aliado a este conhecimento, procura-se também entender teoricamente quais são os determinantes do desenvolvimento de SLCA, ou seja, entender quais são as situações onde este é desenvolvido eficientemente para que, então, seja possível mapear quais são as oportunidades de desenvolvimento que melhor se aplicam ao contexto brasileiro.

### 2.1. Tipologia de Segmentos e Feições do Mercado Brasileiro

Roselino (2006) se utiliza da classificação CNAE – fonte CONCLA - de serviços de informática, porém realiza ajustes para caracterizar apenas a indústria de software propondo uma nova tipologia buscando entender os determinantes da dinâmica em cada um dos segmentos:

Figura 5 – Segmentação da Indústria Brasileira de Software

Apresentação das Categorias das Empresas definidas a partir da Tipologia Proposta segundo a Fonte de Receita Predominante

<b>Categoria</b>	<b>Descrição das Atividades Principais das Empresas:</b>
<b>Categoria 1:</b> <i>Serviços de Informática</i>	Consultoria em Hardware (configurações e redes), serviços de manutenção e reparação e outras atividades relacionadas à Informática, inclusive comercialização de equipamentos.
<b>Categoria 2:</b> <i>Serviços em Software (Baixo Valor Agregado)</i>	Serviços ligados à Internet (exceto provedores de acesso), criação e manutenção de bancos de dados, processamento de dados para terceiros, suporte e terceirização.
<b>Categoria 3:</b> <i>Serviços em Software (Alto Valor Agregado)</i>	Desenvolvimento de software sob encomenda (análise, projeto, programação, testes, implantação e documentação) e desenvolvimento de projetos e modelagens de banco de dados.
<b>Categoria 4:</b> <i>Desenvolvimento e Comercialização de Software Produto</i>	Desenvolvimento e produção de software pronto para uso (inclusive customização), comercialização, licenciamento e locação de software pronto para uso (inclusive de terceiros).

Fonte: Roselino/ (2006) p. 147

As categorias de 2 a 4 são tidas como aquelas que compõem a “Indústria Brasileira de Software”. Ademais, procurou-se realizar uma separação entre a

origem do capital em “nacionais”, “estrangeiras”, “públicas” e “privadas”. Partindo deste ponto, torna-se possível dividir o setor no tocante à participação entre empresas nacionais e privadas. A partir deste ponto, o autor segue a dividir as empresas quanto a número de empresas, geração de receita líquida total e pessoal ocupado. Começando pelas empresas nacionais privadas, temos a seguinte divisão:

**Figura 6 – Segmentação da Indústria de Software – Empresas Nacionais**

**Caracterização das Empresas Nacionais Privadas de Software – Valores em R\$ milhões (2002)**

	Serviços em Software (baixo valor)	Serviços em Software (alto valor)	Software Produto	Total
<b>Número de Empresas</b>	368	140	149	<b>657</b>
<b>%</b>	56%	21%	23%	<b>100%</b>
<b>Receita Líquida Total</b>	3.800	968	894	<b>5.663</b>
<b>%</b>	67%	17%	16%	<b>100%</b>
<b>Pessoal Ocupado</b>	60.082	9.437	8.274	<b>77.793</b>
<b>%</b>	77%	12%	11%	<b>100%</b>

*Fonte: Roselino/ (2006) p. 150*

A predominância de serviços em software de baixo valor pode se explicar pelo fato de esta atividade ser mais intensiva em mão-de-obra e por haver menos barreiras à entrada relativas à escala do que nas outras atividades. As atividades desenvolvidas são de baixa complexidade em termos tecnológicos, de modo que há poucas ou até mesmo nenhuma imposição de padrões tecnológicos. Ademais, a dispersão geográfica ocorre por dois motivos. Primeiramente, devido às necessidades de mão-de-obra que pode acarretar em custos crescentes e, em segundo lugar, devido à importância das relações de comprador/desenvolvedor que favorece a atuação local destas empresas.

A predominância no número de empresas (56%) ocorre no setor de baixo valor enquanto as participações nos setores de alto valor e software produto são aproximadamente iguais. Percebe-se também que as empresas destes dois setores têm um quociente de “receita líquida/pessoal ocupado” muito superior ao setor de baixo valor, indicando assim, uma eficiência mais elevada por pessoa ocupada na geração da renda. Com isso percebe-se a heterogeneidade da indústria nacional. Ou seja, há um número considerável de empresas nacionais em todos os setores que são diferentes entre si, mas também há uma grande dependência de empresas estrangeiras. Existem possibilidades de expansão para as empresas nacionais destes últimos dois setores, porém estas estão expostas a uma grande pressão

concorrencial, conforme observaremos a geração de receita líquida das empresas estrangeiras que atuam no mercado brasileiro.

Para o caso das empresas estrangeiras de software temos uma quantidade significativamente reduzida de empresas gerando uma receita proporcionalmente superior àquelas nacionais.

**Figura 7 – Segmentação da Indústria de Software – Empresas Estrangeiras**  
**Caracterização das Empresas Estrangeiras de Software no Mercado Brasileiro – Valores em R\$ milhões (2002)**

	Serviços em Software (baixo valor)	Serviços em Software (alto valor)	Software Produto	Total
<b>Número de Empresas</b>	47	11	29	<b>76</b>
<b>%</b>	47%	14%	38%	<b>100%</b>
<b>Receita Líquida Total</b>	1.349	951	1.161	<b>3.461</b>
<b>%</b>	39%	27%	34%	<b>100%</b>
<b>Pessoal Ocupado</b>	6.235	2.847	2.798	<b>11.880</b>
<b>%</b>	52%	24%	24%	<b>100%</b>

Primeiramente, percebe-se que estas empresas têm uma produtividade – dada pela relação “receita líquida total/pessoal ocupado” – mais elevada do que as empresas privadas nacionais em todos os três setores. Em segundo lugar, conforme argumenta o autor, o setor que mais ocupa pessoas é o de serviços em software de alto valor. Sobre o setor de software produto, pode-se perceber que:

“a receita operacional líquida obtida pelas empresas de software produto para cada pessoa ocupada era quase o dobro do valor correspondente para as empresas voltadas a serviços de baixo valor. Parte dessa maior produtividade das empresas estrangeiras se explica certamente pela comercialização no mercado brasileiro de produtos desenvolvidos fora do país, que passariam apenas por processos de tradução e/ou customização para as características da demanda local” (Roselino, 2006, P. 154).

Este fato aponta para elevadas barreiras à entrada proveniente de ganhos de escala obtidos pela empresas estrangeiras neste setor.

Analisando a divisão entre empresas nacionais e estrangeiras do ponto de vista da indústria como um todo, temos que no segmento de serviços em software de baixo valor agregado, este é aquele que tem o maior número de pessoal empregado, maior parcela do total de receita líquida além de ser dominado por empresas de capital nacional. Nesse segmento, o autor coloca as empresas

públicas nacionais, que em sua maioria prestam serviços de manutenção de bancos de dados para o governo. Sobre este setor, Salatti aponta que:

“um fator importante para análise é que o custo só é considerado variável-chave para as empresas com foco em serviço de baixo valor. Para as outras, isso deixa de ser fundamental e outras variáveis, como número de clientes e quotas de mercado ganham importância” (Salatti, 2004, p.32).

Conforme exposto na tabela acima, no setor de serviços em software de alto valor agregado, tem-se uma participação equilibrada entre empresas nacionais e privadas no tocante a geração de receita líquida. A grande diferença, contudo, está no fato de que há muito mais empresas nacionais (140 contra 11) com pouco pessoal empregado. Com isto, configura-se uma realidade bem diversa entre empresas nacionais e estrangeiras no que se refere à eficiência.

Há de se atentar, portanto, à similaridade da teoria com a prática no caso deste setor no Brasil no tocante à dinâmica concorrencial. O autor expõe o fato de que a maioria destas empresas estrangeiras é ligada a marcas globais de modo a replicar sua atuação no Brasil. Com isto, podem auferir ganhos de escala ao beneficiar-se das experiências da empresa como um todo em seus projetos passados e estrutura organizacional, que impulsionam sua eficiência.

