Capítulo 8

Software Livre e Propriedade Intelectual: Aspectos Jurídicos, Licenças e Modelos de Negócio

Fabio Kon, Nelson Lago, Paulo Meirelles e Vanessa Sabino

Abstract

Free, Libre, and Open Source Software (FLOSS) has increasingly been shown to be a viable alternative for the production, distribution, and usage of quality software in several academic, cientific, business, government, and commercial environments. Nonetheless, business and contract models traditionally used in the contemporary society are not perfectly suited for the typical mechanisms by which free software is produced, distributed, and used. This chapter discusses how specific characteristics of FLOSS affect its usage by individuals, companies, and governments and, in particular, it focuses on how these characteristics affect business models within the software industry. The chapter presents a short historical perspective and description of the movement's dynamics and describes some of the major FLOSS licences and their consequences. It discusses the opportunities and difficulties on the interaction with the community, and the major business models associated with FLOSS, with examples of successful approaches.

Resumo

O software livre tem se apresentado como uma alternativa viável para a produção, distribuição e utilização de software de qualidade em uma grande gama de contextos acadêmicos, científicos, empresariais, governamentais e comerciais. No entanto, os modelos de negócio e de contratos tradicionalmente utilizados na sociedade contemporânea não são perfeitamente adequados para a forma como o software livre é produzido, disseminado e utilizado. Este capítulo discute como as especificidades do software livre se refletem sobre o seu uso por indivíduos, empresas e governos e, em particular, nos modelos de negócio aplicados à indústria de software. Além de um breve histórico do movimento e de uma descrição da sua dinâmica, são descritas as principais licenças de software livre e suas consequências, as oportunidades e dificuldades de interação com a comunidade e os principais modelos de negócio relacionados ao software livre, com exemplos de casos de sucesso.

8.1. Introdução

Um dos componentes fundamentais de qualquer sistema de computação é o software, que efetivamente faz uso do hardware para atingir os mais diversos objetivos. De fato, pode-se dizer que o software expressa a solução abstrata dos mais diversos problemas computacionais, enquanto o hardware é o meio pelo qual o software produz resultados palpáveis. Isso significa que o software traz consigo um amplo corpo de conhecimento relacionado aos mais diversos problemas aos quais a computação costuma ser aplicada.

Assim, diferentemente do que acontece com o hardware, o software se apresenta cada vez mais como um tema de interesse geral, e não apenas para profissionais da área de computação, na medida em que vários aspectos relacionados a ele vão além de características puramente técnicas. Em particular, alguns desses aspectos têm levantado interesse crescente: (a) O processo de desenvolvimento do software, (b) os mecanismos econômicos que regem esse desenvolvimento e seu uso, (c) o relacionamento entre desenvolvedores, fornecedores e usuários do software e, finalmente, (d) os aspectos éticos e legais relacionados ao software.

Boa parte do software atualmente usado e desenvolvido, tanto em computadores pessoais quanto em servidores ou aplicações verticais, é disponibilizado sob licenças restritivas. Essas licenças, em maior ou menor grau, impõem restrições ao seu uso, distribuição ou acesso ao código-fonte. Esse tipo de licenciamento é possível porque o software está sujeito à proteção da lei a respeito dos direitos de autor, que garante ao autor o direito exclusivo de exploração de sua obra. Isso permite ao autor autorizar ou não determinadas formas de uso do software por parte dos usuários. Chamamos o software disponibilizado sob licenças que impõem restrições desse tipo de *software restrito*, *exclusivo* ou *proprietário*[1].

Em contraste com o software restrito, o *software livre* propõe um novo mecanismo de licenciamento, em que o software pode ser utilizado, redistribuído e modificado praticamente sem restrições. Essa abordagem para o licenciamento do software tem impacto muito maior que o que se poderia imaginar, pois estabelece uma dinâmica única e potencialmente muito positiva em relação aos aspectos citados acima.

8.1.1. Por que Software Livre?

O software pode ser visto de diversas maneiras: De um lado, pode ser encarado como uma ferramenta ou meio de produção; De outro, como uma forma de conhecimento; Ainda de outro, como uma extensão do pensamento humano etc. A cada ponto de vista correspondem as mais diversas ideias sobre como maximizar seu benefício para a sociedade, e a cada um deles corresponde também uma visão sobre o software livre.

Essas ideias e pontos de vista são determinantes na opção ou não pelo software livre. No entanto, pode-se observar que o software livre tem sido crescentemente adotado por pessoas, empresas e governos. De maneira geral, podemos agrupar as principais razões para essa escolha em três categorias:

[1] Esta última denominação, embora comum, é um anglicismo recente.

Software livre como opção ética: Por este ponto de vista, na medida em que o software pode ser facilmente compartilhado, a imposição de restrições artificiais ao compartilhamento é eticamente questionável, já que o benefício que pode ser obtido pelo software é diminuído. Além disso, o incentivo ao compartilhamento incentiva indiretamente o espírito de comunidade.

Software livre como parte de uma melhor sociedade no futuro: Por este ponto de vista, restrições ao software podem se traduzir em dificuldades para governos, empresas e cidadãos, já que o software traz consigo regras e decisões tomadas em função de interesses diversos. Além disso, a falta de acesso ao código-fonte diminui as possibilidades de aprendizado da computação e do entendimento das normas expressas pelo software.

Software livre como base para negócios: Por este ponto de vista, diferentemente do que se poderia pensar à primeira vista, o software livre oferece oportunidades e vantagens quanto à exploração comercial do software, tanto para fornecedores quanto para usuários.

Embora essas três razões sejam importantes e muitas vezes se confundam, neste texto o foco principal será nos aspectos econômicos associados ao software livre.

8.1.2. Problemas do Software Restrito

Embora haja um grande volume de software disponibilizado sob licenças restritivas, diversos problemas afetam esse software. A solução desses problemas ainda é uma questão aberta, e novos modelos de desenvolvimento e comercialização de software podem ser interessantes para a indústria e a sociedade em geral.

Um dos problemas centrais de qualquer sistema de software é a complexidade. Como não existem restrições físicas no software (como resistência de materiais, custos de fabricação, peso etc.), não existem elementos exteriores que possam restringir a evolução de um sistema. Assim, a maioria dos programas não-triviais atinge rapidamente grandes níveis de complexidade, aproximando-se do limite na capacidade de compreensão do programador [Stallman 2002].

Uma consequência dessa complexidade é o alto custo. Embora não haja custos significativos com equipamentos, o desenvolvimento de software envolve o empenho de profissionais qualificados por longos períodos, gerando um forte impacto no custo de um projeto. No entanto, o fato de o software poder ser copiado (diferentemente do que ocorre com os equipamentos de uma linha de produção, por exemplo) sugere que o empenho de diferentes empresas em desenvolver soluções similares é um esforço desperdiçado. O compartilhamento de ideias e de código entre diversos programadores pode permitir a diluição dos custos de desenvolvimento entre todos os envolvidos.

Outra consequência dessa complexidade é a perda de qualidade. Com o aumento de complexidade, torna-se cada vez mais difícil realizar testes eficientes, compreender todas os casos possíveis, eliminar erros e ainda implementar novas funcionalidades. No entanto, o compartilhamento do código, envolvendo um maior número de programadores

e usuários, pode permitir a identificação e correção mais rápida de problemas, bem como uma maior evolução do sistema[2].

Um outro problema do software restrito vem da necessidade de garantir que as restrições sejam cumpridas pelos usuários, especialmente no tocante às cópias ilegais. A administração desse aspecto envolve um grande esforço por parte dos usuários, que precisam implementar métodos para evitar que terceiros realizem cópias ilegais a partir de suas cópias legais, bem como lidar com mecanismos anti-cópias que muitas vezes trazem consigo inconvenientes. A situação para os fornecedores não é muito melhor, pois é necessário aplicar recursos nesse problema, bem como gerir os possíveis problemas que possam vir a surgir com clientes por conta de problemas ou erros referentes à legalidade das cópias. Vale também observar que as cópias ilegais muitas vezes funcionam como mecanismo de divulgação do software[3] e, portanto, restringi-las pode ter efeitos negativos para os fornecedores.

Em sistemas restritos, a migração de um fornecedor para outro muitas vezes exige uma migração para um novo pacote de software. Se o fornecedor simplesmente sair do mercado por alguma razão, essa migração se torna compulsória; Se o fornecedor impuser termos de renovação de contrato desvantajosos, os custos associados a essa migração acabam por levar o usuário a aceitar esses termos. Essa situação ocorre porque o código-fonte normalmente não está diponível para o usuário; Essa restrição, no entanto, não advém de nenhuma característica instrínseca do software, mas sim do mecanismo de licenciamento, gerando um aumento artificial nos custos e riscos associados ao software.

Finalmente, os modelos de exploração comercial do software com base na venda de licenças de uso não tira nenhum benefício das características específicas do software: A possibilidade de cópia e compartilhamento com custo nulo e o fato de o software ser uma forma de conhecimento que pode ser expandido através da troca de ideias. Ao invés disso, esses modelos procuram comercializar o direito de uso do software da mesma maneira que bens físicos, criando dificuldades artificiais desnecessariamente.

Poder-se-ia pensar que essa abordagem é necessária pois a venda dessas licenças seria o único meio para viabilizar os custos ligados ao desenvolvimento, suporte, implantação, entre outros, do software. No entanto, a maior parte (mais de 80%) tanto do dinheiro gasto com software pelas empresas quanto dos postos de trabalho no mercado de software são voltados para aplicações personalizadas e treinamento, por exemplo, e não para a compra de licenças de software [Ghosh et al. 2006, p. 77]. Assim, todo o esforço para garantir o funcionamento desse modelo tem impacto real sobre apenas 20% da renda dos fornecedores. Como consequência, aplicações personalizadas muitas vezes precisam ser desenvolvidas do zero, com o consequente alto custo, mesmo que já existam soluções similares.

[2]Uma evidência a favor desse ponto de vista é o estado atual do núcleo de sistema operacional conhecido como Linux. Embora seja similar aos sistemas Unix que já existiam há mais de dez anos e que eram desenvolvidos por empresas de porte como Sun Microsystems e IBM, o Linux desbancou esses sistemas em praticamente todas as suas aplicações tradicionais, como servidores de rede, sistemas de alto desempenho, sistemas de tempo real etc. além de ter sido adaptado também para outras áreas, como sistemas de uso doméstico, sistemas embarcados, telefones celulares etc.

[3]Programas disponibilizados como *shareware* ou *demos* se beneficiam da oferta do programa como meio de divulgação.

8.2. Software Livre como Alternativa

Em princípio, o que diferencia o software livre do software restrito é apenas a forma de licenciamento; No entanto, essa diferença tem por consequência diversas outras, algumas bastante acentuadas. Várias delas se traduzem em benefícios em relação aos problemas elencados acima. Assim, o software livre se apresenta como uma alternativa vantajosa e de qualidade para o desenvolvimento de sistemas computacionais em diferentes ambientes e contextos, incluindo universidades, empresas, governos e ONGs.

8.2.1. Vantagens do Software Livre

Uma vantagem oferecida pelo software livre em comparação ao software restrito vem do fato que o código-fonte pode ser livremente compartilhado. Esse compartilhamento pode simplificar o desenvolvimento de aplicações personalizadas, que não precisam ser programadas a partir do zero, mas podem se basear em soluções já existentes. Na medida em que o desenvolvimento de aplicações personalizadas é um dos focos do desenvolvimento de software em geral [Ghosh et al. 2006, p. 77], essa vantagem tem impacto significativo na redução de custos e na diminuição na duplicação de esforços, tirando proveito da característica abstrata do software.

Outra vantagem resultante do compartilhamento do código se refere à possível melhoria na qualidade, mesmo frente aos problemas inerentes à sua complexidade [Raymond 1997]. Isso se deve ao maior número de desenvolvedores e usuários envolvidos com o software: De um lado, um número maior de desenvolvedores, com diferentes perspectivas e necessidades, é capaz de identificar e corrigir mais *bugs* em menos tempo; De outro, um número maior de usuários gera situações de uso e necessidades mais variadas, o que se traduz em um maior número de *bugs* identificados e mais sugestões de melhorias.

A reputação do programador também acaba se tornando um fator relevante para a qualidade do software livre. Enquanto o código-fonte do software restrito é geralmente secreto, o código-fonte do software livre é público. Como consequência dessa exposição, o orgulho pessoal do programador, que sabe que sua produção será avaliada por outros e possivelmente terá reflexos em sua carreira profissional, o leva a ser mais cuidadoso.

O software livre também traz vantagens do ponto de vista econômico. Diferentemente do que ocorre com o software restrito, o software livre promove o estabelecimento de vários fornecedores que competem entre si com base no mesmo software. Essa competição mais forte entre fornecedores traz vantagens para os usuários, pois dá melhores garantias quanto ao desenvolvimento futuro do sistema e induz a uma redução nos preços.

De forma similar, os fornecedores também se beneficiam do compartilhamento do software livre, pois tanto os custos quanto os riscos associados ao desenvolvimento do software são diluídos entre os diversos concorrentes.

Finalmente, como já mencionado, o software livre se insere na principal fatia do mercado de software (onde circula 80% do dinheiro) sem estar sujeito aos problemas do software restrito. Por conta disso e das vantagens econômicas descritas anteriormente, ele possibilita e até mesmo incentiva o surgimento de pequenas empresas que podem atender seus mercados locais. Por sua vez, esse incentivo às pequenas empresas e a consequente redução na dependência de empresas estrangeiras são economicamente interessantes para

o Brasil.

8.2.2. Desvantagens do Software Livre

Apesar das vantagens mencionadas, o software livre também tem limitações. Algumas delas são comuns ao software livre e ao restrito; Outras advém de suas características específicas, colocando-o numa situação desfavorável frente ao software restrito.

Uma das desvantagens do software livre comumente citadas consiste na ausência de garantias. A maior parte do software livre deixa claro em suas licenças que os autores se eximem de qualquer forma de responsabilidade por problemas gerados pelo software. Embora isso também costume ser verdadeiro com relação ao software restrito, a legislação em geral limita o quanto o autor ou fornecedor pode se eximir de responsabilidade. No entanto, em caso de conflito legal, o valor efetivamente pago pelo produto tende a ser levado em consideração, o que significa que a responsabilidade no caso do software livre tende a ser minimizada. Por outro lado, existem empresas que oferecem contratos de prestação de serviços baseados em software livre onde essas empresas assumem formalmente a responsabilidade pelo funcionamento adequado do produto.

Outro problema comum com relação ao software livre é a avaliação de qualidade. No caso do software restrito, um dos meios de avaliação de qualidade é identificar as características da empresa que desenvolve o software: Tamanho, equipe etc. No caso do software livre, se não há uma empresa de porte razoável por trás do desenvolvimento, é difícil avaliar a qualidade e identificar, entre as várias alternativas disponíveis, qual a mais adequada.

Um problema relacionado se refere à perspectiva futura do software. Mais uma vez, no caso do software restrito é possível tentar estimar as perspectivas para o futuro do software através da avaliação da empresa que o desenvolve. No entanto, é mais difícil avaliar se um determinado software livre vai continuar sendo mantido no futuro.

Por outro lado, a disponibilidade do código-fonte alivia esses dois problemas, pois é possível avaliar a qualidade do próprio código e, mesmo que o software seja abandonado por seus desenvolvedores originais, o usuário pode continuar investindo em sua manutenção.

Ainda outro problema é a sustentabilidade do software e das empresas envolvidas com ele. Como mencionado, o compartilhamento do código tende a incentivar a criação de várias empresas que, com base no mesmo software, concorrem entre si através da oferta de serviços baseados nesse software. Essa maior concorrência e a impossibilidade prática da venda de licenças tendem a tornar o software uma *commodity* e podem reduzir o faturamento e o lucro efetivo dessas empresas, possivelmente inviabilizando a sustentação das empresas [Weber 2004]. Isso pode, indiretamente, ter impacto negativo sobre o desenvolvimento do software, que pode acabar por ter um número cada vez menor de desenvolvedores.

Outro aspecto potencialmente problemático advém da relevância que é dada à reputação, que pode trazer algumas distorções para a posição das empresas no mercado e na comunidade. Uma consequência dessa valorização é que uma nova empresa que procure se estabelecer através da oferta de serviços com software livre tem dificuldade em

obter visibilidade, pois as empresas já conhecidas costumam ser mais valorizadas. Mesmo que a nova empresa implemente novas funcionalidades no software, essas funcionalidades são automaticamente absorvidas pelas concorrentes, eliminando a possibilidade de uma diferenciação rápida no mercado [DiBona et al. 1999]. Esse só não é o caso quando a licença permite que o software livre seja relicenciado como software restrito; No entanto, nesse caso, é difícil conciliar essa abordagem com a imagem junto à comunidade. Em geral, empresas que se utilizam de software livre dessa maneira mas não contribuem em nada com o bom andamento da comunidade e do software compartilhado não são bem recebidas.

O compartilhamento do código também significa que não há segredos industriais. Em alguns contextos, contratos envolvendo segredos industriais impossibilitam a implantação de uma solução como software livre; Em outros, a posse exclusiva do conhecimento relacionado é uma vantagem competitiva significativa, tornando a publicação de sistemas de software livre que lidam com esse conhecimento economicamente desvantajosa.

Um outro aspecto problemático se refere à imagem do software livre. A pouca experiência do mercado em lidar com o software livre e o próprio fato de o software ser, em geral, gratuito, podem gerar dúvidas sobre a viabilidade econômica, confiabilidade ou a qualidade do software.

Finalmente, uma dificuldade que é compartilhada pelo software livre e pelo sofware restrito é a existência de patentes de software. Esse tipo de patente é muito comumente considerada desvantajosa para a comunidade e o mercado de software, a ponto de não ser reconhecida na maioria dos países[4]. No entanto, na prática, vários países reconhecem, pelo menos parcialmente, patentes desse tipo[5] e, na medida em que o software livre é um fenômeno mundial, restrições causadas por patentes em um país têm efeito sobre toda a comunidade.

8.2.3. Breve Histórico do Movimento do Software Livre

O software livre muitas vezes é considerado um fenômeno recente que veio à tona rapidamente nos últimos anos. No entanto, desde o início da Computação a maior parte dos desenvolvedores trabalhava da forma que hoje identificamos com o software livre: Compartilhando código de forma aberta.

Com a evolução do mercado de Computação, o software passou a ser identificado como algo de valor dentro da estratégia de algumas empresas. Foi assim que, no final da década de 1970 e início dos anos 1980, a situação mudou completamente e o software restrito estabeleceu-se como a "forma natural" de distribuição de software [González-Barahona et al. 2009]. Por outro lado, como uma resposta natural a esse movimento, foi durante esse mesmo período que programas explicitamente livres começaram a surgir e os fundamentos do que conhecemos hoje como software livre começaram a ser conceituados.

Ao longo do tempo, esses princípios se transformaram em uma tendência. Com a popularização da Internet na década de 1990, o software livre como movimento e

[4]entre eles o Brasil: "Não se considera invenção[...] métodos matemáticos[...], concepções puramente abstratas[...], programas de computador em si" [PRESIDÊNCIA DA REPÚBLICA 1996, artigo 10].

[5]em particular os EUA

alternativa tecnológica progrediu e amadureceu, tornando-se um fenômeno significativo que ajudou a transformar a indústria de software. De forma complementar, o próprio crescimento da Internet como uma "rede das redes" aberta e livre se deu graças à existência de uma implementação completa do protocolo TCP/IP sob a forma de software livre na época[6], o que facilitou e barateou drasticamente sua implantação em várias plataformas (inclusive nos vários sistemas da família MS-Windows). O fato de o padrão TCP/IP ser livre e aberto e a existência de uma implementação como software livre fomentaram o crescimento da Internet em detrimento de outras redes existentes à época, baseadas em tecnologias restritas. Essa sinergia é um dos muitos exemplos dos resultados benéficos da adoção de software e padrões livres.

8.2.3.1. No início, o software era livre

Nos anos 1960, os computadores de grande porte, utilizados quase que exclusivamente em grandes empresas e instituições governamentais, dominavam o mercado de Computação. Nessa época, não era comum do ponto de vista comercial a ideia do software como algo separado do hardware. O software era entregue junto com o código-fonte ou, em muitas vezes, apenas o código-fonte era entregue [González-Barahona et al. 2009]. Existiam grupos de usuários, tais como SHARE (usuários de sistemas IBM) e DECUS (usuários da DEC – *Digital Equipment Corporation*), que compartilhavam código e informações. A seção da revista *Communications da ACM*, chamada "Algoritmos", foi outro bom exemplo de um fórum de discussão e compartilhamento. Assim, podemos constatar que, no início, o software era livre: pelo menos aqueles que tinham acesso à tecnologia da época podiam normalmente ter acesso ao código-fonte, modificá-lo e compartilhá-lo com seus amigos, colegas e demais usuários dos grupos de interesse. De fato, esse mecanismo de compartilhamento de código e conhecimento foi responsável, em parte, pelo grande avanço da Ciência da Computação em seus primeiros anos de existência.

Linha do tempo:

1956: O governo dos EUA proíbe a AT&T de entrar no comércio de software (isso levou, posteriormente, o *Bell Labs* a distribuir livremente o seu Unix).

1960: O software é distribuído com seu código-fonte e sem nenhuma restrição em grupos de software como SHARE (IBM) e DECUS (DEC).

1969: O RFC (*Request for Comments*), que descreve a primeira Internet (depois chamada de ARPANET), é publicado; Ken Thompson e Dennis Ritchie, pesquisadores do Bell Labs, criaram a primeira versão do Unix, um sistema operacional multitarefa.

8.2.3.2. Surgimento do software restrito e o berço do software livre

A IBM era a fabricante líder do mercado de computadores de grande porte e estava significativamente à frente de sua concorrência. Em 30 de junho de 1969, a IBM anunciou que, a partir de 1970, iria vender parte de seus programas separada do hardware [Grad

[6] A "pilha" TCP/IP do sistema BSD, distribuída sob licença permissiva — cf. Seção 8.4.1.

2002]. Desde então, a indústria de software mudou sua cultura, o que tornou cada vez mais comum as restrições de acesso e as possibilidades de compartilhamento do código entre os desenvolvedores. Estudar o software real, utilizado nos computadores comerciais, passou a ser algo mais limitado tanto do ponto de vista técnico quanto legal. Assim, esse novo modelo, em que o software era comercializado como os produtos na prateleira de um supermercado, foi se transformando no padrão da indústria de software. Sua consolidação pode ser exemplificada pela "carta aberta aos hobistas" [7], escrita por Bill Gates aos 21 anos, que levanta o potencial comercial do software como produto no mercado de microcomputadores e questiona a viabilidade do desenvolvimento fora desse "modelo de prateleira". Nessa carta, Gates afirma que o total de *royalties* recebidos pelo Altair BASIC era equivalente a apenas dois dólares por hora gasta em seu desenvolvimento e documentação. Ele ainda alega que a prática de compartilhamento de software não é justa e afirma que tal prática evita que software bem feito seja escrito.

Por outro lado, no mesmo período, houve iniciativas que colocaram em prática algumas das características do que viria a ser apresentado mais tarde como software livre. Algumas dessas iniciativas levaram à produção de sistemas de software livre que estão em uso até os dias de hoje. Entre eles, destacam-se o SPICE, TeX e Unix. Em 1973, o SPICE (*Simulation Program with Integrated Circuit Emphasis*) foi colocado em domínio público por Donald Pederson. Com o tempo, ele tornou-se o software de referência para simuladores de circuitos integrados. Já em 1978, Donald Knuth, da Universidade de Stanford, começou a trabalhar no TeX, um sistema eletrônico de formatação de textos, distribuído como software livre. Entre os citados, o mais interessante e complexo é o caso do Unix, um dos primeiros sistemas operacionais portáteis, originalmente criado por Thompson e Ritchie, entre outros, do AT&T Bell Labs.

Desde 1972, o Unix está em contínuo desenvolvimento e deu origem a diversas variantes, comercializadas por dezenas de empresas. No mesmo ano de sua criação, o Unix começou a ser distribuído nas universidades e centros de pesquisas. Em 1973, o Unix chegou na Universidade de Berkeley, na Califórnia, que passou a manter o Unix BSD (*Berkeley Software Distribution*). Na década de 1980, a AT&T mudou sua política de acesso às novas versões do Unix, tornando-o mais caro e com restrições de distribuição.

O Unix tornou-se extremamente popular entre os desenvolvedores por conta da sua filosofia de compartilhamento e relação com as universidades e centros de pesquisa. Mesmo assim, em 1991, a AT&T demonstrou uma total mudança de mentalidade, ao tentar processar a Universidade de Berkeley por conta da publicação do código Unix BSD que o grupo de pesquisa de Berkeley (CSRG – *Computer Systems Research Group*) tinha criado. Desde 1973, o CSRG havia sido um dos principais centros de desenvolvimento do Unix e aplicações a ele relacionadas. Durante os anos 1980, muitas e importantes modificações foram feitas pelo grupo de Berkeley, inclusive no núcleo (*kernel*) do Unix. Além disso, muitas empresas passaram a usar a versão BSD como base para seus sistemas Unix. Dessa maneira, Berkeley tornou-se umas das duas principais fontes do Unix, junto com a "oficial" AT&T [González-Barahona et al. 2009]. Alguns consideram a Universidade de Berkeley como o berço do software livre.

[7] Bill Gates: `en.wikipedia.org/wiki/Open_Letter_to_Hobbyists`

Linha do tempo:

1970: Surge a ARPANET, precursora da Internet; A IBM começa a vender seu software separadamente, estabelecendo assim o início da indústria do software restrito.

1972: Unix começa a ser distribuído em universidades e centros de pesquisa.

1973: Inicia-se a história do Unix BSD com a chegada do Unix à Universidade de Berkeley, na Califórnia; SPICE é colocado por Donald Pederson em domínio público.

1975: Lançada a primeira versão do Ingres, banco de dados livre (o ancestral do PostgreSQL).

1976: Bill Gates escreve a "carta aberta aos hobistas".

1977: A Arpanet atinge mais de 100 computadores.

1978: Donald Knuth, da Universidade de Stanford, começou a trabalhar no TeX, distribuído como software livre.

8.2.3.3. O nascimento do movimento do software livre

Os primeiros projetos organizados de forma consciente para serem software livre foram criados no início da década de 1980, quando, também, os fundamentos éticos, legais e, inclusive, financeiros desse movimento começaram a ser estabelecidos [González-Barahona et al. 2009].

Em 1984, Richard Stallman, então funcionário do laboratório de inteligência artificial do MIT, deixou seu emprego e começou a trabalhar no projeto GNU. Stallman gostava de compartilhar seus interesses tecnológicos, conhecimento e seu código, o que o levou a recusar a assinatura de acordos de exclusividade e de não compartilhamento que eram exigidos em seu ambiente de trabalho no MIT. Segundo Stallman, sua comunidade científica, pessoas dentro de uma rede de universidades que trabalhavam em colaboração, foi destruída por interesses comerciais. O sistema operacional livre que eles usavam tornou-se obsoleto e as pessoas adaptaram-se ao novo modelo comercial, baseado no software restrito, de código fechado. Stallman sentiu-se como tendo perdido sua comunidade e seu modo de viver. O uso de software restrito em seu dia-a-dia o deixou impotente diante de situações que ele mesmo estava acostumado a resolver [DiBona et al. 1999].

Richard Stallman deixou o MIT com a ideia de construir um sistema de software completo, para uso geral e totalmente livre. O sistema e o projeto, que seria responsável por fazer isso virar realidade, foi denominado GNU (*GNU's Not Unix*, um acrônimo recursivo). Stallman incluiu, no projeto, o TeX e o sistema de janela X. Ele começou, então, a escrever o GCC e o Emacs, ferramentas populares até hoje.