Logo, pode-se inferir que: “nesse segmento já se identifica o recurso crescente de práticas voltadas à captura de ganhos de escala, como a aplicação das técnicas de componetização e crescente reuso de módulos” (Roselino, 2006, P. 159).

Já no segmento de software produto, trata-se aquele em que operam com maior intensidade as tendências de concentração de mercado. Conforme explicitado no capítulo 1, sabe-se que os custos de desenvolvimento deste tipo de produto são muito altos e, portanto, os ganhos de escala são de extrema importância, ainda mais do que no caso do mercado de serviços em software de alto valor agregado. Ademais, as externalidades de rede atuam nesse mesmo sentido, fortalecendo as barreiras à entrada de modo que a transição entre trajetórias tecnológicas é dificultada e a posição de empresas dominantes torna-se consolidada globalmente.

Com isso, não só no mercado brasileiro, mas em diversos outros, a participação de empresas nacionais não tradicionais é dificultada e a maior parte da

geração de receita líquida do segmento provém de poucas empresas estrangeiras, relativamente ao número de empresas nacionais. Essas são as características que:

“definem o “modelo de negócio” do software nos segmentos identificados, como o papel desempenhado pelos ganhos crescentes de escala nos segmentos de serviços de alto valor, e sua maior expressão no segmento de software produto” (Roselino, 2006, P. 163).

O autor argumenta que as empresas nacionais atuantes neste segmento, geralmente procuram nichos específicos ainda não explorados por concorrentes. Apesar de encontrarem uma base diversa de consumidores, esta é fragmentada, dispersa entre regiões e tem tamanho muito reduzido se comparada à presença global de produtos de empresas estrangeiras. Por isso, não se verificam grandes ganhos de escala nestes nichos, que são tão necessários para as atividades deste setor. Então,

“num segmento em que as vantagens relativas à diluição dos custos de desenvolvimento em uma grande base de clientes têm papel crítico, este quadro parece pesar como um fator de desvantagem para as empresas nacionais” (Roselino, 2006, P. 163).

Quanto ao desempenho exportador da indústria brasileira de software, o estudo de Roselino (2006) aponta para dificuldade da mensuração do montante de exportações em software. Dificuldades estas advindas principalmente do caráter imaterial do software, que pode ser adquirido via download ou uma simples mídia de modo que fica dificultoso qualquer tipo de controle preciso sobre compras e vendas ao exterior.

Historicamente, o volume exportado brasileiro corresponde a aproximadamente 3% da receita gerada (ABES). Em 2002, dentre os setores que mais se destacam “*são as empresas estrangeiras voltadas ao desenvolvimento de serviços de alto valor agregado (com 50%), e estrangeiras voltadas a software produto (com 41,6%) as responsáveis pela quase totalidade da receita externa aferida*” (Roselino, 2006, P.175). Mesmo sendo um dos setores com o maior número de pessoal empregado e maior número de empresas, o setor de serviços em software de baixo valor agregado tem pouquíssima participação no montante total, primeiramente devido à própria natureza do serviço, a qual se concentra em regiões delimitadas.

Trata-se, portanto de uma indústria voltada para o mercado interno. Dadas as características da dinâmica concorrencial em torno do software, é possível entender as dificuldades que empresas encontram em internacionalizar-se e, com isto, a decisão para tal estratégia pode não receber grande atenção por parte das empresas. Ademais, as demandas do mercado interno Brasil, incentivam muito mais o desenvolvimento voltado para dentro. Logo, torna-se prioritário incentivar, em paralelo à observação do advento das exportações, formas de reduzir a acentuada dependência de importações visando um aumento da eficiência e competitividade.

## **2.2. Desenvolvimento Socialmente Eficiente de Software Livre**

Uma vez compreendidas as feições da indústria brasileira de software pode-se proceder a entender como determinar as situações onde é mais eficiente o desenvolvimento de software baseado em código aberto.

De acordo com Bessen (2005), produtos de software têm diversos recursos (funcionalidades) e o número de combinações formado pelas interações entre estes pode ser astronômico. Um bom software é aquele que consegue harmonizar um grupo de recursos pré escolhidos dentro de uma banda e seu sucesso dependerá de esta banda satisfazer o máximo de consumidores possíveis. Dadas as demandas por recursos, existem os pacotes de software (pre-packaged software, a exemplo do Pacote Office da Microsoft) e API's, (Application Programming Interface) que permitem aos usuários acessarem características menos evidentes destes pacotes como um tipo de "expansão" do pacote abarcando novos recursos. Assim, o modelo contempla basicamente o setor de software produto. Porém, ao longo da discussão, pode-se perceber a motivação pela qual são contratados serviços em software de baixo e alto valor agregado.

Ademais, Bessen (2005) chama atenção para a complexidade do processo produtivo de um software que o diferencia de outras *commodities*. Cada um destes recursos que interagem com outros deve ser depurado individualmente e ter seu funcionamento harmônico com os demais assegurado. Este processo pode gerar incontáveis combinações. Logo, a complexidade tem três conseqüências:

- Afeta como o pacote de aplicativos é desenvolvido
- Limita o número de recursos disponíveis

- Aumenta o custo de desenvolver contratos para aplicações customizadas

A complexidade de um software faz com que maior parte dos custos do desenvolvimento de um software venha de depuração e testes em detrimento do próprio desenvolvimento (Cusumano1991). Logo, quanto mais recursos um software se propõe ter, maior o seu custo de desenvolvimento. Do outro lado, há a questão do sucesso do software estar relacionado ao atendimento do maior número possível de necessidades.

Bessen (2005) desenvolve um modelo baseado no instrumental de equilíbrio de Nash para estudar como se dão as decisões quanto a comprar software proprietário de empresas especializadas ou desenvolver por conta própria um e também determinar em que situação cada uma destas é mais ou menos eficiente. O autor se baseia no artigo de Aghion e Tirole(1994), o qual supõe que:

- Os desenvolvedores de software de uma empresa têm a mesma capacidade de desenvolvimento que os seus clientes;
- Empresa e cliente realizam esforços para desenvolver determinada aplicação ( $e$  e  $E$ , respectivamente) além de estabelecer que o custo marginal de produção é zero;
- Empresas desenvolvedoras de software estabelecem monopólios em cada tipo de aplicação que desenvolvem com pioneirismo;
- A maior parte do custo de desenvolvimento de SLCA provém de testes e remoção de defeitos com a interação de recursos.

Quanto mais esforços empreendidos, maior a probabilidade  $p$  de sucesso de modo que  $p = (e + E)$ . O sucesso de um software depende de sua capacidade de satisfazer a demanda por determinado recursos de TIC (Tecnologias de Informação e Comunicação) denominada  $m$ , de um total de  $M$  recursos disponíveis. O cliente, paga uma taxa de licença  $y$  para a empresa que desenvolve seu produto, porém pode revender este produto para outros interessados posteriormente e receber um rendimento  $V$ , que é dividido com o desenvolvedor em uma proporção previamente acordada. Esta proporção determina quanto de esforço cada parte coloca no desenvolvimento da aplicação. Logo, os agentes tomam a decisão do quanto

pretendem investir no desenvolvimento do software e a parcela que desejam para si mesmos a fim de maximizar a utilidade que terão caso este obtenha sucesso. Este processo se dá em três etapas:

- Os dois lados negociam sobre possíveis taxas de licença e direitos autorais.;
- Os dois lados investem  $eE$ ;
- Se o software obtém sucesso, renegociam as taxas de licença e direitos autorais.

Seja  $y$  a taxa de licença; o desenvolvedor maximiza utilidade em  $p(e+E)y-e$ . O cliente maximiza sua utilidade em  $p(e+E)(V-y)-E$

Todo este processo é socialmente ineficiente, uma vez que a determinação da alocação de fatores só pode ser determinada com o conhecimento do código fonte de um software, que é ao mesmo tempo sua “planta” e o produto final acarretando em altos custos. Logo, uma empresa desenvolvedora de software pode tomar a iniciativa e desenvolver um pacote de software. Neste modelo, a empresa sabe *ex ante* o número  $N$  de clientes que desejam usar um produto que contém  $m$  recursos, de um total de  $M$  possíveis. Com base nas três premissas do modelo e nas relações acima estabelecidas, Bessen (2005) conclui que só existem duas situações em que os clientes não compram o pacote: o preço é muito alto ( $V_i < w$ ) ou por que o software produto é muito simples para as necessidades de clientes ( $M^* < m$ ). Se ele compra o pacote, é porque recebe maior utilidade do que se o desenvolvesse por conta própria, dado que o código fonte de um software bem sucedido pertence a uma empresa e estaria indisponível. Porém, quando um SLCA é estabelecido com sucesso, o núcleo de seu código está disponível para clientes que dele desejem utilizar-se, sem custo adicional. Supondo que um SLCA tenha estabelecido código para a mesma quantidade  $m_{\sim}$  de recursos que o pacote de aplicativos ( $m_{\sim} = m$ ), então, a adoção de softwares mais complexos desenvolvidos em código livre ( $m_{\sim} > m$ ) terá maior eficiência social do que o pacote de aplicativos acrescidos com APIs dado que o custo de desenvolvimento deste torna-se mais baixo para iniciativas de código aberto.

Logo, para clientes com necessidades mais complexas e maior capacidade própria de desenvolvimento, as aplicações de SLCA fazem aumentar o alcance do mercado. A motivação para clientes participarem de SLCA é basicamente a possibilidade de diminuição de custos e dependência de fornecedores. Para empresas desenvolvedoras, torna-se uma alternativa viável incentivar a pesquisa de SLCA pela possibilidade de aumentar o número de programas e soluções com que trabalham e, conseqüentemente, o volume de serviços prestados em torno destes, uma vez que não só a capacidade de desenvolvimento é importante para que este tipo de solução seja viável. Implementação, adequação de equipamentos, treinamento e suporte entre outros constituem esta gama que gira em torno destas aplicações.

### **2.3. Influência de Software Livre nos Segmentos de Mercado**

Depois de estudadas as características do mercado de software e serviços de TI do ponto de vista teórico e do prático no caso brasileiro além de estabelecidos critérios para mapear as situações de maior eficiência, estamos habilitados para analisar as influências de SLCA em cada um dos segmentos do mercado.

A análise deste passará pela tipologia estabelecida por Roselino (2006) – explicitada na primeira seção deste capítulo - e será feita com base em uma pesquisa que visou entender as ações de diversas empresas brasileiras que adotam este modelo e, também, aquelas que demandam este tipo de aplicação.

Conforme assinalado anteriormente, para adotar uma solução em software, além dos custos, há a questão de interoperabilidade que assinala para a capacidade do software lidar com os padrões de arquivo estabelecido (minorando custos de conversão e aprendizado para lidar com outros padrões). Portanto, em cada segmento, as empresas têm maior ou menor capacidade de impor ou lidar com os padrões além de flexibilidade e custos com software, de modo que têm percepções e interesses diferentes sobre as mesmas aplicações baseadas em SLCA.

A adoção de SLCA afeta cada segmento de maneira diferente, como veremos adiante. Cada um destes segmentos será analisado do ponto de vista da estratégia da adoção de SLCA, de como a empresa interage com SLCA e dos efeitos potenciais no mercado.

## **Serviços em software de baixo valor agregado**

Este é o segmento do mercado com o maior número de empresas, maior geração de receita líquida total e de pessoal ocupado, porém o menos eficiente do ponto de vista de geração de receita líquida por pessoal ocupado. Ademais, tem potencial de imposição de padrões ou preços nula ou muito limitada, é pouco concentrado (várias pequenas empresas) e esparso geograficamente.

O tipo de empresa que se encaixa neste segmento geralmente realiza a conjunção de software e hardware para as empresas locais. Como ela e seus clientes apenas adquirem software e hardware e não desenvolvem nenhum destes internamente, os custos com estes fatores são dados exogenamente, conforme explicitado no primeiro capítulo no modelo de Riehle (2007). O fator sobre o qual têm maior controle é o custo da mão-de-obra. Dado que a capacitação média necessária neste segmento é baixa e o número de funcionários por empresa é reduzido, há pouco ou nenhum desenvolvimento de software dentro destas empresas. Os serviços prestados geralmente concentram-se na fase de implantação de sistemas para empresas, tais como gestão de bancos de dados e programas de gestão pouco complexos, mudança de hardware e outros.

Neste contexto, a adoção de software livre afetaria diretamente a estrutura de custos das empresas. Porém, há outra característica a ser considerada: a solução precisa ser bem difundida. Ou seja, como há pouco desenvolvimento de software dentro de empresas, a liberdade de uso do código de fonte é de importância secundária. O que interessa às empresas é que esta (solução) seja capaz de lidar com os padrões mais difundidos e não imponha grandes dificuldades de adaptação para seus clientes, uma vez que tal ação poderia ser custoso e, finalmente, desinteressante. Logo, a solução deve atuar de maneira complementar a outros softwares proprietários mais bem aceitos.

Com isto, tem-se que uma diminuição nos custos com software pode aumentar a eficiência de uma empresa prestadora de serviços na geração de receita líquida ou ainda possibilitar a expansão das atividades. Apesar de os ganhos de escala não serem de grande importância neste setor, a notoriedade da empresa e

sua reputação podem crescer no âmbito local gerando ganhos adicionais e cumulativos para as empresas.

Aquelas que conseguirem uma maior gama de clientes, baseadas na flexibilidade de preços tornar-se-iam mais competitivas de modo a promover um acirramento do mercado local para estes serviços. Com isto, outras concorrentes ver-se-iam forçadas a adotar novas estratégias.

A empresa é passiva quanto aos rumos de desenvolvimento do software, uma vez que geralmente não têm capacidade de lidar com o software a este nível. Em um primeiro momento, estabelece uma estratégia predominantemente parasitária. Porém, estas desempenham um importante papel, que é o de difundir as aplicações a um nível regional e lidarem diretamente com as estas quanto à implementação e suporte. Assim, possuem conhecimentos tácitos acerca dos pontos positivos e dos pontos de melhora de cada aplicação das quais fazem uso de modo a estabelecer potencial de interação com as comunidades em um nível de feedback com base em suas experiências. Podem apontar *bugs*, pontos prioritários de trabalho e demandas adicionais.

Este segmento, apesar de ser o mais fragmentado e menos eficiente, é aquele com maior volume em termos de geração de empregos e receita líquida sendo, assim, de grande importância para o mercado como um todo. Ademais, a diminuição com estes custos de software pode significar uma diminuição de renda enviada para o exterior em forma de royalties e licenças. Provavelmente, o maior obstáculo enfrentado é o da informação tanto no sentido de empresas descobrirem que existem estes tipos de solução quanto para os clientes perceberem oportunidades de redução de custos e de expansão de investimentos em soluções baseadas em TICs (Tecnologias de Informação e Comunicação).

### **Serviços em software de alto valor agregado**

Este segmento dá cabo de atividades mais complexas que o anterior e possui uma configuração de mercado abrangente. Trata-se de um mercado concentrado: poucas grandes empresas com muitos funcionários, o que evidencia a importância dos ganhos de escala neste segmento. Quanto maior o portfólio de cada empresa, maior a acumulação de conhecimento que pode ser reusado para projetos futuros

além de maior credibilidade junto ao mercado que percebe determinada empresa como capaz de desenvolver soluções que levem em conta a especificidade de seus processos. Por fim, um último ponto importante é que nem sempre há padrões tecnológicos para a elaboração de um software sob encomenda. Com isso, uma empresa que desenvolve com sucesso uma aplicação para um cliente de determinado setor, pode conquistar um nicho de mercado composto por outras empresas daquele setor de maneira similar a uma empresa de software produto.

A atividade dessas empresas começa desde a fase de design do software, passando pela implementação, até os serviços de suporte prestados, fase esta que ganha importância quando da adoção do SLCA.