Desde o início do projeto GNU, Richard Stallman estava particularmente preocupado com as liberdades que os usuários do software deveriam ter. Para tanto, ele criou um mecanismo legal a fim de garantir que, além daqueles que receberiam os programas diretamente do projeto GNU, todos pudessem desfrutar os direitos de copiar, redistribuir e modificar o software. Também aqueles que recebessem algum software após qualquer número de redistribuições e, eventualmente, modificações, deveriam poder gozar dos mesmos direitos associados ao software original distribuído pelo projeto GNU [González-Barahona et al. 2009]. Por essa razão, ele elaborou a licença GPL (que discutimos na seção de licenças de software livre deste capítulo). Além disso, para institucionalizar

o projeto GNU, bem como obter fundos para desenvolver e proteger o software livre, de acordo com os princípios éticos que ele publicou no "Manifesto GNU" [Stallman 1990], Stallman fundou a *Free Software Foundation* (FSF). Dessa forma, definitivamente nomeado, nasce o Movimento do Software Livre. Stallman é uma figura marcante do movimento pelas suas ideias radicais e pelo seu comprometimento ideológico na defesa da ética e da liberdade. Para Stallman, cada software restrito é um problema social, uma vez que o não compartilhamento ou privação de acesso ao código-fonte é uma forma de impor o poder dos desenvolvedores ou das empresas aos usuários.

Tecnicamente, o projeto GNU foi bem estruturado, inspirado na modularidade do Unix, e com metas muito claras. A metodologia usada foi baseada em grupos relativamente pequenos, geralmente formados por voluntários. A Internet já era usada, de acordo com suas limitações, uma vez que ela não estava amplamente implantada. Mais tarde, seis anos depois de seu início, o GNU não tinha ainda o núcleo (*Kernel*) do seu sistema. Entretanto, ele era bem popular entre muitos usuários das diferentes variantes do Unix e amplamente usado pelos pesquisadores e desenvolvedores de universidades, em especial, porque o que já tinha sido produzido dentro do projeto era reconhecido como estável e de boa qualidade [González-Barahona et al. 2009]. Em suma, através do projeto GNU, Richard Stallman cria e consolida o movimento do software livre.

Linha do tempo:

1981: A IBM fecha acordo com Microsoft para que ela forneça o DOS para o PC, mas desconsidera a relevância do software e abre mão do copyright do DOS, abrindo espaço para a expansão da Microsoft graças ao mercado de clones do PC surgido posteriormente.

1983: Stallman posta mensagem no grupo *net.unix-wizards* com o assunto "new Unix implementation". Anuncia a criação do GNU e explica seus princípios para a necessidade de criação de um novo Unix. Ele menciona que serão necessários um núcleo, um editor e um compilador, entre outras ferramentas. No final ele pede contribuições na forma de máquinas, dinheiro e ajuda para escrever o software.

1984: Richard Stallman pede demissão do AI Lab do MIT para se dedicar ao projeto GNU, e usa o termo "software livre" no " Manifesto GNU"; O primeiro software do projeto GNU é liberado, o GNU Emacs, escrito por Stallman e Guy L. Steele.

1985: O consórcio X distribui o sistema de janela X como software livre; Richard Stallman funda a FSF. FSF define software livre baseado em 4 liberdades fundamentais; GCC, escrito por Stallman e Len Towe, tem sua primeira versão finalizada.

1987: FSF vende cópias do software GNU em fita magnética por 150 dólares para arrecadar dinheiro.

1989: FSF cria o conceito de copyleft e a GPL para garantir as 4 liberdades do software; Michael Stonebreaker, criador do Ingres, lança PostgreSQL como software livre; Cygnus, a primeira empresa que essencialmente começou a prover serviços para software livre, é fundada por Michael Tiemann, David Henkel-Wallace e John Gilmore; Começa a ser desenvolvido o NS (*Network Simulator*), um simulador livre de rede de telecomunicações que passaria a ser o mais usado por pesquisadores de todo o mundo.

8.2.3.4. O Linux e a Internet: Coração e asas para o projeto GNU e o movimento do software livre

Em julho de 1991, Linus Torvalds, um estudante finlandês da Universidade de Helsinki, de 21 anos de idade, divulgou sua primeira mensagem mencionando o seu projeto de construir um sistema livre similar ao Minix (sistema operacional baseado em Unix desenvolvido e licenciado para propósitos acadêmicos por Andrew Tanenbaum). Linus já estava na pós-graduação e resolveu fazer experiências com o novo computador 386 que recebera na época. Ele conseguiu fazer com que um primeiro esboço do que seria o núcleo de seu sistema operacional executasse dois programas concorrentemente. Assim, anunciou na Internet que tinha um protótipo de sistema operacional [Torvalds e Diamond 2002]. Em setembro do mesmo ano, Linus lançou uma versão oficial. Em março de 1994, a versão 1.0, a primeira a ser chamada de estável, foi liberada. Durante esse período, centenas de desenvolvedores se juntaram ao projeto para integrar todo o sistema GNU em torno do núcleo do Linux. Ao contrário dos BSDs, o núcleo do Linux e um grande número de componentes integrados em torno dele foram distribuídos sob licença GPL. Dessa forma, nasceu o sistema operacional GNU/Linux. Linus relata que não queria dinheiro para esta empreitada por uma série de razões. Quando ele postou o Linux originalmente, ele sentiu que estava seguindo os passos de centenas de cientistas e outros acadêmicos: Pessoas que construíram seu trabalho apoiando-se em outros, ou seja, "apoiando-se nos ombros de gigantes" [Torvalds e Diamond 2002].

Em 1992, surgiram as primeiras distribuições GNU/Linux, entre elas a SLS, que mais tarde deu origem ao Slackware (ainda distribuída atualmente). Isso levou à criação de uma competição no mundo dos sistemas empacotados em torno do GNU/Linux. Cada distribuição tenta oferecer uma versão melhor configurada para uso do GNU/Linux em determinado contexto e partem da mesma base para competir entre si, de acordo com as melhorias consideradas importantes pelas suas bases de usuários [González-Barahona et al. 2009].

Desse momento em diante, o modelo original de desenvolvimento de software e compartilhamento de conhecimento foi resgatado, porém coexistindo com o modelo de software restrito que ganhara força nas duas décadas anteriores. O aspecto mais revolucionário do movimento de software livre não está na questão do código ser aberto, mas no fato de ter sido a primeira comunidade a explorar as novas possibilidades de desenvolvimento e colaboração, geograficamente distribuídos, que a Internet possibilita. A Internet viabilizou a criação das primeiras comunidades, como a BSD e a FSF, bem como o desenvolvimento do Linux. A Internet teve seu maior impacto na indústria de software através do desenvolvimento de software livre, que vem transformando essa indústria desde então [Wasserman 2011].

Linha do Tempo

1990: A FSF anuncia que pretende desenvolver um núcleo para o sistema GNU, chamado de GNU Hurd, com o objetivo de completar o sistema operacional.

1991: William e Lynne Jolitz escrevem uma série sobre "como adaptar o BSD Unix para PC i386"; Linus Torvalds anuncia a criação do "Free Minix", usando as ferramentas do projeto GNU,

como o GCC, e poucos meses depois lança a primeira versão do Linux.

1992: A força aérea dos Estados Unidos faz um contrato com a universidade de Nova York para desenvolver uma versão livre do compilador ADA. Eles escolhem o GNU GCC como base e criam o GNAT (*GNU NYU Ada 95 Translator*); Lançado 386BSD 0.1, que depois deu origem ao NetBSD, FreeBSD e OpenBSD.

1993: A empresa SuSE é fundada com negócios são baseados em Slackware traduzido para o Alemão; Ian Murdock inicia o desenvolvimento da distribuição Debian do GNU/Linux; FreeBSD 1.0 é disponibilizado na Internet.

1994: A *Ada Core Technologies* é fundada pelos desenvolvedores do GNAT e se torna líder do mercado de compiladores ADA; Debian GNU/Linux 0.91 é lançado, como resultado do desenvolvimento voluntário de 20 pessoas; Marc Ewing lança a primeira versão do Red Hat Linux.

1995: Bob Young funda a *Red Hat Software* ao comprar a distribuição Red Hat Linux; O Red Hat Linux 2.0 é lançado, a primeira com o formato de empacotamento RPM; Apache 0.6.2, primeira versão oficial, é lançado; Criado o MySQL.

1996: O projeto KDE é anunciado com o objetivo de tratar os problemas de usabilidade para o usuário final de ambientes similares ao Unix, que utilizam o sistema de janelas X.

8.2.3.5. Eric Raymond, a criação da OSI e o restante da história

Vários outros fatos marcam o crescimento e amadurecimento do movimento do software livre na década de 1990. Entre esses, em 1997, Eric Raymond, apresenta o artigo e palestra "A catedral e o bazar", onde discute as vantagens técnicas do software livre e aborda os mecanismos de funcionamento do desenvolvimento descentralizado. As experiências e o relato que Raymond escreveu influenciaram diretamente a Netscape, que, em 1998, liberou o código-fonte do navegador Mozilla sob licença livre, uma vez que ela estava sob a pressão da competição com o Internet Explorer da Microsoft.

Em 1998, Raymond também foi um dos protagonistas, junto com Linus Torvalds, da criação da *Open Source Initiative* (OSI), defendendo a adoção do software livre por razões técnicas e sugerindo o uso da expressão "open source" ao invés de "free software" para desvincular o movimento da ideologia da Free Software Foundation. A principal motivação para a adoção da expressão "open source" foi introduzir o software livre no mundo dos negócios de uma forma mais palatável para empresas mais conservadoras, evitando a ambiguidade do termo "free" (que pode significar tanto livre quanto gratuito, na língua inglesa).

Para divulgar o lançamento desta nova forma de apresentar o movimento do software livre, Raymond publica, em fevereiro de 1998, o texto *Goodbye, "free software"; hello, "open source"* [8].

Alguns membros da comunidade, em especial Richard Stallman e parte da comunidade em torno da FSF, não concordaram com o uso do termo pois, para eles, o termo software livre é mais apropriado. Assim, Stallman publica, em resposta, o texto: *Why "Free Software" is better than "Open Source"* [9].

[8] `www.catb.org/~esr/open-source.html`

[9] `www.gnu.org/philosophy/free-software-for-freedom.html`

Como a receptividade do termo *Open Source* nas empresas foi bem significativa, ele foi adotado por muitos como a melhor maneira de se referir ao software livre, particularmente na língua inglesa. Na prática, FSF e OSI concordam com o compartilhamento de código. Além disso, apesar de Stallman e muitos do movimento do software livre não concordarem com uso do termo "open source" e a postura mais pragmática da OSI, eles veem a OSI com um aliado na luta contra o software restrito. A título de exemplo, em janeiro de 2011, ambas organizações trabalharam juntas em defesa do software livre. Elas assinaram conjuntamente um comunicado para o departamento de justiça americano[10] se opondo à operação de venda do portifólio de patentes da Novell para a *CPTN Holdings*.

A pluralidade de ideias e concorrência natural entre os sistemas e aplicações dentro do movimento software livre fazem parte de seu mecanismo de evolução, bem como influencia positivamente em sua qualidade. A concorrência entre os navegadores, ferramentas de escritório, gerenciador de janelas e banco de dados são os exemplos mais conhecidos. Do restante da história do software livre até os dias atuais, podemos encontrar uma grande quantidade de soluções de alta qualidade que foram e estão sendo liberadas sob licença livre, em geral apoiadas tanto pela OSI quanto pela FSF. Com este breve resumo da história do software livre, mostramos que ele, como movimento e como solução tecnológica de alta qualidade e segurança, já é realidade há mais de duas décadas.

Linha do Tempo

1997: Eric Raymond apresenta a palestra "A catedral e o bazar"; Miguel de Icaza anuncia o projeto GNOME, um concorrente do KDE que nasce como resposta da FSF a problemas de licença com o KDE.

1998: Netscape libera o código-fonte do navegador Mozilla sob licença livre; Eric Raymond cria o movimento *Open Source* e cria a OSI; Corel anuncia o NetWinder, uma rede de computador baseada em Linux; Sun Microsystems e Adaptec são as primeiras grandes empresas a fazerem parte da Linux International; IBM anuncia que irá comercializar e dar suporte à Apache; Debian GNU/Linux 2.0, como trabalho de mais de 300 voluntários; KDE 1.0 é lançado e incorporado a várias distribuições GNU/Linux; Linus é capa da revista *Forbes*, representando o reconhecimento do mundo corporativo ao Linux e ao software livre; Microsoft reconhece o GNU/Linux e o software livre como importantes concorrentes e descreve como atacá-los, de acordo com os *Halloween Documents*.

1999: Dell, HP e SGI anunciam que irão dar suporte a GNU/Linux em seus computadores; GNOME 1.0 é lançado; Red Hat Software compra a Cygnus e se torna a maior empresa do mundo na área de software livre; Criado o SourceForge.net, maior repositório de projetos de software livre.

2000: Mozilla M13, o primeiro considerado estável, é lançado; Criada a fundação GNOME; A Sun libera o código-fonte do seu StartOffice sob licença LGPL e é criado o projeto OpenOffice.org.

2001: Linux 2.4 é lançado; IBM anuncia investimento de US$1bi no Linux; Surge a Wikipedia.

2002: Consórcio ObjectWeb é fundado pela Bull, France Telecom e INRIA, na França; KDE 3.0 e GNOME 2.0 são lançados e os gerenciadores de janelas chegam ao nível de concorrer com os *desktops* comerciais; Mozilla 1.0, o primeiro oficial estável, é disponibilizado; OpenOffice.org 1.0 é lançado; As primeiras licenças Creative Commons para compartilhamento de outros tipos de obras intelectuais (que não software) são publicadas.

[10] `www.fsf.org/news/osi-fsf-joint-position-cptn`

2003: Motorola inicia vendas do A760, o primeiro telefone celular com um Linux; Mozilla Foundation é criada; Lançado o Fedora Core, versão comunitária do Red Hat Linux; SCO processa IBM por suposto código de sua propriedade inserido no Linux, levantando dúvidas no mercado sobre a legitimidade do software livre e da licença GPL, mas perde.

2004: Novell compra da SuSE por 210 milhões de dólares; Primeira versão do Ubuntu e do Firefox são lançadas;

2005: MandrakeSoft compra a empresa brasileira Conectiva e a americana Lycoris, resultando na Mandriva; ODF (*open document format*), usado pelo OpenOffice 2.0, é reconhecido como padrão pela OASIS; Sun Microsystems disponibiliza o Open Solaris, versão livre do sistema operacional Solaris; Nicholas Negroponte anuncia o projeto OLPC (*One Laptop Per Child*), projeto do MIT tendo como base o Linux.

2006: Primeiro protótipo do XO (*One Laptop per Child*) é disponibilizado; Sun libera a máquina virtual Java sob licença GPL (alterada para permitir que se distribua em aplicações comerciais); OpenDocument Format (ODF), padronizado pela OASIS ODF TC, se torna um padrão ISO; Firefox atinge 200 milhões de downloads (12% do mercado mundial e 20% do Europeu); Neo1973, usando a plataforma OpenMoko para telefones móveis, é apresentado.

2007: A Sun libera também a JDK sob GPLv2; Após longas controvérsias, FSF lança a versão definitiva da GPL versão 3; O estudo FLOSSImpact sobre o efeito (especialmente econômico) do software livre, financiado pela Comissão Europeia, é publicado (o primeiro estudo de larga escala sobre o assunto).

2008: Nokia compra TrollTech, dona da biblioteca multiplataforma e livre Qt, e anuncia transformação do Symbian em software livre.

2009: Oracle compra Sun Microsystems por US$ 7,4 bilhões e entra definivamente no mercado de software livre, inclusive adquirindo o MySQL que havia sido comprado pela Sun.

2010: Oracle fecha escritórios da Sun na América Latina e não mantém a mesma relação com a comunidade software livre. A comunidade do Java ligada ao movimento do software livre volta a discutir a preferência e investimentos no OpenJDK.

2011: Linus Torvalds disponibiliza a versão 2.6.38 kernel do Linux. Entre as novidades, uma modificação no gerenciamento de processos (o *wonder patch*), que permite um melhor desempenho aos ambientes *Desktop*.

2011: OSI define uma nova política de estruturação interna que prevê a participação direta da comunidade em sua governança. Anuncia-se que os membros do corpo diretivo da OSI serão agora eleitos pelas instituições associadas a OSI e pelos grupos de trabalho da comunidade.

8.2.3.6. Software livre no Brasil

Assim como na "história geral" sobre o software livre, em que temos a participação fundamental da Universidade de Berkeley e do MIT, a história do movimento do software livre no Brasil pode ser traçada, a partir do início da década de 1990, com a chegada e instalação do GNU/Linux em departamentos de Ciência da Computação de universidades de ponta. A título de exemplo, podemos citar o caso do Instituto de Matemática e Estatística da Universidade de São Paulo, que já hospedava um curso de Ciência da Computação desde 1972, fundado pelos pioneiros Imre Simon e Valdemar Setzer. Em 1993, o professor do IME-USP, Marco Dimas Gubitoso (Gubi), foi primeiro a instalar o GNU/Linux na USP, possivelmente o primeiro em uma universidade brasileira. Gubi é o primeiro brasileiro a se cadastrar no Linux Counter [11] (número 2393). Portanto, vamos

[11] Linux Conter: `www.counter.li.org`

começar a contar a história do GNU/Linux e do software livre a partir desse ponto.

A chegada do GNU/Linux à USP, ocorreu quando o também professor da USP, Arnaldo Mandel, baixou o código-fonte do GNU/Linux e deixou em um dos servidores do departamento para quem quisesse experimentar. Na época, os professores e pesquisadores do IME-USP utilizavam o sistema SunOS, posteriormente chamado de Solaris, sistema restrito, nas estações de trabalho Sparc da Sun Microsystems. A distribuição instalada por Gubi foi a SoftLandinga (SLS), precursora da Slackware, usando disquetes de 3.5", em um 386sx com 4MB de RAM e 40MB de disco. Depois disso, ele enviou um e-mail para algumas pessoas do departamento contando a sua experiência com o GNU/Linux. Após seu relato, alguns professores da USP aderiram ao GNU/Linux, entre eles o professor Imre Simon, um dos pioneiros e um dos mais importantes líderes na área de ciência da computação no país. Imre tornou-se um grande divulgador e incentivador de formas abertas de compartilhamento e produção de conhecimento. Percorreu o país proferindo palestras e ministrando cursos e fundou a incubadora de conteúdos da FAPESP.

Ainda no primeiro semestre de 1993, um dos orientados de mestrado de Arnaldo Mandel, Fabio Kon, decide fazer seu mestrado na área de sistemas de arquivos distribuídos. Para tanto, desenvolve seu protótipo de uma extensão do NFS utilizando *Leases* fazendo alterações tanto no núcleo do Linux quanto no seu servidor NFS, tornando-se possivelmente o primeiro brasileiro a programar dentro do *kernel* do Linux.

Gubi começou a divulgar o GNU/Linux para os alunos de ciência da computação, dizendo que ele era uma versão do famoso sistema Unix que finalmente poderia ser utilizado em larga escala dentro das universidades brasileiras. Posteriormente, o IME-USP chegou a importar vários CDs de instalação para distribuir no instituto. Essa promoção e utilização do GNU/Linux dentro do IME-USP, motivou, em 1994, dois alunos de ciência da computação da USP, Félix Almeida e Adriano Rodrigues, a montarem um grupo de usuários GNU/Linux. Em 1995, eles fundaram a rede Linux do IME[12]. Provavelmente, a primeira dentro de uma universidade brasileira, e que até hoje está em funcionamento, ainda sendo administrada por alunos do IME-USP. A cultura do software livre na USP fez com que vários projetos criados fossem liberados sob licenças livres nos anos seguintes. Como uma consequência desse processo, o Departamento de Ciência da Computação aprovou oficialmente a criação do Centro de Competência em Software Livre da USP, em 2008. O objetivo do CCSL-USP[13] é incentivar a pesquisa, o ensino, o desenvolvimento e o uso do software livre/aberto dentro e fora da universidade.

O marco da expansão do software livre no Brasil é a fundação da Conectiva, no estado do Paraná, que foi a primeira empresa brasileira a comercializar e oferecer suporte para uma versão em português do Brasil de uma distribuição Linux. Uma década depois, a Conectiva foi comprada pela francesa MandrakeSoft, tornando-se a Mandriva. A outra parte dessa história, muito importante, foi quando funcionários de órgãos públicos de tecnologia da informação do estado do Rio Grande do Sul, alguns deles também ligados aos movimentos sindicais daquele estado, transbordaram a questão do software livre do meio técnico e fundaram o Projeto Software Livre do Rio Grande do Sul e Brasil em 1999. O mesmo grupo, em 2003, fundou a Associação Software Livre (ASL.org), uma ONG

[12]Rede Linux IME-USP: `www.linux.ime.usp.br`

[13]Centro de Competência da USP: `ccsl.usp.br`

para a promoção do software livre no Brasil. Dessa forma, o movimento do software livre no Brasil começou a crescer, em especial, impulsionado pelas edições do Fórum Internacional de Software Livre (FISL), que, desde 2000, ocorre todos os anos na cidade de Porto alegre. O FISL trouxe ao Brasil os grandes protagonistas da história do software livre, entre eles, Richard Stallman, Eric Raymond e John "MadDog" Hall (presidente da *Linux International*).

A partir do início dos anos 2000, o Brasil passou a ter uma papel fundamental no desenvolvimento, disseminação e como caso de uso e adoção de software livre no mundo. Em 2001, o brasileiro Marcelo Tosatti, com 18 anos, desenvolvedor da Conectiva, foi escolhido por Linus Torvalds como mantenedor oficial da versão 2.4 do núcleo do Linux. Em 2003, a Presidência da República publica um decreto que institui comitês técnicos para a adoção do software livre em todo os órgãos/instituições do Governo Brasileiro. Isso, somando ao fato do então presidente Luiz Inácio Lula da Silva e membros do seu governo, como o então presidente do Instituto Nacional de Tecnologia da Informação, Sérgio Amadeu da Silveira, adotarem uma forte posição a favor do software livre voltou os olhos da comunidade mundial para o Brasil.

Mais tarde, a partir de 2006, o Brasil torna-se um dos protagonistas na implementação de projetos pilotos para uso do OLPC, do MIT. Entretanto, até hoje, os computadores XO não foram adotados em larga escala no Brasil por uma questão de interferência de interesse de empresas, que vislumbram no projeto OLPC seus interesses comerciais. Em 2007, o *middleware* do Sistema Brasileiro de TV Digital, Ginga, desenvolvido pela equipe de Luiz Fernando Gomes Soares da PUC-Rio e de Guido Lemos da UFPB é liberado sob licença GPL. Para muitos da comunidade brasileira, naquele momento, o Ginga torna-se o principal software livre desenvolvido no Brasil. Em 2009, a linguagem NCL e o Ginga-NCL, criadas para oferecer interatividade em sistemas de TV Digital, foram aprovados como padrão pela União Internacional de Telecomunicações (UIT), órgão de padronização e regulamentação em telecomunicações ligado às Nações Unidas. Em outubro de 2008, com a liderança do CCSL-USP e do projeto europeu QualiPSo, é fundada a rede internacional de centros de competência em software livre[14] e o CCSL-USP é o primeiro centro a integrar essa rede. A rede lança o *Manifesto for FLOSS Competence Centers*[15].

Anualmente, desde 2008, o governo federal, através do SERPRO, promove em Brasília o Congresso Internacional Software Livre e Governo Eletrônico (CONSEGI) reunindo milhares de profissionais de órgãos públicos e privados, políticos, cientistas, educadores e estudantes para vários dias de palestras, workshops, cursos, mesas redondas e outros eventos sobre software livre e assuntos relacionados.

Por fim, outro simbólico acontecimento para a comunidade software livre brasileira, ocorre no FISL de 2009, quando o presidente Lula, acompanhado da então Ministra-Chefe da Casa Civil, Dilma Russeff, torna-se o primeiro presidente do mundo a visitar e discursar em um evento de uma comunidade de software livre. Lula e Dilma exaltaram a escolha e as ações do governo em prol do software livre, bem como reafirmaram os compromissos das políticas a favor do software e da cultura livre. Isso deu um grande

[14] `www.flosscc.org`
[15] `www.flosscc.org/manifesto`

ânimo e euforia para a comunidade brasileira iniciar uma nova década de transformação no país, através do software livre.

Linha do Tempo

1993: GNU/Linux é instalado e adotado por alguns professores da USP tanto como ferramenta de uso diário quanto como objeto de pesquisa.

1995: Criado a rede GNU/Linux do IME-USP; Nasce a Conectiva.

1999: Surge o Projeto Software Livre Brasil (PSL-BR); O Deputado Federal do estado da Bahia, Walter Pinheiro, cria um projeto que trata a obrigatoriedade dos órgãos da administração publica brasileiros a utilizarem software com código aberto.

2000: Acontece o primeiro Fórum Internacional de Software Livre (FISL), tendo Richard Stallman como principal convidado; Participantes nacionais incluem nomes como Imre Simon e Sérgio Amadeu; Dataprev libera o Cacic, primeiro software livre produzido pelo governo brasileiro.

2001: Marcelo Tosatti é escolhido para ser o mantenedor do núcleo do Linux.

2003: Governo brasileiro publica decreto em prol da adoção do software livre em todo os órgãos do governo brasileiro; Criada a ONG ASL.org.

2005: CCSL-IME-USP (Centro de Competência em Software Livre do IME-USP) é aprovado como projeto apoiado pela FINEP e USP anuncia sua criação.

2006: Corretor Gramatical CoGrOO é lançado e ajuda o *OpenOffice.org Writer* na briga contra o *Microsoft Word* no Brasil; Pilotos do projeto OLPC começam a ser realizados no Brasil; Fundada a cooperativa de tecnologias livres (Colivre), na Bahia, que mais tarde criaria o software livre para redes sociais e economia solidária – Noosfero.

2007: Linguagem NCL e ambiente Ginga-NCL são liberado também sob licença GPL; Criado o Portal do Software Público Brasileiro.

2008: Governo Federal define a contratação e uso de Software Livre via instrução normativa.

2009: O CCSL-USP é admitido como um dos centros do projeto QualiPSo; O Projeto Software Livre Brasil lança sua rede social, usando a plataforma brasileira Noosfero; Presidente Lula e Dilma Rousseff, Ministra-Chefe da Casa Civil, fazem discursos históricos apoiando e defendendo o software livre no Brasil.

8.2.4. Definição de Software Livre segundo a FSF e a OSI: as 4 liberdades

As duas principais organizações internacionais responsáveis pela proteção e promoção do software livre, a *Free Software Foundation* (FSF) e a *Open Source Initiative* (OSI), atuam também para garantir que os termos *Free Software* e *Open Source* sejam utilizados de forma correta. O objetivo é evitar que empresas ou grupos usem essas expressões de forma indevida como ferramenta de marketing, dizendo que um software é livre quando na verdade não é.

A *Free Software Foundation* define[16] que um programa de computador é software livre se e somente se ele oferece aos usuários as quatro seguintes liberdades[17]:

[16] Definição de *Free Software*: `www.gnu.org/philosophy/free-sw.html`

[17] Em vários sistemas de computação, é conveniente iniciar a contagem de itens com zero e não um; Numa referência a isso, as quatro liberdades foram definidas pela FSF como sendo as liberdades 0, 1, 2 e 3.

0. Liberdade para executar o programa
1. Liberdade para estudar e modificar o programa.
2. Liberdade para redistribuir o programa.
3. Liberdade para melhorar e redistribuir as melhorias ao programa.

Em particular, para que essas quatro liberdades sejam satisfeitas, é necessário que o programa seja distribuído juntamente com o seu código-fonte e que não sejam colocados empencilhos para que os usuários alterem e redistribuam esse código.

Já a *Open Source Initiative* (OSI), por conta da ambiguidade da palavra "free" em inglês, prefere a expressão *Open Source*, que em língua portuguesa é costumeiramente traduzida por software de código aberto, software aberto ou software livre. Segundo a OSI, para que um software seja considerado de código aberto, não basta apenas que seu código-fonte seja disponível; Ele precisa satisfazer, na forma em que for distribuído para uso, as dez condições seguintes[18], inspiradas nas *Orientações sobre Software Livre*[19] do projeto Debian:

Livre Redistribuição. Sua licença não pode restringir ninguém, proibindo que se venda ou doe o software a terceiros. A licença não pode exigir que se cobre o pagamento de *royalties* ou outros valores, embora tal pagamento não seja proibido.