Um exemplo de adoção de SLCA que impulsiona este setor pode vir do lado da demanda de empresas por implementação de soluções em software livre visando economia de recursos com licenças e royalties, tais como sistema operacional Linux, LibreOffice, Moodle, etc. Além da economia de recursos, em prazos de tempo maiores, empresas podem buscar desenvolver suas próprias aplicações e, quebrar relações de *lock-in* com fornecedores, baseando-se em SLCA. Exemplos nacionais de grandes demandantes corporativos de software livre são empresas públicas do setor financeiro como o Banco do Brasil – que já adotou a suíte de aplicativos BrOffice e tem como meta implantar o sistema operacional Linux em todos os seus terminais de auto atendimento, quebrando a dependência do sistema operacional da IBM que era utilizado até então – a Caixa Econômica Federal – cujo sistema de loterias é baseado em código aberto – e os órgãos governamentais. A ação destes segue em consonância com a política governamental para SLCA que será pormenorizada no capítulo 3. Então, o aspecto de serviços e suporte deste segmento passa a ganhar importância na fase de implementação e o aspecto de design de software ganha importância no desenvolvimento e aprimoramento da solução adotada.

Vale lembrar que comunidades de SLCA geralmente trabalham com arquitetura modular. Um módulo desempenha uma determinada função independentemente de outras, assim, estas comunidades são também espécies de bibliotecas que contam com um acervo diverso de módulos disponíveis livremente. Uma empresa pode se beneficiar destas bibliotecas ao utilizar módulos já

desenvolvidos previamente para suas atividades, conforme o modelo de inovação privado-coletivo explicitado no capítulo 1, a empresa ganharia acesso também aos esforços desenvolvidos pelos membros das comunidades. Com isso, poderiam reutilizar conhecimento de modo a acelerar os ganhos de escala que seriam obtidos apenas depois de determinado período de serviços prestados. Ademais, a existência destes acervos de módulos poderia também dirimir barreiras à entrada de novas empresas, uma vez que o acréscimo no número de participantes pode acelerar a submissão de módulos e aumentar o acervo disponível e assim por diante. Com efeito, o mercado tornar-se-ia mais competitivo e mais abrangente, favorecendo, também, a inovação no desenvolvimento de software.

A necessidade por vários e diversos módulos levanta a importância das empresas manterem algum tipo de relação com comunidades de desenvolvedores por duas razões correlacionadas entre si. A primeira é a necessidade de direcionar esforços para os tipos de atividades desejadas. Pode ser que nem sempre um módulo esteja disponível, portanto, torna-se importante incentivar o desenvolvimento de alguma área do tipo. Em segundo lugar, é necessário acumular certa credibilidade com uma comunidade, de modo que esta não se veja lesada pela atividade parasitária de terceiros dentro da comunidade, o que poderia desmotivar programadores a fazerem parte da comunidade ou, ainda, que mudem a configuração de licenças inviabilizando algum tipo de ação promovida pela empresa em questão. Conforme Varian e Shapiro (2007) explicam, há também ganhos de credibilidade e capacitação técnica dos funcionários ao se associarem às comunidades:

“Distributors and systems integrators who are active participants in the opensource community offer their customers expertise and a demonstrated commitment to keeping their knowledge at the cutting edge of software development—a compelling credential that helps them attract business. In many ways, these open source firms’ business model recasts software as a service industry, rather than as a products industry (Varian & Shapiro, 2003, p.7)”.

Por conta destas razões, a empresa deve pensar cuidadosamente no tipo de estratégia que adotará para cultivar o melhor destas comunidades. O modelo de estratégia comensalista parece o mais adequado, pois, disponibilizando conteúdo e resultados de seus trabalhos, uma empresa consegue a credibilidade e o direcionamento para certas áreas de interesse. Uma estratégia totalmente simbiótica

seria de pouco retorno devido à diversidade de serviços prestados por este tipo de empresa, ainda sim, dependendo do tamanho da empresa, pode ser interessante formar comunidades em torno de uma gama de aplicações de modo a promover sua adesão e, assim, fornecer serviços diversos em torno dessas. Este tipo de estratégia envolve um misto entre atividades deste segmento e do de software produto, como o caso da estratégia da IBM, que será explicada ao tratarmos da adoção de SLCA neste último.

### **Software Produto**

Neste segmento, verificam-se com maior intensidade as tendências de concentração de mercado comparativamente com os outros dois segmentos. Dado que os custos marginais de reprodução são quase nulos e o processo produtivo do software concentra-se quase inteiramente na fase de design, a fatia de um mercado dominada por uma empresa é de importância central para o seu desempenho. Situação essa reforçada pela influência de efeitos de rede. Com isso, o espaço de atuação de empresas deste segmento é bastante restrito e há necessidade de acionar novas demandas e encontrar novos mercados.

Por mais que a produção de software livre gere efeitos de rede e economias de escala - que são inerentes de seu processo produtivo - a aceitação de produtos novos é dificultada, dada a atuação global de players deste segmento. Atuação esta que gera robustas barreiras à entrada. De fato, aplicações em SLCA que concorrem com softwares proprietários não são facilmente aceitas, mais ainda é viabilizar soluções comerciais em torno desse modelo. Eventualmente, alguns programas concorrentes ganham espaço – como os casos do Linux e OpenOffice (LibreOffice) – ganham notoriedade, especialmente pelos custos nulos de uso, porém, em um dilatado espaço de tempo.

Porém, não se trata apenas de desenvolver funcionalidades adicionais a SLCA e comercializá-las. Arranjos de licenças podem não permitir tal ação, além da possibilidade de a diferença de custo para um software proprietário já estabelecido não seja suficientemente expressiva de modo que as estratégias de empresas que operassem nesses moldes poderiam não ser atrativas ou lucrativas o suficiente. Uma importante característica do SLCA que permite aplicações comerciais está na

própria motivação para que usuários se integrem às comunidades. A necessidade por aplicações ainda não existentes e o livre fluxo de informações são essenciais para desenvolvimento de soluções inovadoras. Linguagens estatísticas (como R), de programação (como Java), formatos de arquivos (como o codec de áudio FLAC) entre outras iniciativas visam justamente criar novas funcionalidades ou alternativas superiores ou gratuitas. Ao estudar a adoção de Linux por governos, Varian e Shapiro (2007) assinalam que os retornos para empresas giram em torno da integração de software produto e serviços, como se procurassem abarcar os três segmentos do mercado:

“Returns can come in various forms, including internal use of the resulting software. But the biggest commercial attraction is the sale or licensing of complementary products or services: the prospect of earning revenues from applications software, hardware, and services can make investment in Linux” (Varian & Shapiro, 2003, p.6).

Assim, empresas que conseguem traduzir estas invenções em inovações podem explorar novos segmentos de mercado. Quanto melhor procederem em suas relações com as comunidades desenvolvedoras, melhor podem explorar vantagens competitivas e aprimorar suas soluções continuamente. Estratégias simbióticas desse tipo foram e ainda são utilizadas por empresas como a antiga MySQL, que hoje pertence à Oracle. Outro exemplo, desta mesma empresa, situa-se no campo de interoperabilidade. A empresa SAP procurou antecipar-se a novas tendências e padrões levando em conta a crescente importância de SLCA e a interoperabilidade de suas aplicações com estes padrões, criando comunidades especificamente voltadas para estas questões, conforme anuncia em seu *website*:

“Standards and open source community ensure that SAP's platform is open and interoperable with other vendor solutions in heterogeneous IT environments (...) Today it is hard to predict what kinds of systems will form a company's business network in the future. The best way to hedge against this challenge is through industry standards. Working closely with companies in the SAP ecosystem, SAP leads initiatives in business semantic standards (UN/CEFACT), the banking industry, Web services interoperability, and open source support (OSS)” (SAP COMMUNITY NETWORK, referência online).

Ademais, há de se levar em conta que o desenvolvimento de novas aplicações necessita dos serviços necessários à sua implementação, (tanto de baixo como alto valor agregado) para demandantes corporativos e governos de modo que os três segmentos devem operar em consonância. Com efeito, empresas podem

criar SLCA e vender serviços relacionados a estes. Exemplo deste tipo de estratégia, também simbiótica, é aquela desenvolvida pela IBM. Durante as últimas décadas, a empresa hospeda mais de 120 projetos, abriu código de aplicações chave e investe pesadamente em outros. Assim, a empresa mantém comunidades ativas e desenvolveu um portfólio de soluções disponíveis para diversos públicos a preços baixos e até nulos alavancando a presença da companhia em diversos segmentos relacionados a software, em especial a prestação de serviços.

De diversas maneiras, este segmento pode utilizar de SLCA. Porém, este depende de outros segmentos para que possa lograr crescimento em aceitação e conhecimento de seus produtos. O desafio está em conquistar parcelas de mercado muito fragmentadas e vencer a concorrência com grandes empresas estrangeiras que já percebem e elaboram estratégias baseadas em SLCA.

Ao fim desta exposição sobre a influência de SLCA nos três segmentos de mercado que compõem a indústria, pôde-se perceber a variedade de estratégias que são utilizadas por empresas de cada um dos três segmentos e as diferentes variáveis que exercem mais ou menos influência em cada um destes seja direta ou indiretamente. Percebe-se que os pressupostos contidos na teoria de Bessen (2005) sobre desenvolvimento socialmente eficiente de SLCA são verificados no mercado brasileiro e, ainda, são impulsionadas pelas suas especificidades. Notadamente a diversidade de setores econômicos em expansão que demandam soluções complexas não satisfeitas por aplicações horizontais (software produto) ou menos custosas e conseqüentemente, maiores demandas por serviços de TI, onde se situam as maiores possibilidades de expansão para o modelo de desenvolvimento de SLCA. Esta mudança de foco também encontra correspondência nas estratégias de grandes empresas internacionais e políticas governamentais, como aquelas que são desenvolvidas pelo governo brasileiro, objeto de estudo do próximo capítulo.

### **3. Intervenção Pública Para Software Livre/Código Aberto**

Recentemente, governos de diversos países como Brasil, Estados Unidos, Itália, França, Alemanha entre outros vêm não somente adotando aplicações de SLCA em seus departamentos como também realizando políticas em cima do assunto, fornecendo apoio direto e indireto na forma de desenvolvimento, consultoria e até incentivos fiscais.

A despeito do mercado de software como um todo, o governo não só o regula do ponto de vista normativo como também é grande comprador, dadas a abrangência e escala de suas atividades. Este duplo papel exercido, conforme assinalado por Comino e Manetti (2010), é de crucial importância na determinação dos rumos dos mercados de software de cada país. O governo pode ou não intervir no mercado de acordo com sua percepção sobre a existência e intensidade das falhas de mercado.

Além deste motivo, os próprios méritos do software, relacionados aos custos incorridos de sua utilização e direitos de propriedade são levados em conta de modo que o governo busca a melhor solução tanto no curto quanto no médio e longo prazo. São estas as duas razões que, levadas conjuntamente em consideração, determinam os rumos da política governamental acerca de SLCA.

Este capítulo procura sistematizar a argumentação em torno da intervenção estatal tanto em termos favoráveis quanto desfavoráveis a esta atuação. Assim, procura-se entender como se dá a decisão de intervir ou não no mercado de software. Necessário para tal decisão, é entender a visão do governo acerca deste mercado ao longo do tempo e as iniciativas já realizadas que configuraram a existência de órgãos, agências e empresas que atuam e auxiliaram a modelar as feições do mercado. A seguir, serão explanadas as ações do governo neste assunto e discutidos seus efeitos no mercado em cada um dos setores.

#### **3.1. Intervenção Estatal: Argumentação**

Da decisão do governo de atuar ou não no mercado de software utilizando-se de políticas voltadas ao desenvolvimento de SLCA, entram em cena diversos pontos de vista tanto a favor quanto contra tal ação.

## **Argumentos favoráveis à intervenção**

De acordo com os autores Comino e Manetti (2010), a argumentação favorável à intervenção estatal em favor de SLCA baseia-se em cinco argumentos que levam em conta como o governo lida com estas aplicações e como pode se beneficiar destas.

### **1. SLCA é visto como um produto tecnicamente superior**

Este primeiro argumento leva em conta que uma comunidade ativa de desenvolvedores consegue melhorar continuamente um software pela resolução de *bugs* e por incrementos no código fonte. Exemplifica uma ação governamental que se encaixa neste argumento, a disponibilização do chamado portal software público (<http://www.softwarepublico.gov.br>), o qual lista e junta diversos projetos de software e mobiliza desenvolvedores em torno destes.

Ademais, há de se levar em conta que a flexibilidade de um software permitida pela disponibilidade do código fonte, possibilita mudanças de acordo com as suas próprias necessidades e favorecem interoperabilidade, reduzindo custos associados a pagamentos de licenças do diferentes softwares para diferentes formatos e, também, aqueles custos relacionados a aprendizado

### **2. Possibilita reduções de custo e reuso de código**

Este argumento possui três implicações, a saber, redução de custos com licenças; diminuição da dependência de fornecedores e reuso do código fonte.

Primeiramente, o arranjo de licenças geralmente permite uma redução de custo, sendo uma das principais razões, ou pelo menos a primeira a ser levada em conta, quando da decisão acerca de sua adoção. A segunda implicação resulta em uma redução de custos no médio e longo prazo, dado que é possível evitar situações de *lock-in* com fornecedores e garantir que o acesso aos dados ao longo do tempo seja livre ou de reduzidos custos comparativamente com software proprietário. Dado o volume de dados gerados pelo governo, pode haver expressiva redução de gastos com este ao adotar soluções SLCA.

Finalmente, uma solução adotada para um ramo do governo em particular pode ser útil para outros ramos gerando redução de custos por conta do reuso de código fonte previamente disponível uma vez que o governo possuiria a soberania sobre uma aplicação e pode direcioná-la conforme suas diretrizes.

### **3. Estimula inovação na indústria de software**

Este argumento passa pela questão de patentes. Bessen (2002) atribui mérito ao SLCA por aumentar o alcance do mercado de software, limitado por assimetrias de informação e incompletude contratual (devido à complexidade e incerteza, conforme explicitado no capítulo 2, seção 2). Para ele, a única forma de ação do governo seria a remoção da falha de mercado que o próprio governo cria, a saber, o fortalecimento da proteção a patentes de software.

Ainda sobre este tópico, Bessen e Maskin (2009) argumentam que o ritmo do progresso tecnológico em indústrias onde o crescimento tecnológico é cumulativo pode ser acelerado pela redução da força de mecanismos de *copyrights* de modo que:

“o número de inventores em potencial que têm acesso à tecnologia aumenta e, dado que seus projetos P&D são imperfeitamente correlacionados, isto estimula as inovações subsequentes” (COMINO e MANETTI, 2010, p.5).

### **4. SLCA Favorece a Competição**

Este argumento relaciona-se com o anterior, no tocante que o crescimento no número de agentes do mercado aumenta a concorrência. Comino e Manenti (2002), destacam uma falha de mercado, relacionada à ignorância dos usuários quanto à existência do SLCA e concluem que a melhor ação do governo seria dirimir estas assimetrias de informação

O governo pode, ainda, disponibilizar o código de suas aplicações ao público, proporcionando que pessoas e empresas utilizem-se destas aplicações aumentando as pressões concorrencias no mercado dado que as empresas podem expandir o número de soluções com que trabalham e estariam aptas a possuírem os mesmos conhecimentos técnicos acerca do funcionamento de cada uma delas.

## 5. Promove práticas de governo eletrônico (e-government)

Práticas de governo eletrônico são aquelas relacionadas ao uso de tecnologias de informação e comunicação (TIC) para entregar serviços governamentais. O próprio governo brasileiro mantém ações de governo eletrônico baseadas em SLCA e as define da seguinte maneira:

“O desenvolvimento de programas de Governo Eletrônico tem como princípio a utilização das modernas tecnologias de informação e comunicação (TICs) para democratizar o acesso à informação, ampliar discussões e dinamizar a prestação de serviços públicos com foco na eficiência e efetividade das funções governamentais (...). No Brasil, a política de Governo Eletrônico segue um conjunto de diretrizes que atuam em três frentes fundamentais: junto ao cidadão; na melhoria da sua própria gestão interna; e na integração com parceiros e fornecedores” (<http://www.governoeletronico.gov.br/o-gov.br>).