Código-Fonte. O programa precisa obrigatoriamente incluir código-fonte e permitir a distribuição tanto do código-fonte quanto do programa já compilado. Quando o produto não é distribuído junto com seu código-fonte, é necessário que uma forma de se obter o seu código-fonte seja anunciada publicamente de uma forma fácil de se obter, preferivemente através de *download* gratuito da Internet.

Obras Derivadas. A licença deve permitir modificações e obras derivadas e deve permitir que essas modificações sejam redistribuídas dentro dos mesmos termos da licença original.

Integridade do Código do Autor. A licença pode proibir que se distribua o código-fonte original modificado desde que, neste caso, a licença permita a distribuição de arquivos de diferenças (*patch files*) contendo o código-fonte que foi modificado. A licença deve então explicitamente permitir a distribuição do software construído através das modificações do código-fonte original. A licença pode exigir que esses trabalhos derivados usem um nome ou número de versão diferente do software original.

Não Discriminação Contra Pessoas ou Grupos. A licença não pode discriminar contra pessoas ou grupos. Por exemplo, a licença não pode proibir que um programa seja distribuído para pessoas residentes em um determinado país.

[18] Definição de *Open Source*: `www.opensource.org/docs/osd`.
[19] *Debian Free Software Guidelines*: `www.debian.org/social_contract`.

Não Discriminação Contra Áreas de Utilização. A licença não pode restringir os usuários de fazer uso do programa numa área específica. Por exemplo, não pode proibir que o programa seja usado para fins comerciais, ou para fins militares, por mais nobre que esta última possa ser.

Distribuição da Licença. Os direitos associados ao programa através da licença são automaticamente repassados a todas as pessoas às quais o programa é redistribuído sem a necessidade de definição ou aceitação de uma nova licença.

Licença Não Pode Ser Específica a um Produto. Os direitos associados a um programa não dependem de qual distribuição em particular aquele programa está inserido. Se o programa é retirado de uma distribuição, os direitos garantidos por sua licença continuam valendo.

Licenças Não Podem Restringir Outro Software. A licença não pode colocar restrições em relação a outros programas que sejam distribuídos junto com o software em questão. Por exemplo, a licença não pode exigir que todos os outros programas distribuídos no mesmo pacote sejam também software aberto.

Licenças Devem Ser Neutras em Relação a Tecnologias. Nenhuma exigência da licença pode ser específica a uma determinada tecnologia ou estilo de interface.

Dessa forma, vemos que enquanto a definição de software livre da FSF concentra-se prioritariamente na questão da liberdade do usuário, a definição de Software Aberto da OSI abarca as mesmas características mas incluem algumas restrições adicionais focadas no modelo corporativo e em negócios comerciais montados em torno do software. No entanto, para fins práticos, o resultado de ambos os conjuntos de restrições é praticamente igual; O conjunto de licenças aprovadas pela FSF e pela OSI é quase idêntico e, portanto, em termos pragmáticos, podemos considerar que o movimento pelo software livre e a iniciativa pelo software aberto se preocupam com o mesmo software, apenas com pontos de vista diferentes. De forma similar, os termos “free software”, “software livre”, “open source” e “software aberto” são, ao menos quando usados nesse contexto, sinônimos; a escolha por um ou outro é questão de preferência e identificação com um ou outro desses grupos. Dado que em português não há os problemas semânticos associados à expressão “free software” em inglês, neste capítulo, utilizamos o termo software livre para nos referir ao software distribuído com qualquer licença aprovada pela OSI ou pela FSF.

8.2.5. Comunidades Envolvidas no Movimento do Software Livre

O movimento pelo software livre envolve um grande número de pessoas, instituições e empresas. Cada uma delas encara e se envolve com o software livre à sua maneira; No entanto, é possível identificar alguns grupos com características razoavelmente similares.

Provavelmente a instituição mais antiga com vínculo formal com o software livre é a *Free Software Foundation* (Fundação para o Software Livre), tendo sido criada em 1985 a partir da iniciativa de Richard Stallman. A FSF é responsável por boa parte do software livre tipicamente usado em distribuições linux, graças ao seu projeto inicial de desenvolver um sistema operacional completo e livre.

A FSF, juntamente com vários membros da comunidade de software livre, considera que a principal característica do software livre é o respeito à liberdade do usuário em usar e redistribuir o programa. Para essa fatia da comunidade, softwares restritos devem ser evitados por princípio, mesmo que ofereçam vantagens como melhor funcionalidade.

Embora a FSF seja a favor de esforços comerciais baseados no software livre, ela é incondicionalmente contra ações que sejam contrárias ao espírito de liberdade do software livre. Essa postura às vezes considerada inflexível da FSF é muitas vezes criticada, tanto dentro quanto fora da comunidade[20]. Por outro lado, a forte preocupação da FSF com a integridade ética e legal do software livre tornou-a uma referência importante para usuários e desenvolvedores a respeito desses tópicos em relação ao software livre. Hoje esse papel tem sido em parte cumprido pelo *Software Freedom Law Center*[21].

Com outra proposta, a *Open Source Iniciative* (Iniciativa pelo Software Aberto) foi criada para incentivar uma aproximação de entidades comerciais com o software livre. Sua atuação principal é a de certificar quais licenças de software se enquadram nas restrições listadas anteriormente e, portanto, são licenças de software livre, além de promover a divulgação do software livre e suas vantagens tecnológicas e econômicas.

A OSI, assim como muitos membros da comunidade, considera que o software é, em primeiro lugar, uma ferramenta, e que o mérito dessa ferramenta deve ser julgado com base em critérios técnicos. Para eles, o software livre no longo prazo é economicamente mais eficiente e de melhor qualidade e, por isso, deve ser incentivado. Além disso, a participação de empresas no ecossistema do software livre é considerada fundamental, pois são as empresas que viabilizam o aumento no desenvolvimento, implantação e uso do software livre.

Como consequência, a OSI não se opõe ao uso de software restrito e não é contrária a soluções híbridas, consistindo em sistemas de software onde apenas uma parte do código é livre. Ela atua como um órgão regulador, certificando as licenças de software livre, e divulgando as vantagens técnicas do software livre, defendendo uma visão pragmática do tema.

Outros membros da comunidade concordam com a OSI em que o software é uma ferramenta, mas não consideram que o software livre tenha alguma vantagem intrínseca. Para esse grupo, o software livre é apenas mais uma opção a se avaliar ao escolher um novo software.

Além desses grupos, definidos com base na sua maneira de encarar o software livre, outras entidades importantes participam do ecossistema do software livre.

Uma delas é a fundação apache[22], que consiste em uma grande comunidade de desenvolvimento responsável por vários sistemas de software livre bastante difundidos e de alta qualidade. Várias empresas oferecem apoio à fundação apache, seja através de doações, seja através do empenho de alguns de seus funcionários em projetos da fundação.

[20] "The conferees decided it was time to dump the moralizing and confrontational attitude that had been associated with "free software" in the past[...]" [The Open Source Initiative - OSI 1999].

[21] `http://www.softwarefreedom.org`

[22] `http://www.apache.org`

A fundação eclipse[23] também consiste em uma comunidade de desenvolvimento responsável por diversos projetos de alta qualidade. Diferentemente da fundação apache, no entanto, ela surgiu como um consórcio de empresas, e hoje possui como membros oficiais empresas e pessoas que colaboram financeiramente para o bom andamento de seus trabalhos.

No Brasil, a Associação Software Livre[24] também reúne empresas e pessoas com o objetivo de divulgar o software livre, com diversas frentes de atuação. Em particular, ela é responsável pela organização do Fórum Internacional de Software Livre (FISL)[25], um dos maiores eventos sobre software livre do mundo.

Finalmente, é importante lembrar que o software livre é baseado na ideia de compartilhamento do conhecimento. Assim, é natural que existam sinergias entre a comunidade do software livre e as comunidades que giram em torno da família de licenças *Creative Commons*[26] e da *Wikipedia*[27].

8.3. Desenvolvimento de Software Livre

O desenvolvimento de software livre possui características distintas do modelo restrito; A relação com o mercado, o processo de desenvolvimento e o produto a ser oferecido devem, portanto, ser abordados de maneiras distintas.

8.3.1. *A Catedral e o Bazar* e o Movimento pelo Código Aberto

No artigo *A Catedral e o Bazar* [Raymond 1997], Eric Raymond levanta os aspectos que contribuem para que um projeto de software livre tenha sucesso; Suas observações formaram a base do movimento pelo código aberto (*open source movement*) por ele iniciado na virada do século. Nesse artigo, Raymond identifica que alguns dos primeiros projetos de software livre de sucesso, como o núcleo do Emacs desenvolvido por Richard Stallman, por exemplo, utilizavam um modelo semelhante àquele utilizado para a construção de catedrais góticas e também usado largamente no desenvolvimento de software restrito. Esse software se prestava a essa comparação porque era habilmente criado com extremo cuidado por arquitetos altamente qualificados ou por pequenos grupos desses "magos" trabalhando em esplêndido isolamento, com nenhuma versão *beta* para ser liberada antes de seu tempo.

No entanto, ao observar o modelo de desenvolvimento do Linux, Raymond vislumbrou um mundo completamente diferente, mais semelhante a um barulhento Bazar, onde centenas ou milhares de desenvolvedores davam a sua contribuição que era, então, gerenciada por um pequeno grupo ou por um "ditador benevolente" que dava orientações sobre a qualidade das modificações propostas e as aceitava ou não. Ao identificar essas características, Raymond aplicou essas práticas de desenvolvimento do modelo Bazar de forma consciente em um projeto sob sua coordenação, o Fetchmail. O sucesso nessa empreitada o motivou a escrever o artigo, onde ele elenca as principais características

[23] `http://www.eclipse.org`
[24] `http://www.asl.org.br`
[25] `http://fisl.softwarelivre.org`
[26] `http://www.creativecommons.org`
[27] `http://www.wikipedia.org`

desse modelo, típicas em boa parte dos projetos de software livre da atualidade, como sendo:

- *Bons programas nascem de necessidades pessoais.* Um projeto tem maiores chances de sucesso quando o desenvolvedor principal ou grupo de desenvolvedores principais tem interesse e sentem a necessidade pessoal de utilizar aquele software.
- *Bons programadores sabem escrever bom código; Excelentes programadores sabem reescrever e reutilizar código.* Raymond menciona a "preguiça construtiva" como a ideia de que não se deve reinventar a roda, mas sim reaproveitar o que já existe e, se for o caso, modificar o que já existe para melhorá-lo.
- *Esteja preparado para jogar fora código-fonte se necessário e começar de novo.* Dificilmente se vai acertar na primeira vez.
- *Os usuários devem ser tratados como co-desenvolvedores.* Esse é o melhor caminho para o aprimoramento do código e depuração eficaz.
- *Libere código cedo e libere frequentemente; E ouça seus usuários.* Um erro comum de pessoas e grupos que se iniciam no mundo do software livre é achar que seu software ainda não está pronto para ser liberado, que agora ainda não é o momento certo para se fazer isso. Segundo Raymond, o quanto antes o código for liberado e quanto maior a frequência de liberação de novas versões, melhor será o retorno obtido dos usuários e a possibilidade de angariar contribuidores para o projeto. Não era incomum para Linus Torvalds, no início de 1991, liberar mais de uma versão do núcleo do Linux por dia! Isso possibilitou, naquela época, um grau de energia e motivação para o desenvolvimento colaborativo de software de forma distribuída nunca antes vista na história da Computação.
- *Dados olhos suficientes, todos erros são triviais.* Raymond chamou essa frase de Lei de Linus. Com milhares de pessoas lendo o código-fonte do Linux, os eventuais erros eram localizados e reportados muito rapidamente. Da mesma forma, com centenas de pessoas com conhecimento técnico para resolver aqueles erros, rapidamente aparecia um voluntário com a solução do problema.
- *Trate seus testadores das versões Beta como um recurso valioso e eles logo tornar-se-ão um recurso valioso.* Não há nada mais eficaz para encontrar problemas num programa e sugerir melhorias em suas funcionalidades do que um grupo de usuários ativos e motivados querendo utilizar esse programa e testar as novas funcionalidades o quanto antes.
- *A perfeição (em projetar) é alcançada não quando não há mais nada a adicionar, mas quando não há nada para jogar fora.* Essa é uma ideia quase que de consenso entre os grandes cientistas e engenheiros. Deve-se buscar sempre as soluções mais simples.
- *A melhor coisa depois de ter boas ideias é reconhecer as boas ideias de seus contribuidores.* Às vezes, essa última é melhor. Um bom líder de um projeto de

> software livre não é necessariamente aquele que tem ótimas ideias, mas sim aquele que é capaz de criar o ecossistema de colaboração que permita que as boas ideias emerjam e sejam valorizadas e adotadas.

Sete meses após a publicação do artigo de Eric Raymond, fortemente influenciada pelas ideias apresentadas, a Netscape Communications, Inc. anunciou seus planos de abrir o código do seu navegador Web. Essa iniciativa, posteriormente, levou ao desenvolvimento do Mozilla Firefox, utilizado hoje em dia por centenas de milhões de internautas em todo o globo.

8.3.2. Interação com a Comunidade

Graças às atividades de produção de código, documentação, relatos de defeitos entre outras, as comunidades de software livre vêm construindo coletivamente sistemas de software reconhecidamente de qualidade, em um ambiente de colaboração constante para atualização e evolução desses sistemas, organizados na forma de um rossio [Simon e Vieira 2008]. Nesse contexto, os usuários não necessariamente restringem-se a ser apenas agentes passivos, mas podem atuar como colaboradores ou produtores do software que usam.

Esse fenômeno de produção coletiva extrapolou o movimento do software livre, especialmente a partir da primeira década do século 21, com o surgimento de serviços criados e mantidos pelos próprios usuários na Internet, como a Wikipedia, YouTube, blogs pessoais, TVs e rádios online [Tapscott e Williams 2006]. Tais serviços, somados às redes sociais como Facebook, Orkut e Twitter, fazem com que as pessoas realmente acreditem que podem influenciar outras através de seus próprios meios de comunicação [Castells 2006]. Esse cenário, em que não fica clara uma diferenciação entre consumidor e produtor de informação e, no caso do software, usuário e desenvolvedor, pode ser chamado de *cultura livre*.