Esta ação também revela preocupação não só com custos enfrentados governo, mas também com progresso tecnológico e direcionamento do investimento, conforme explicitado nos “Princípios e Diretrizes Gerais Para o Governo Eletrônico”

“O software livre deve ser entendido como opção tecnológica do governo federal. Onde possível, deve ser promovida sua utilização. Para tanto, deve-se priorizar soluções, programas e serviços baseados em software livre que promovam a otimização de recursos e investimentos em tecnologia da informação. Entretanto, a opção pelo software livre não pode ser entendida somente como motivada por aspectos econômicos, mas pelas possibilidades que abre no campo da produção e circulação de conhecimento, no acesso a novas tecnologias e no estímulo ao desenvolvimento de software em ambientes colaborativos e ao desenvolvimento de software nacional (...). A escolha do software livre como opção prioritária onde cabível, encontra suporte também na preocupação em garantir ao cidadão o direito de acesso aos serviços públicos sem obrigá-lo a usar plataformas específicas” (<http://www.governoeletronico.gov.br/o-gov.br/principios>).

Assim, a percepção do governo brasileiro é que SLCA possibilita desenvolver soluções de governo eletrônico mais rapidamente e com menor custo do que se fossem baseadas em software proprietário além de evidenciar uma política que prefere adotar padrões livres. Deste modo, seria possível expandir o uso das tecnologias de informação e comunicação para diversos outros ramos ou até praticar políticas de inclusão digital com custos reduzidos. Como, por exemplo, os programas de governo eletrônico de Atendimento ao Cidadão, Telecentros Comunitários e outros, conforme consta na Mensagem do Presidente para o Guia Livre – Referência de Migração Para Software Livre do Governo Federal (GRUPO DE TRABALHO MIGRAÇÃO PARA SOFTWARE LIVRE, 2005).

## **Argumentos Desfavoráveis à Intervenção**

A argumentação contrária a atuação do governo gira em torno da interpretação do governo se realmente existem falhas de mercado e se a utilização de SLCA é realmente a melhor opção para lidar com estas além da adequabilidade da adoção destas soluções. O desafio, entretanto, é reunir observações empíricas que confirmem a argumentação nos primeiros dois pontos discutidos abaixo.

### **1. Competitividade no Mercado e Necessidade de Intervenção**

Ao observar-se a evolução de iniciativas de código aberto até os dias de hoje, há de reconhecer que diversas destas já logram sucesso e vêm ganhando cada vez mais notoriedade e espaço no mercado. Por conta disto, alguns autores como Evans (2002) percebem que o mercado de software já é altamente competitivo e não vêm grandes falhas de mercado que justificariam a atuação do Estado. Ademais, observaram que intervenções que estimulassem alternativas a software proprietário estariam escolhendo um vencedor sem levar em conta o volume e, especialmente, a produtividade dos investimentos em P&D realizados por empresas ao longo do tempo.

### **2. Custo Total de Posse e de Migração**

Os autores Varian e Shapiro (2003) chamam atenção para o fato de que não só o custo com licenças constitui o montante gasto com software. Treinamento de usuários, atualizações, manutenção e suporte entre outros também estão relacionados com gastos com software e constituem os custos totais de posse (do inglês: Total Cost of Ownership).

A variação destes fatores ao longo do tempo deve ser levada em consideração antes de determinada solução ser implementada. Ainda que estes autores argumentem que a diferença de custos entre software proprietário e SLCA pode ser afetada pelo custo do fator trabalho entre cada país, a estimativa do montante de custos cortados pela adoção de SLCA seria mais dificilmente percebida ou até não suficiente, dado que há custos de migração associados a deixar de utilizar uma solução proprietária.

Migrar de uma solução já existente para outra envolve custos com treinamento e adequação de formatos já que uma das maneiras de assegurar a fidelidade de usuários a um software é o grau de aceitação de seu padrão de formato de arquivos remontando à questão de *lock-in* de fornecedores.

Quanto maior a capacidade de um distribuidor de software proprietário em estabelecer padrões e maior sua parcela de mercado, maiores os gastos com a migração para outras soluções. Em contrapartida, SLCA com seu formato livre de arquivo pode oferecer ganhos de longo prazo ou até de curto prazo, caso seja capaz de lidar com os padrões proprietários estabelecidos.

### **3. Arranjos de Licenças**

Lessig (2002) argumenta que o arranjo de licenças comumente utilizado por governos ao redor do mundo gira em torno da licença GPL que é utilizada visando assegurar que o software seja não excludente e não rival pode, de fato, aumentar o bem estar. Porém, os termos de licenças deste tipo podem ser restritivos do ponto de vista da apropriação privada destes resultados. Isto se dá no sentido da possibilidade de o desenvolvimento de soluções complementares – de código fechado – de aplicações baseadas em código aberto. Então, o autor sugere que licenças menos restritivas sejam utilizadas, tais como BSD.

#### **3.2. O Arcabouço Institucional da Indústria Brasileira de Software**

No Brasil, desde os tempos da ditadura militar, a indústria de software já era alvo de políticas de fomento e, neste período, já foram criadas agências e políticas que seriam de vital importância na modelação das feições atuais da indústria e de vários de seus principais agentes. Em meados dos anos 60, especificamente em 1964, foi criado o SERPRO (Serviço Federal de Processamento de Dados), que ao longo dos anos foi parte de diversas políticas e que viria a tornar-se a maior empresa pública de prestação em serviços em TI da América Latina atualmente. Hoje, é um dos principais instrumentos de política de software do governo brasileiro como o Protocolo de Brasília, adoção de SLCA nos Ministérios, Governo Eletrônico entre outros, dado que:

“O Serpro, como provedor de soluções para o governo federal, tem um papel fundamental na definição e implementação de alternativas livres para o estado brasileiro. A empresa deixa de ser apenas uma receptora de

tecnologias oferecidas por fornecedores e toma posição de participante efetiva no desenvolvimento de soluções livres para o país, devendo se portar como agente principal no direcionamento de tecnologia de software livre para o governo federal” (<http://www.serpro.gov.br/tecnologia/software-livre/programa-serpro-de-software-livre-pssl>)

Durante os anos 70 foi criada a CAPRE (Coordenação de Atividades de Processamento de Dados) composto pelo então denominado BNDE, pelo SERPRO e por representantes das forças armadas. No momento histórico em questão, software livre não era a questão, mas sim a percepção por parte de diversos setores da sociedade de que a soberania nacional tecnológica em diversos setores era de fundamental importância para o desenvolvimento da nação e o software como insumo tecnológico destes.

“Na vertente militar a preocupação inicial estava relacionada à importância crescente da informática como instrumento capaz (...) de fortalecer a segurança nacional. Já a vertente civil era formada por economistas de formação Cepalina alocados em órgãos públicos como BNDE e por pesquisadores e técnicos originários de instituições científicas nacionais (com destaque para o ITA)” (DIEGUES, 2010, p. 195).

Esta primeira fase visava entender a indústria nacional de software e a mobilizar forças políticas para que então se desse início à formulação de políticas voltadas ao seu fomento. Outro importante resultado dessa ação foi a criação da COBRA (Computadores e Sistemas Brasileiros) em 1974 que foi aquela que *“centralizou sua atuação nos segmentos de minicomputadores e foi responsável pelo primeiro computador totalmente projetado, desenvolvido e industrializado no Brasil”* (DIEGUES, 2010, p. 196). Hoje, no entanto, a empresa é controlada majoritariamente pelo Banco do Brasil (99,88%) e concentra-se nas áreas de serviços em software de alto valor agregado e de software produto, baseando-se em soluções de software livre como segue:

“A Fábrica de Software da Cobra Tecnologia desenvolve soluções destinadas às instituições financeiras e às organizações da administração pública. Todas as ferramentas utilizadas são software livres e o código fonte produzido é entregue ao cliente, garantindo a continuidade do produto e, ao mesmo tempo, independência do fornecedor” ([http://www.cobra.com.br/home/images/pdf/cobra\\_fabrica.pdf](http://www.cobra.com.br/home/images/pdf/cobra_fabrica.pdf)).

Ao longo da década de 70 e início dos anos 80, observou-se que a deterioração das contas externas brasileiras determinou mudanças políticas e é neste contexto que surgem as primeiras políticas de reserva de mercado, visando incentivar a produção de Hardware. A CAPRE foi substituída pela SEI (Secretária

Especial de Informática) vinculada ao conselho de segurança nacional (DIEGUES, 2010, p. 197) e responsável pela elaboração da PNI (Política Nacional de Informática) que estabelecia proteção a diversas empresas nacionais. Período esse encerrado emblematicamente por um embate com os Estados Unidos em 1988. A legislação brasileira só permitia que softwares fossem comercializados no Brasil caso não existisse similar nacional. Com isto, o Brasil negou-se a conceder a permissão para a comercialização do aplicativo MS DOS da Microsoft afirmando que já existia o similar nacional chamado SISNE. Frente a ameaças de sanções comerciais por parte das autoridades americanas, foi concedida a permissão da entrada do produto. Apesar do desfecho desfavorável há de reconhecer os primeiros passos rumo ao crescimento do setor de informática no Brasil. (DIEGUES, 2010, p. 198 e 199)

A primeira política especificamente voltada para o segmento de software foi a criação do programa SOFTEX 2000 em 1993 durante o governo Collor que acompanhou a mudança na visão de como deveria ser o papel do Estado para diversos setores da economia, passando de uma ação intervencionista para uma ação reguladora. O programa vislumbrava o software na sua importância para os demais setores da economia como responsável por aumento na competitividade destes e via a *“possibilidade de se replicar no Brasil um modelo de desenvolvimento da indústria de software baseado nas exportações tal qual observado na Índia e na Irlanda (DIEGUES, 2010, p. 206).*

A meta inicial do programa era fazer com que as exportações brasileiras de softwares chegassem a R\$2 bilhões até o ano 2000 a partir da criação de núcleos regionais localizados em todas as regiões do país que disponibilizavam formas de auxílio a empresas locais tais como: *“disponibilização de infra-estrutura física para a localização de empresas, incubadora, auxílio gerencial, de marketing, em finanças e para a atualização tecnológica, recursos para a participação em feiras e eventos no exterior além de cursos em diversas áreas” (DIEGUES, 2010, p. 203, 204),* além de associação com núcleos universitários e instituições de pesquisa combinadamente à instituição da Lei da Informática também de 1993, a qual convertia o instrumental de reserva de mercado para incentivos fiscais a empresas de informática

Em termos de meta, o programa não logrou o objetivo a que se propôs. Como vimos no capítulo 2, o setor de software brasileiro sempre foi muito mais voltado para dentro de si, acompanhando a evolução da própria indústria brasileira do que primariamente voltado para o exterior. As próprias políticas governamentais contribuíram para esta trajetória desde as políticas de reserva de mercado até o choque de competitividade sofrido pela economia brasileira nos primeiros anos da década de 90, de modo que, conforme Roselino (2006) explicita:

“qualquer avaliação dos resultados do SOFTEX deveria considerar não apenas o impacto dessa iniciativa no incremento dos valores comercializados de software brasileiro no exterior, mas também os impactos sobre o desenvolvimento tecnológico relacionado à atividade, que resultaria em efeitos positivos para toda sorte de atividades que se relacionam com o software” (Roselino, 2006, p. 124).

O ano de 2003, marca a publicação das diretrizes da PICTE (Política Industrial, Tecnológica e de Comércio Exterior). Este conjunto de medidas resgata parcialmente a percepção de importância das políticas industriais promovidas pelo Estado, porém, centrada no aumento da competitividade das empresas brasileiras como passo necessário para aumento da participação internacional das empresas nacionais, e, por fim, do desenvolvimento industrial.

Neste contexto, o setor de software foi visto como estratégico juntamente de outros setores como o de fármacos e bens de capital. Esta política ainda possui a meta de elevar o montante de exportações brasileiras de software, desta vez, fixou-se a meta de R\$ 4 bilhões. O diferencial desta política, contudo, está em fixar como meio para o fim um aumento substancial do tamanho e da participação de empresas nacionais no mercado interno, porém ainda mantém um olhar para o desempenho exportador pouco condizente com a natureza do mercado nacional. Com isso, *“Tal orientação, por sua vez, decorre da percepção expressa em suas diretrizes de que há uma correlação positiva entre o porte das empresas e o desempenho exportador e inovativo”* (DIEGUES, 2010, p. 210). Passagem esta que explicita as assimetrias entre as empresas nacionais e as estrangeiras que aqui operam.

### **3.3. Análise da Ação Governamental**

É no contexto acima explicitado que ocorreram as primeiras ações de intervenção governamental visando fomentar o desenvolvimento de SLCA. Estas datam do decreto assinado pelo presidente Luiz Inácio Lula da Silva durante o ano

de 2003 instituindo dois comitês técnicos do governo eletrônico: Implementação de Software Livre, coordenado pelo ITI (Instituto Nacional de Tecnologia da Informação) e Sistemas Legados e Licenças de Software e, a seguir, viu-se a diretoria da Casa Civil que aconselhou o estudo da viabilidade da adoção de SLCA pelos ministérios. São estas as bases que norteiam o programa de adoção de software livre promovido pelo governo brasileiro. Com isso, foram estruturadas:

“as estratégias de intervenção, adotadas como orientações para todas as ações de Governo Eletrônico, gestão do conhecimento e gestão da TI no governo federal: Promoção da cidadania como prioridade; Indissociabilidade entre inclusão digital e o governo eletrônico; Utilização do software livre como recurso estratégico; Gestão do Conhecimento como instrumento estratégico de articulação e gestão das políticas públicas; Racionalização dos recursos; Adoção de políticas, normas e padrões comuns; Integração com outros níveis de governo e com os demais poderes.” (Oficinas de Planejamento Estratégico. RELATÓRIO CONSOLIDADO. Comitê Executivo do Governo Eletrônico. Maio de 2004. pág 8).

Desde então SERPRO e COBRA tornaram-se as principais empresas que lidam com este tipo de aplicação, além da DATAPREV (responsável pela gestão da Base de Dados Sociais Brasileira, tido como o maior banco de dados da América Latina) entre outros. A primeira realiza serviços de processamento de dados para o governo, notadamente os serviços prestados à secretaria da fazenda como o ReceitaNet, SISCOMEX (Sistema Integrado de Comércio Exterior). Já a empresa COBRA Tecnologia é especializada em serviços para o setor financeiro e é a principal empresa fornecedora de serviços de TI para o Banco do Brasil, que é um dos maiores usuários corporativa de SLCA no mundo e é tido como centro de excelência em Software Livre. Além dos ministérios, outros ramos do governo já adotaram soluções em SLCA, tais como Caixa Econômica Federal, Banco do Brasil, EMBRAPA entre outros são demandantes de software livre (relação completa destes, que é datada do ano de 2010 e pode ser consultada na seguinte página da web: <http://www.softwarelivre.gov.br/levantamento/levantamento/levantamento>).

Ademais, o Portal do Software Público Brasileiro disponibiliza o código fonte de diversas soluções livres e comunidades de desenvolvimento em vários segmentos tais como saúde, educação, geoprocessamento entre outros que são desenvolvidos por órgãos públicos. Estas soluções são criadas no formato de comunidades de desenvolvedores e o código fonte destas soluções é gratuitamente disponibilizado mediante cadastro. Com isso, o governo logrou um passo para

aprimorar continuamente suas aplicações além de promover um espaço de aprendizado para empresas que desejam aprimorar-se nestas soluções favorecendo, assim a inovação.