No caso do software livre, essa cultura tipicamente se pauta fortemente nas questões técnicas e se organiza como uma meritocracia, onde se valoriza fortemente as colaborações feitas para o projeto. No entanto, as questões éticas, quando se apresentam, em geral são consideradas tão importantes quanto as técnicas, já que o envolvimento com um projeto é determinado pelo interesse pessoal. Pessoas e organizações que permanecem no longo prazo colaborando de alguma maneira com o desenvolvimento de um software livre realmente acreditam que estão fazendo a diferença e ajudando o mundo de alguma forma, e essa motivação faz com que sua dedicação seja diferenciada. Mesmo no caso (cada vez mais comum) de desenvolvedores pagos para trabalhar em um projeto livre, os aspectos éticos e o relacionamento com a comunidade norteiam sua participação.

Com modelos de negócio baseados na prestação serviços, muitas empresas desenvolvem ou colaboram com um software livre para melhor reaproveitar o conhecimento produzido coletivamente, bem como atingir numa escala maior seu mercado consumidor. A interação dessas empresas com uma comunidade de software livre já existente ou em formação em torno de um de seus produtos deve respeitar essa cultura, já que o desrespeito aos seus valores pode decretar o fracasso na exploração de seu potencial.

Além disso, o desenvolvimento efetivo de software livre em conjunto com colaboradores externos à empresa envolve administrar ou participar de uma equipe de desenvolvimento onde não há hierarquia formal, não há mecanismos de pressão para o cumprimento de prazos e não há grande formalismo em processos. Para que esse processo tenha sucesso, essas empresas devem orientar seus funcionários a seguir diversas práticas para atrair contribuições externas [Corbucci 2011].

Uma das estratégias centrais para atrair contribuidores e evitar que eles abandonem o projeto é garantir a qualidade do código (por exemplo, baixa complexidade estrutural e modularizado), o que favorece a criação de um círculo virtuoso em que o código promove o crescimento da comunidade e a comunidade ativa promove melhorias no código. Por outro lado, é importante observar também que pode haver alguns tipos de software que não têm apelo significativo junto à comunidade. Por exemplo, o volume de desenvolvedores e usuários interessados em sistemas livres similares a outros já existentes e disponíveis sob licenças restritivas mas de forma gratuita costuma ser pequeno. Isso pode ser observado em *drivers* de dispositivo, como os necessários para as placas de vídeo de alto desempenho da nVidia e ATI/AMD; em programas como o *plugin* flash; com bibliotecas, como a motif, a Qt e a bilbioteca padrão da linguagem Java (antes de suas licenças serem transformadas em software livre); etc.

Na busca por uma formalização sobre qual metodologia as empresas podem adotar para interagir melhor com as comunidades de software livre, estudos mostram que métodos ágeis e software livre têm formas de trabalhos semelhantes, e o desenvolvimento de software livre é considerado um método ágil por Martin Fowler [Fowler 2000]. Um relatório técnico sobre metodologias de desenvolvimento ágil concluiu que o desenvolvimento de software livre é um método ágil [Abrahamsson et al. 2002]. Os mesmos autores apontam fortes semelhanças entre métodos ágeis e software livre em outro estudo [Warsta e Abrahamsson 2003]. Mais recentemente, uma ampla pesquisa sobre as comunidades de desenvolvimento ágil e de software livre [Corbucci 2011] resultou em um mapeamento completo entre as práticas comuns usadas por essas comunidades. Conceitualmente, os valores semelhantes são:

- Indivíduos e interações são mais importantes que processos e ferramentas.
- Software em funcionamento é mais importante que documentação abrangente.
- Colaboração com o cliente (usuários) é mais importante que negociação de contratos.
- Responder às mudanças é mais importante que seguir um plano.

De acordo com esse mesmo estudo [Corbucci 2011], várias práticas disseminadas pelas metodologias ágeis são usadas no dia-a-dia dos desenvolvedores e equipes das comunidades de software livre:

- Código compartilhado (coletivo).
- Design simples.

- Repositório único de código (SVN, Git, Bazaar, Mercurial).
- Integração contínua.
- Código e teste.
- Desenvolvimento dirigido por testes (TDD).
- Refatoração.

A preocupação com esses aspectos é importante porque muitos projetos de software livre não vão além dos estágios iniciais de planejamento, e muitos acabam sendo abandonados antes de produzir resultados razoáveis. Isso sugere que, mesmo com o grande sucesso do software livre, as comunidades, com ou sem a participação de empresas, podem melhorar suas formas de organização.

Para ilustrar esse cenário, podemos observar alguns dados que extraímos, em novembro de 2009, do SourceForge.net, um dos mais populares repositórios de projetos de software livre. Entre os seus 201.494 projetos cadastrados, 60.642 lançaram mais de uma versão, 40.228 foram baixados mais de uma vez, 23.754 têm mais de um membro, e apenas 12.141 projetos satisfazem esses três critérios de seleção juntos. Isso sugere que não mais que 6% dos projetos no SourceForge.net foram capazes de constituir uma comunidade de usuários e desenvolvedores que se beneficiem do estilo de desenvolvimento Bazar [Raymond 1997].

Se, de um lado, pode-se dizer que muitas iniciativas falham, por outro pode-se ver que há grande disposição para a criação de projetos de software livre. Assim, há muito espaço para que as empresas envolvam essas pessoas em projetos de forma que empresas e comunidades juntas possam se beneficiar dessa colaboração e do modelo do software livre.

8.4. Principais Licenças de Software Livre

Software livre, em geral, é de fácil acesso; Porém, a simples obtenção de um programa não significa que se possa fazer o que quiser com ele. Na ausência de licença específica, as leis de direito autoral impõem várias restrições para diversas formas de utilização da obra, tais como cópia, reprodução, etc. Por isso, essas utilizações dependem, para sua legalidade, de autorização prévia e expressa do autor, que pela mesma lei tem autonomia para concedê-las. A licença de um software livre é o documento através do qual os detentores dos direitos sobre o programa de computador autorizam usos de sua obra que, de outra forma, estariam vedados ou limitados pelas leis vigentes no local de seu uso ou produção. Esses usos autorizados permitem que desenvolvedores possam adaptar o software para necessidades mais específicas, utilizá-lo como fundação para construção de programas mais complexos, entre diversas outras possibilidades.

Há inúmeras possibilidades para redigir o texto de uma licença de software livre, mas a prática mais comum e recomendada é reaproveitar alguma das licenças já consolidadas na comunidade. Dessa forma, reduz-se a proliferação de licenças, que gera trabalho adicional para os usuários, uma vez que torna-se necessário para eles estudar

os termos de cada nova licença presente no software que irão utilizar. Apresentaremos aqui algumas das principais licenças utilizadas atualmente pela comunidade, visando seu melhor entendimento e aplicação.

Uma das vantagens do software livre é a possibilidade de reutilização do código em diferentes contextos, eventualmente através da combinação de trechos de código-fonte desenvolvidos independentemente. No entanto, diferentes sistemas de software podem estar sujeitos a diferentes licenças, e uma combinação envolvendo código dos dois precisa atender às restrições das duas. Assim, uma das questões mais importantes em relação ao licenciamento do software livre diz respeito à compatibilidade entre licenças.

Para simplificar essa discussão, iremos classificar as licenças de acordo com a presença ou não de termos que impõem restrições de relicenciamento quando da redistribuição do trabalho, da criação de trabalhos dele derivados ou da distribuição ou redistribuição destes. Assim, em relação a essa característica, consideramos as licenças *permissivas* ou *recíprocas*. No caso das recíprocas, é conveniente observar que algumas são mais abrangentes que outras no tocante aos casos em que as restrições se aplicam. Consequentemente, as dividimos entre recíprocas parciais, que também recebem a denominação de *copyleft* fraco, e recíprocas totais.

8.4.1. Permissivas

As licenças permissivas, também chamadas de licenças acadêmicas por alguns autores, como Rosen [Rosen 2005] e Laurent [Laurent 2004], em referência às origens das licenças BSD (*University of California, Berkeley*) e MIT (*Massachusetts Institute of Technology*), impõem poucas restrições às pessoas que obtém o produto. Muitas vezes, tais licenças são usadas em projetos de pesquisa de universidades, que servem como prova de conceito de alguma tecnologia que poderá ser explorada comercialmente no futuro. No caso das licenças permissivas, não é feita nenhuma restrição ao licenciamento de trabalhos derivados, podendo estes, inclusive, ser distribuídos sob uma licença restrita.

Licenças permissivas são uma ótima opção para projetos cujo objetivo é atingir o maior número de pessoas, não importando se na forma de software livre ou de software restrito. Temos vários exemplos desse fenômeno no BSD Unix, que continha o software de TCP/IP que hoje é usado na maior parte das implementações desse protocolo, incluindo a da Microsoft [Laurent 2004]. Outro exemplo é o BIND (*Berkeley Internet Name Daemon*), cuja implementação livre é até hoje usada nos principais servidores de DNS, apesar de haver também várias implementações restritas. Segundo Laurent, há bilhões de dólares em atividade econômica associada apenas com a pilha de software para Internet originalmente liberada sob a licença BSD.

Alguns argumentam que o uso desse tipo de licença não incentiva o modelo de software livre, pois empresas se aproveitam da comunidade para desenvolver software que será posteriormente tornado restrito. Porém, quando são usadas licenças permissivas, em geral os autores estão cientes dessa possibilidade e não veem isso como um problema. Um caso conhecido em que, de fato, os autores se arrependeram da licença que adotaram é o do Kerberos, desenvolvido no MIT, que posteriormente foi adotado pela Microsoft, que desenvolveu extensões restitas e incompatíveis para ele [Laurent 2004]. Mas o mais provável, caso a licença não permitisse isso, seria que a Microsoft adotasse algum outro

sistema de segurança e o Kerberos não se tornaria tão popular.

Por outro lado, em alguns casos, não é necessário que haja restrições na licença para garantir a continuidade do modelo de desenvolvimento livre, como no exemplo do servidor Apache. Duas características explicam o seu domínio no mercado: A marca forte, cujo uso é protegido pela própria licença do Apache, e a importância da conformidade com os padrões, que evita a disseminação de extensões restritas.

8.4.1.1. A Licença BSD

A BSD[28] foi a primeira licença de software livre escrita e até hoje é uma das mais usadas. Criada originalmente pela Universidade da Califórnia em Berkeley para seu sistema operacional derivado do UNIX (Berkeley Software Distribution), a licença BSD é usada como modelo por uma ampla gama de software licenciado de modo permissivo.

Os principais motivos que levaram a licença BSD a ser tão difundida são a simplicidade de seu texto e o fato dela ter sido inicialmente adotada por um projeto amplamente disseminado, o que criou um ciclo virtuoso em que mais comunidades a adotaram, tornando-a ainda mais reconhecida. Por outro lado, algumas mudanças significativas foram feitas no texto da licença ao longo do tempo, incluindo a remoção de cláusulas importantes, o que pode ocasionar dúvidas quanto ao significado exato da expressão "licença BSD".

A versão mais difundida dessa licença atualmente estabelece como únicas exigências que (a) o nome do autor original não seja utilizado em trabalhos derivados sem permissão específica por escrito, visando proteger sua reputação (dado que o autor pode não ter relação alguma com as modificações realizadas) e, (b) no caso de redistribuição do código-fonte ou binário, modificado ou não, é necessário que seja mencionado o copyright original e os termos da licença.

8.4.1.2. A Licença MIT/X11

A Licença MIT[29], criada pelo *Massachusetts Institute of Technology*, é também conhecida como Licença X11 ou X, por ter sido redigida para o X Window System, desenvolvido no MIT em 1987.

Essa também é uma licença permissiva e é considerada equivalente à licença BSD mencionada acima, porém sem a primeira restrição ("(a)"). Seu texto é bem mais explícito ao tratar dos direitos que estão sendo transferidos, afirmando que qualquer pessoa que obtém uma cópia do software e seus arquivos de documentação associados pode lidar com eles sem restrição, incluindo sem limitação os direitos de usar, copiar, modificar, mesclar, publicar, distribuir, sublicenciar e/ou vender cópias do software. As condições impostas para tanto são apenas manter o aviso de *copyright* e uma cópia da licença em todas as cópias ou porções substanciais do software.

[28] `www.opensource.org/licenses/bsd-license.php`
[29] `www.opensource.org/licenses/mit-license.php`

Essa licença é a recomendada pela *Free Software Foundation* quando se busca uma licença permissiva, pois é bastante conhecida e, ao contrário da BSD, não possui múltiplas versões com cláusulas que podem gerar dificuldades adicionais, tais como a cláusula de propaganda da BSD que pode gerar incompatibilidades com outras licenças.

8.4.1.3. A Licença Apache

A Licença Apache está atualmente na versão 2.0[30] e é usada por um dos projetos mais conhecidos de software livre, o servidor web Apache, assim como pela maior parte dos outros projetos da Fundação Apache, além de um grande número de projetos independentes que optaram por usar essa licença.

A Apache também é uma licença permissiva e, na versão 1.1, seu texto era bastante similar ao da BSD. Em 2004, a licença foi totalmente reescrita e seu texto ficou bem mais longo e complexo, detalhando melhor os direitos concedidos e suas condições. Essa é a principal vantagem da licença Apache, pois ao ter seus termos definidos de forma mais precisa deixa menos margem a interpretações conflitantes com os interesses dos envolvidos. Por outro lado, a licença é mais extensa e mais difícil de ser compreendida por leigos.

Para formalizar o processo de contribuições para os projetos da fundação, foi também criado o *Apache Contributor License Agreement* (Acordo de licenciamento do colaborador apache). Através dele, colaboradores externos à fundação transmitem à *Apache Software Foundation* todos os direitos de propriedade intelectual necessários para que a fundação possa licenciar suas contribuições, o que permite a ela modificar os termos de licenciamento de seus projetos no futuro.

8.4.2. Recíprocas Totais

As licenças recíprocas totais determinam que qualquer trabalho derivado precisa ser distribuído sob os mesmos termos da licença original. Isso também é chamado de *copyleft*[31], um termo criado pela *Free Software Foundation*. A licença que deu origem à ideia de *copyleft* foi a *GNU General Public License*, comumente chamada apenas de GPL[32], que será discutida adiante.