Porém, conforme argumenta Diegues (2010), há de se levar em conta que todo este potencial ainda é limitado. O autor leva em consideração que o potencial de impacto de SLCA ainda é reduzido argumentando que ainda coloca-se em questão a viabilidade de negócios baseados em SLCA e a compatibilidade desta iniciativa com a PITCE.

“vale chamar a atenção para o fato de que desde 2003 a Casa Civil estabeleceu como diretriz geral para o Governo Federal a aquisição de softwares livres. Longe de entrar no mérito do impacto desta medida para a redução dos custos dos sistemas de informação adotados na esfera pública, é importante salientar que ainda há diversas discussões e incertezas, inclusive em âmbito internacional, sobre as características e a viabilidade dos modelos de negócio das empresas baseadas em software livre (...). Também vale destacar que, aparentemente, tal orientação também possui uma certa incompatibilidade com as diretrizes da PITCE de consolidação das empresas nacionais de software, dado o fato que os modelos de negócios da esmagadora maioria destas empresas e principalmente das maiores empresas brasileiras não são baseados em software livre” (DIEGUES, 2010, p. 230).

Conforme observamos nos dois capítulos anteriores, modelos de negócios baseados em software livre se dão das mais diversas maneiras. O mais importante, contudo, a ser considerado em cada uma das estratégias é a relação a médio e longo prazo com comunidades, a continuidade da aplicação, o crescimento de sua aceitação e melhoria contínua. Foi observado também que não só a intervenção do governo vem impulsionando a adoção de soluções SLCA. Empresas de grande porte e de vários países, que tradicionalmente desenvolvem aplicações em código fechado, já têm algum tipo de iniciativa visando obter ganhos com SLCA ou mesmo se antecipando à tendência.

Ademais, conforme argumentado, soluções SLCA são muitas vezes de caráter complementar a soluções de software proprietário e tem o potencial de abrir novos caminhos para a expansão de outros segmentos menos explorados, notadamente o de serviços em software. Deve-se lembrar que as grandes barreiras à entrada em setores de software produto resultam em lentidão e dificuldade na ascensão de novos programas enquanto que este mesmo mercado é um dos mais dinâmicos em termos de tendências e inovações. Logo, tem-se que o cenário

apresenta grandes dificuldades e ao mesmo tempo oferece grandes oportunidades de modo que a questão da estratégia desenvolvida e a difusão de informações são de importância central para seu desenvolvimento.

Sobre a intervenção governamental, percebe-se que o governo brasileiro enxerga falhas de mercado e necessidade de fomento ao setor. As diretrizes observam SLCA como um produto tecnicamente superior capaz de reduzir custos e ampliar as ações de governo eletrônico e procuram transbordar estes ganhos para a sociedade como, por exemplo, as ações do portal do software público, a elaboração de guias de migração para SLCA e ação de empresas estatais para o público e para outras empresas. De fato, são poucas as empresas que não são estatais que se lançam em projetos baseados em SLCA, contudo, a iniciativa ainda é recente e já logrou expansão na aceitação e conhecimento acerca de software livre dentro do país além de contribuir para uma diversificação e ocorrência de inovação dentro das atividades no mercado. O redirecionamento de gastos com licenças - que geralmente vão para fora do país - para outros setores da indústria, certamente impulsiona esta diversificação e a sua própria expansão como um todo. Com isso, pode-se verificar consonância com o que está disposto na PICTE, pois a intervenção busca uma ampliação do setor, porém em meios ainda pouco explorados de modo a aumentar os limites e abrangência da indústria nacional como um todo.

#### 4. Conclusão

O processo de produção de SLCA é dos mais diversos, assim como as suas interpretações econômicas. Percebe-se que nem todas as iniciativas comerciais lograram sucesso e nem todas as intervenções governamentais surtiram efeitos imediatos. Por outro lado, o que a maioria das ações de sucesso tiveram em comum foram a compreensão e foco nas motivações dos integrantes das comunidades e como estas se comportam, nos segmentos de mercado a que se destinavam e em como transformar invenções em inovações. Não apenas os condicionantes para o desenvolvimento eficiente de SLCA - no sentido de possibilitar a melhor alocação de esforços possível capaz de garantir o crescimento deste mercado – estão bem presentes na conjuntura brasileira, mas também se pode contar com apoio do governo para diversas áreas.

Assim, encontra-se ao mesmo tempo um cenário promissor, porém pouco explorado neste segmento, de modo que ainda há muito para se aprender, sendo assim, dificultoso predizer seus rumos nos anos vindouros. Já é possível observar, principalmente no exterior, as primeiras turbulências pela qual vêm passando as grandes empresas focadas em SLCA. Ao mesmo passo, há que se atentar para as grandes oportunidades de crescimentos, incentivados principalmente pela crescente demanda por boa parte dos outros segmentos da economia por estes produtos e serviços, especialmente pelo estado, o qual também é desenvolvedor e fomenta a participação da sociedade em suas iniciativas. Vale à pena destacar que os bancos públicos brasileiros estão entre os maiores demandantes mundiais de soluções em software livre e que seu programa de adoção de SLCA vem obtendo resultados em cada vez mais abrangentes nas diferentes esferas administrativas. Ademais, o caráter dinâmico da indústria de software no tocante a inovações e tendências abre espaço para novas iniciativas de modo a ressaltar também a importância da qualificação, do empreendedorismo e dos efeitos de rede.

O sucesso do SLCA significaria mais do que constatações acerca de como funcionam modelos de produção baseados em colaboração e tecnologia aberta. Significa também a expansão e diversificação de um setor industrial extremamente relevante para o mundo e muito promissor para o Brasil.

## 5. Bibliografia

ABES: “Mercado Brasileiro de Software: panorama e tendências”, 2010

AMICCI, F. L. “**Software sob Encomenda: um estudo exploratório de segmentação em posicionamento no mercado empresarial**”, dissertação de mestrado, Departamento de Administração – Universidade de São Paulo, FEA/USP, 2004.

AGHION, P. & TIROLE, J. “**The Management of Innovation**”, Quarterly Journal of Economics, 109, pp 1185-1209, 1994

BESSEN, J. “**Open Source Software: Free Provision of Complex Public Goods**” Boston University School of Law and Research on Innovation, 2005

BITZER, Jürgen & SCHRÖDER Philipp J. H. “**The Economics of Open Source Development**” Elsevier (2006)

Cusumano, M. A. “**Japan’s Software Factories: A Challenge to U.S. Management**”, Oxford University Press, 1991.

GRUPO DE TRABALHO MIGRAÇÃO PARA SOFTWARE LIVRE, “**Guia Livre – Referência de Migração Para Software Livre**”, 2005,

LEMONS, R. & JÚNIOR S. V. B. et al “**Estudo sobre o software livre comissionado pelo Instituto Nacional da Tecnologia da Informação (ITI)**”, Fundação Getúlio Vargas FGV, 2005

LINDENBERG, S. “**Intrinsic Motivation in a New Light**”, Kyklos (pp 542/3, 317-343), 2001.

MANENTI, F. & COMINO S. “**F/OSS vs. Closed Source Software: Public Policies for the Software Market**”, Universidade de Padova, (2003)

MONIZ, CERDEIRA, et al “**Eficiências Tecnológica, Econômica e Social**” p. 68).

PRESSMAN, R. “**Software Engineering: a practitioner’s approach**”, McGrawHill, 2001.

RAYMOND, Eric. S. **The Cathedral and The Bazaar**. O’Reily (1999)

RIEHLE, D. **“The Single Vendor Commercial Open Source Business Model”**, University of Erlangen-Nürnberg, 2007

ROSELINO, J.R., **“A Indústria de Software: o “modelo brasileiro” em perspectiva comparada”**, dissertação de doutoramento. Departamento de Economia – Universidade de Campinas UNICAMP, 2006

ROSSI, M. A. **“Decoding the “Free/Open Source (F/OSS) Software Puzzle: a survey of theoretical and empirical contributions”**UniversitàdegliStudi di Siena, 2004.

VARIAN, H. e SHAPIRO C. **“Linux Adoption in The Public Sector: An Economic Analysis”**(2003).