A ideia do *copyleft* é dar permissão a todos para executar, copiar, modificar e distribuir o programa, modificado ou não, mas impedir que sejam adicionadas restrições quando da redistribuição. Tal ideia visa fortalecer o software livre como um todo, não permitindo que melhorias do software sejam retiradas do alcance da comunidade. O resultado esperado é que a quantidade de software livre aumente cada vez mais, beneficiando todos os envolvidos na cadeia produtiva do software livre. Além disso, a reciprocidade contribui para manter a compatibilidade entre diversas versões de um determinado sistema, dado que quando novas funcionalidades são introduzidas de forma restrita fica mais difícil replicá-las em outras versões e derivações.

[30] `www.apache.org/licenses/LICENSE-2.0.html`
[31] `www.gnu.org/copyleft`
[32] `www.gnu.org/licenses/gpl.html`

Por outro lado, tal abordagem também sofre críticas de dentro da comunidade, pois o software licenciado nesse modelo acaba ficando, de certa forma, isolado dos demais, devido a incompatibilidades nas licenças. Na prática, software licenciado sob o modelo permissivo, em geral, pode ser incorporado por software licenciado como recíproco, já que licenças permissivas permitem a redistribuição sob outros termos, inclusive os de licenças recíprocas. Porém, o inverso não é verdadeiro e, assim, software disponibilizado sob licenças recíprocas não pode ser utilizado em projetos de software livre que usam alguma outra licença se esta não for reciprocamente compatível com aquela.

8.4.2.1. A licença GNU GPL

A licença GPL foi escrita em 1989[33] pela *Free Software Foundation*. Dois anos depois, em junho de 1991, foram feitas pequenas modificações na licença, gerando a versão 2.0[34]. Essa versão manteve-se até 2007, quando foi definida a GPLv3.

A GPL é uma das licenças mais utilizadas em projetos de software livre, e é recomendada para projetos que buscam seu crescimento através de contribuições de terceiros, dado que melhorias feitas ao software devem manter-se livres para poderem ser distribuídas. A GPL também é usada frequentemente no modelo comercial de licenciamento dual, discutido mais abaixo. Nesse caso, a empresa provê o software sob a licença GPL, obtendo os benefícios relacionados ao software livre, mas ao mesmo tempo disponibiliza o software também sob alguma outra licença que permite formas de exploração vedadas pela GPL.

Segundo a GPL, a cópia e distribuição do código-fonte do programa, com ou sem modificações, pode ser realizada desde que se mantenham os avisos sobre o *copyright*, a ausência de garantias e a licença. Já no caso da distribuição de um binário, é obrigatório que ele acompanhe o código-fonte ou instruções de como obtê-lo ao custo máximo do meio físico utilizado para transferi-lo. A licença explica ainda que é permitido exigir pagamento pelo ato de transferir uma cópia ou por garantias adicionais que a pessoa decida oferecer, o que permite o uso do software em um modelo de negócio comercial.

Consoante com os objetivos da FSF, a GPL não impõe nenhuma restrição sobre o uso do programa, mas apenas à sua redistribuição. Essas restrições têm o objetivo de garantir que o programa e seus trabalhos derivados sejam sempre redistribuídos sob os mesmos termos, garantindo a liberdade dos usuários. Assim, pode-se dizer que a GPL permite aos usuários fazer qualquer coisa com o programa, exceto impor restrições adicionais a outros usuários.

A licença se aplica a qualquer trabalho derivado, segundo a lei de direitos autorais. Porém, a GPL define trabalho derivado como "um trabalho contendo o programa ou parte dele, literalmente ou com modificações e/ou traduzido para outra língua" (GNU General Public License, version 2, 1991). Essa definição dada a trabalho derivado difere tanto da legislação americana quanto da brasileira. Segundo a definição dada pela GPL, mesmo trabalhos coletivos são considerados como trabalhos derivados. Tal definição é de

[33] `www.gnu.org/licenses/old-licenses/gpl-1.0.html`
[34] `www.gnu.org/licenses/old-licenses/gpl-2.0.html`

importância fundamental na aplicação da GPL, e em raras situações pode dar origem a controvérsias quanto à necessidade de reciprocidade em alguns usos atípicos do software.

A incompatibilidade de outras licenças com a GPL surge na cláusula que afirma que "Você não poderá impor aos recebedores qualquer outra restrição ao exercício dos direitos então adquiridos". Juntando isso às regras impostas na redistribuição, software coberto sob licenças que têm alguma restrição não pode ser combinados com software GPL.

O modelo de software livre proposto pela GPL é garantido mesmo na presença de decisões judiciais que iriam contra seus princípios. Está no texto que, caso não seja possível cumprir alguma decisão judicial ou lei local e ao mesmo tempo seguir os termos da licença, então não é permitido que o programa seja distribuído nessas circunstâncias ou nesse contexto.

A *Free Software Foundation* recomenda que o autor que usa a GPL permita que seu trabalho esteja licenciado sob a versão mais recente da licença ou qualquer versão posterior, de forma que quando surgir uma nova versão o usuário da licença possa escolher qual das versões utilizar. Dessa forma, evita-se incompatibilidade entre programas mais antigos e mais novos que optaram por utilizar a GPL. Porém, muitas pessoas preferem ter maior controle sobre quais são os termos em que seu software está licenciado e, assim, não deixam aberta essa possibilidade. Um exemplo importante dessa escolha é o núcleo do Linux.

8.4.2.2. A GPLv3

A mais nova versão da GPL[35], lançada em 29 de junho de 2007, após um longo período de discussão e revisão pública, foi criada para evitar algumas situações consideradas indesejáveis pela *Free Software Foundation*. Além disso, algumas partes foram reescritas de forma a adaptar a licença a novas formas de compartilhamento de programa e a deixá-la mais adequada a legislações em que os termos originais poderiam ser interpretados de forma diferente da esperada pela *Free Software Foundation*. Também foram realizadas alterações para facilitar a compatibilidade com outras licenças, em particular a Apache 2.0. A seguir serão discutidas as principais mudanças.

Um dos principais motivos para mudar para a GPLv3, segundo seus criadores, é evitar o fenômeno conhecido como *tivoização* (em referência ao aparelho TiVo, que funciona como um gravador digital de vídeos). O TiVo inclui software derivado do Linux, licenciado sob a GPL 2.0. O código está disponível e pode ser modificado; Porém, tais modificações não podem ser utilizadas no aparelho TiVo, pois ele faz uma checagem da assinatura digital do software e executa apenas as versões permitidas pelo fabricante. Essa questão de assinaturas digitais foi bastante controversa durante a elaboração da GPLv3, pois ao mesmo tempo em que as assinaturas restringem a liberdade dos usuários, defendida pela *Free Software Foundation*, elas são uma ferramenta importante para implementar segurança em alguns sistemas. A estratégia para impedir a *tivoização* é exigir que o fabricante provenha toda informação necessária para instalar versões modificadas

[35] `www.gnu.org/licenses/gpl-3.0-standalone.html`

do software no aparelho. Essa informação pode ir desde instruções básicas até chaves de autorização que possam ser necessárias. Porém, tal exigência é limitada a aparelhos considerados "produtos de usuário". Assim, alguns equipamentos de uso específico, como por exemplo máquinas de votação, estariam isentos da necessidade de reduzir seu nível de segurança para se adequar à licença.

Outra mudança na GPLv3 é relacionada a mecanismos de DRM, ou *Digital Rights Management*[36]. É sabido que a *Free Software Foundation* é contra o uso de DRM. Ainda assim, ela não quis impedir que software livre fosse utilizado para implementá-lo, já que isso limitaria a liberdade dos usuários. Como alternativa, decidiram limitar o impacto do tratado de *copyright* da *World Intellectual Property Organization* (WIPO), adotado em 20 de dezembro de 1996 sobre o assunto. Nesse tratado, o artigo 11, sobre obrigações a respeito de medidas tecnológicas, afirma que os países devem adotar medidas legais para reprimir tentativas de burlar sistemas tecnológicos de proteção usados por autores para restringir atos que não são autorizados, em conexão com o exercício de seus direitos de acordo com esse tratado ou a Convenção de Berna. A solução presente na GPLv3 para restringir a eficácia do DRM é afirmar que qualquer trabalho sob a GPL não pode ser considerado uma "medida tecnológica efetiva". Ou seja, é permitido incluir sistemas de DRM no software, mas não é ilegal quebrá-los.

Outro ponto tratado em maior profundidade na nova versão da GPL é a questão das patentes. Uma das motivações para as mudanças que foram implementadas na licença foi um acordo entre a Microsoft e a Novell, decorrente de uma alegação da Microsoft de que a Novell estaria infringindo suas patentes na distribuição SUSE Linux. Segundo o acordo, a Microsoft não processaria usuários por infração de patentes desde que o software fosse obtido de alguém que estivesse pagando à Microsoft para obter tais direitos. A GPLv3 é bem específica quanto ao caso e, na seção sobre patentes, afirma que uma organização não poderá distribuir trabalhos cobertos por essa licença caso faça parte de um desses acordos discriminatórios. Essa questão das patentes é uma das principais causas de receio por parte das empresas que detêm propriedade intelectual nessa forma. Como a comunidade de software livre muitas vezes busca o apoio dessas empresas, esse item tornou-se um forte fator para limitar a adoção da GPLv3.

8.4.2.3. AGPL

A AGPL é uma adaptação da GPL que inclui um termo sobre uso de um software através de uma rede. Inicialmente ela foi escrita pela empresa Affero Inc.[37], mas em 2007 foi lançada a AGPLv3 como parte do grupo de licenças da *Free Software Foundation*.

A condição adicionalmente imposta é que no caso do programa original dar aos usuários que interagiam com ele a opção de pedir o código-fonte completo, tal opção deve ser mantida em qualquer versão modificada. Dessa forma, mesmo não havendo a distribuição de um binário, um aplicativo web público sob esta licença precisa se manter aberto para qualquer usuário que interaja com ele.

[36] `www.defectivebydesign.org`
[37] `www.affero.org`

Dessa forma, ela é recomendada para projetos em que há interação via rede e busca-se o *copyleft*. A AGPL é considerada a mais "viral" das licenças, portanto deve ser evitada em projetos em que haja qualquer expectativa de utilização sob outra licença, a não ser que seja adotado um modelo de licenciamento múltiplo.

8.4.2.4. A mecânica da GPL

Devido a seus termos liberais, as licenças permissivas dificilmente dão ensejo a conflitos jurídicos; As licenças recíprocas, por outro lado, estabelecem regras às vezes rígidas para garantir a reciprocidade. Essa característica as torna mais suscetíveis a gerar situações em que há conflitos de entendimento entre as diversas partes interessadas. Consequentemente, o mecanismo jurídico pelo qual elas funcionam, seu alcance e sua relação com a legislação acabam tendo maior relevância que nas licenças permissivas.

A GPL, em particular, é objeto de bastante discussão, já que é possível identificar casos limítrofes em que a aplicação dos seus termos não é clara ou em que variações na legislação podem afetar seu significado. Por causa disso, a GPL inclui um preâmbulo que explica os princípios que baseiam a licença e seus principais objetivos. Apesar desse preâmbulo ser bastante citado nas discussões, ele não tem valor jurídico, ou seja, por não fazer parte dos termos e condições, suas palavras não precisam ser obedecidas por quem obtém a licença do software. Seu objetivo é apenas melhorar o entendimento da GPL em seu contexto, explicando o que é software livre e a importância do *copyleft*.

Não é raro encontrar textos norte-americanos que afirmam que "a GPL não é um contrato" [Moglen 2001; Jones 2003], e o próprio texto da GPL faz referência indiretamente a isso[38]. Isso vem do fato de a legislação americana ter uma visão mais restrita sobre o que é um contrato que a brasileira. Nos Estados Unidos, um contrato é visto como um acordo formal entre as partes. Em contratos desse tipo, existe a expectativa de aceitação explícita dos termos do contrato e estabelecimento de contrapartidas entre as partes, o que pode inclusive abrir a possibilidade para a exigência de direitos similares àqueles definidos pelo código de defesa do consumidor brasileiro.

A caracterização da GPL nesses termos iria contra os objetivos da FSF, pois exigiria a definição formal de contratos entre desenvolvedores e usuários, o que dificultaria fortemente a evolução do software de maneira colaborativa. Além disso, abriria as portas para exigências referentes a garantias e responsabilidades que trariam um risco inaceitável para o fornecimento de software pela FSF. Finalmente, a FSF estava preocupada em definir termos de licenciamento para o software livre que fossem válidos internacionalmente, e não apenas nos Estados Unidos.

Para evitar essas dificuldades, a GPL é definida como uma mera cessão de direitos de autor e, portanto, está sujeita apenas à legislação de *copyright* nos Estados Unidos e em países que possuam um núcleo comum com estas (como o Brasil). Ela permite o uso incondicional do software, e impõe restrições apenas à sua redistribuição, com ou sem modificações. Como a redistribuição sem autorização do autor é completamente proibida pela legislação, ela na prática consiste não em um conjunto de restrições, mas em um

[38] Cláusula 5 da GPLv2: "You are not required to accept this License, since you have not signed it."

conjunto de condições para a cessão desses direitos. O usuário, ao redistribuir o software, aceita tacitamente essas condições; caso contrário, ele não tem o direito de redistribuir o software e, portanto, infringe a legislação de *copyright*. Esse mecanismo dispensa a necessidade de formalização do acordo entre as partes e, já que corresponde a cessão de direitos sem contrapartidas, minimiza os riscos referentes à responsabilidade dos autores.

A despeito das diferenças entre as leis americana e a de outros países, a grande maioria das nações é signatária da convenção de Berna que estabelece alguns aspectos básicos comuns sobre direitos de autor para esses países. Assim, as consequências da GPL sob o ponto de vista da lei de direito autoral da maioria dos países (inclusive do Brasil) são bastante similares às da lei americana. De fato, a redação da GPL se baseou nesses princípios exatamente para tornar a GPL mais homogênea frente à legislação de diferentes países.

No caso da legislação brasileira, a despeito de suas características diferentes da americana, o efeito da GPL na prática é muito similar. Embora ela possa ser caracterizada como um contrato mesmo sem um acordo formal entre as partes, esse contrato é melhor enquadrado como um contrato benéfico (sem contrapartidas para uma das partes), o que também isola fortemente os autores de obrigações referentes à sua responsabilidade ou outras reciprocidades. Além disso, essa caracterização não altera a relevância da relação da GPL com a legislação de direitos de autor nacional, de forma similar ao que acontece nos Estados Unidos. Assim, embora possa ser enquadrada pela legislação brasileira de maneira ligeiramente diferente, na prática a GPL estabelece mecanismos de funcionamento no país similares aos esperados pelos seus autores.

8.4.3. Recíprocas Parciais

Licenças recíprocas parciais, também chamadas de *copyleft fraco*, determinam que modificações do trabalho coberto por elas devem ser disponibilizadas sob a mesma licença. Porém, quando o trabalho é utilizado apenas como um componente de outro projeto, esse projeto não precisa estar sob a mesma licença. Alguns autores, como Simon Phipps [Sun Microsystems 2006], utilizam a denominação licença baseada em arquivo para essa categoria, enquanto as recíprocas totais seriam licenças baseadas em projeto.

Considera-se que essas licenças são as que melhor equilibram dois importantes fatores do modelo de software livre: Atração de interesse para a comunidade e força e longevidade do código-fonte disponível. Ao mesmo tempo que essas licenças permitem que os desenvolvedores utilizem o trabalho para criar software que será licenciado como preferirem, modificações e melhorias feitas ao próprio trabalho são obrigatoriamente disponibilizadas à comunidade [Sun Microsystems 2006].

A *Free Software Foundation* recomenda o uso desse tipo de licença apenas em casos específicos. Seu argumento é a necessidade de fortalecer o software livre em detrimento do software restrito. Assim, quando as funcionalidades de uma biblioteca não estão facilmente disponíveis para uso em software restrito, seria melhor mantê-las dessa forma, utilizando uma licença recíproca total. Dessa forma, o software livre teria uma vantagem sobre concorrentes restritos. Porém, se as funcionalidades já estão ao alcance de software restrito, e portanto essa vantagem não está em questão, então uma licença recíproca parcial é recomendada, pois esse modelo ajuda a aumentar o número de usuários

da biblioteca.

Alguns advogados, como Lawrence Rosen [Rosen 2005], defendem que o uso de bibliotecas que são apenas ligadas a um novo software não caracterizaria um trabalho derivado, mas sim um trabalho coletivo. Ele faz uma analogia a páginas na web, em que cada uma é um trabalho com direito autoral individual, apesar de muitas vezes estarem presentes ligações de uma para a outra. Segundo ele, esse tipo de relação consiste em um trabalho coletivo. Portanto, nesse cenário, mesmo um software sob uma licença recíproca total poderia ser usado como biblioteca de outro que estaria sob outra licença. Porém, há controvérsias quanto a essa questão e as leis de direito autoral variam de país para país e, portanto, o uso de licenças recíprocas parciais faz-se necessário em casos nos quais o autor quer garantir que o desenvolvimento da biblioteca seja feito no modelo de software livre mas ao mesmo tempo quer permitir seu uso em projetos que utilizam outras licenças.

8.4.3.1. A Licença LGPL

A *GNU Lesser General Public License*, ou LGPL[39], originalmente denominada *GNU Library General Public License*, foi escrita em 1991 pela *Free Software Foundation*. Assim como a GPL e a AGPL, passou por grandes modificações no final de 2007 para adequar-se à versão 3 das licenças.

Seus termos são bastante complexos e é necessário estar atento a todas as regras e exeções presentes no texto, mas a ideia geral é que trabalhos que apenas usam a biblioteca que está sob essa licença, considerados isoladamente, podem ser licenciados de outra forma. Por outro lado, se o trabalho for baseado na biblioteca de forma mais próxima, ele deve ser licenciado como LGPL, GPL ou AGPL. A fronteira exata entre esses dois casos nem sempre está clara.

8.4.3.2. A Licença Mozilla

A *Mozilla Public License*, ou MPL[40], surgiu quando da transformação do Netscape Communicator em software livre. Ela é considerada muito bem escrita e serviu como modelo para muitas das licenças de software livre comerciais que a seguiram [Rosen 2005].

A MPL une características de licenças recíprocas e de licenças permissivas, e portanto também é categorizada como uma licença recíproca parcial. Mas, diferentemente da LGPL, aqui a delimitação é bastante clara: O código coberto pela licença deve ser redistribuído pelos termos da licença Mozilla, porém, esse código também pode ser utilizado em trabalhos ampliados, que podem estar sob outra licença. Na prática, normalmente isso significa que os arquivos que contém código MPL devem seguir essa licença, enquanto os demais estão livres para utilizarem a licença desejada.

[39] `www.fsf.org/licensing/licenses/lgpl.html`
[40] `www.mozilla.org/MPL/MPL-1.1.html`

8.4.4. Dificuldades com licenças de software livre

Gerenciar possíveis conflitos entre licenças é muito importante para não prejudicar a reputação do projeto e evitar complicações legais. Por esse motivo, ao iniciar um novo projeto, devem ser examinados assim que possível os potencias problemas de compatibilidade entre as licenças de componentes que serão utilizados. É recomendável já declarar a licença que será utilizada na especificação do programa, para que os programadores responsáveis já busquem componentes compatíveis [Open Source Observatory and Repository sd].

Para verificar a compatibilidade legal entre componentes é necessário considerar, por um lado, as obrigações que são colocadas nas cláusulas das licenças e, por outro, as diversas formas como componentes de software podem ser utilizados em conjunto e distribuídos. Tudo isso deve ser visto tendo ainda como base toda a legislação que, de várias formas, cobre o licenciamento de software.

As licenças permissivas se mostram as mais fáceis de compatibilizar com outras licenças, na medida em que permitem que trabalhos derivados sejam redistribuídos sob outros termos, até mesmo como software restrito. Dessa forma, ao escolher um software que utiliza uma licença permissiva, as chances de incorrer em problemas de licenciamento são bastante reduzidas. Porém, a liberdade de distribuir um software derivado sob outra licença não implica na possibilidade de usar qualquer uma delas. Cada licença possui termos específicos, que podem causar conflitos com os de outra licença, dependendo de quão restritivos eles são. Isso acontece principalmente quando em uma das pontas temos a GPL, que obriga que o trabalho resultante não apresente nenhuma exigência além daquelas presentes em seus termos. No caso da licença Apache, por exemplo, não há compatibilidade com a versão 2 da GPL, o que impede que seja distribuído um software que misture componentes dessas duas licenças.

Do outro lado do espectro de obrigações temos as licenças recíprocas. Segundo seus termos, o licenciante fica obrigado a redistribuir o software sempre sob termos muito similares; geralmente sob exatamente a mesma licença. Como essas licenças em geral são razoavelmente complexas, analisar seu impacto pode ser um tanto trabalhoso, em particular no tocante a detalhes como variações na legislação dos diferentes países, ao entendimento sobre o que é ou não um trabalho derivado e sobre quando há ou não redistribuição. Por exemplo, pode-se questionar se instalar um software GPL nos computadores de uma empresa é ou não uma forma de redistribuição. De um lado, os computadores são de propriedade da empresa; de outro, os funcionários da empresa podem ser vistos como usuários independentes. De forma similar, a carga dinâmica de bibliotecas através de chamadas de sistema como `dlopen()` pode ser ou não considerada uma forma de construção de um trabalho derivado da biblioteca em questão. Finalmente, segundo Determann [Determann 2006], combinações de programas com software GPL são de certa forma perigosas para empresas com um modelo de licença restrito no caso de elas também quererem distribuir o próprio código sob a GPL.

8.5. Negócios baseados em software livre

Como dito anteriormente, a maior parte do dinheiro gasto em software não está vinculada à compra de licenças; Assim, os modelos de negócios já comumente em uso pelos

fornecedores de software restrito em princípio se aplicam igualmente ao software livre. Por outro lado, como também já mencionado, a maior competição entre fornecedores e a relação com a comunidade estabelecem uma dinâmica específica envolvendo o software livre.

Diversas empresas têm obtido sucesso comercial explorando o software live de várias maneiras diferentes. Às vezes essa exploração envolve não apenas software livre, mas também sistemas híbridos, em que software restrito é agregado ao software livre. Algumas abordagens são específicas para um tipo de produto ou situação específicos, como ocorre com a propaganda no navegador Firefox; Outras são restritas a um nicho de mercado muito limitado, como ocorre com fornecedores de software em mídias físicas (CD/DVD); Outras são adequadas para empresas de grande porte, enquanto ainda outras se prestam melhor a empresas pequenas e médias.

No entanto, provavelmente o aspecto mais importante que diferencia os vários modelos de negócio que envolvem software livre consiste na participação no desenvolvimento. Alguns modelos estão diretamente vinculadas à atividade de desenvolvimento, financiando o envolvimento de programadores profissionais na evolução do software com a consequente melhoria em sua qualidade e funcionalidades. Outros não envolvem o financiamento do desenvolvimento em si, mas promovem a adoção do software no mercado, como treinamento e consultoria. No entanto, todos eles fazem parte do ambiente econômico e tecnológico que viabiliza o software livre no mercado.

Na medida em que o software livre tem crescido consistentemente e deve continuar crescendo, é de se esperar que novos modelos ainda venham a surgir, mas alguns modelos já têm sido usados consistentemente.

8.5.1. Principais modelos de negócio

Um modelo de negócio tradicional envolvendo o software livre é a distribuição remunerada do software. Nesse modelo, o usuário paga o fornecedor para obter o software de maneira mais conveniente que o que seria possível de outra maneira. Por exemplo, a FSF, na década de 80, cobrava pelo envio do código-fonte de seus programas através do correio, já que poucos usuários tinham acesso à Internet e, mesmo para quem o tinha, obter o software através da rede era demorado.

Esse modelo continuou sendo explorado mesmo com a popularização da Internet, porque a velocidade do download em várias regiões era inaceitável. Várias das distribuições Linux que se tornaram populares nos anos 90, como Red Hat e Suse, encontraram nesse mecanismo sua primeira fonte de renda, através da venda de CDs de instalação. Atualmente, esse modelo ainda pode ser explorado em alguns nichos geográficos, mas tende a se tornar cada vez menos relevante.

Uma outra abordagem bem conhecida é a comercialização de extensões restritas com base em um núcleo livre. Nesse caso, tanto a verba obtida com o licenciamento quanto o posicionamento favorável no mercado em relação à prestação de serviços focados nessas extensões oferecem vantagens comerciais para a empresa desenvolvedora. Ao mesmo tempo, a presença da empresa na comunidade que desenvolve o sistema livre básico favorece o seu desenvolvimento e minimiza a dependência do usuário no software

restrito, num equilíbrio intermediário entre opções totalmente restritas e totalmente livres.

Várias empresas e produtos são baseados nessa abordagem; Em particular, a IBM considera esse mecanismo estratégico na sua exploração do software livre, como pode ser visto na sua relação com produtos como o servidor web apache ou o ambiente eclipse. O eclipse, em particular, por sua arquitetura fortemente baseada em plugins, inspirou a criação de diversos produtos de várias empresas diferentes baseados nesse modelo.

Um outro modelo bastante utilizado consiste na liderança de um projeto de desenvolvimento de software livre, preferencialmente iniciado pela própria empresa, e na exploração de serviços relacionados. Nesse caso, dado o papel de liderança no projeto, a reputação da empresa na comunidade que gira em torno do software tende a ser altamente favorável, colocando a empresa numa posição vantajosa no mercado. Essa posição pode permitir à empresa oferecer seus produtos de forma competitiva mesmo com preços acima dos seus concorrentes e, dependendo do tipo de produto, ainda oferece outras oportunidades. Além de serviços de consultoria, suporte, integração etc., a empresa está em condições de oferecer programas de certificação, como ocorre com a Sun e o Java ou a Red Hat e suas várias certificações em torno de seus produtos.

Algumas empresas identificam na venda de licenças um modelo de negócios viável em diversas situações ao mesmo tempo em que reconhecem as vantagens oferecidas pelo software livre. Essas empresas procuram obter os benefícios dos dois modelos através do licenciamento dual: O software desenvolvido por elas está disponível tanto sob licença livre quanto sob licença restrita, nesse último caso mediante pagamento.

Em geral, esse modelo se baseia na licença GPL, que exige que a redistribuição do software derivado siga os termos da própria GPL. Assim, empresas que se interessam por código GPL para a criação de um produto mas não pretendem distribuir esse produto sob uma licença livre estão impedidas de usar o software nesses termos. No entanto, elas podem entrar em contato com o detentor do direito de autor sobre esse software e negociar um outro mecanismo de licenciamento.

Para ter sucesso, esse modelo depende de uma posição favorável na comunidade, pois ele só é possível se a empresa for a detentora de todos os direitos de autor sobre o software. Na medida em que uma das grandes vantagens do software livre está justamente na contribuição de desenvolvedores estranhos à empresa, isso envolve conseguir que os colaboradores abram mão de seus direitos em prol da empresa. Evidentemente, isso só ocorre se esses colaboradores reconhecem o valor do esforço da empresa com relação ao software. Caso contrário, podem ocorrer cisões na comunidade (*forks*) em que os colaboradores passam a manter o software por conta própria, sem repassar seus direitos para a empresa.

A Oracle utiliza essa abordagem com o MySQL e, embora tenha havido cisões, elas não atingiram a maior parte da comunidade e a posição do sistema MySQL como um produto da Oracle é razoavelmente estável. Por outro lado, o mesmo não ocorreu com o projeto OpenOffice, também pertencente à Oracle. A comunidade se cindiu recentemente, dando origem a um novo software, o LibreOffice, que seguirá sendo desenvolvido de forma independente do OpenOffice. O LibreOffice já está sendo incorporado pelos principais distribuidores de sistemas baseados em Linux, em detrimento do OpenOffice,

indicando que essa cisão realmente atingiu fortemente a comunidade e a posição da Oracle.

Um outro mecanismo às vezes utilizado para obter as vantagens oferecidas tanto pelo software livre quanto pelo software restrito é a mudança de licença ao longo do tempo. Nessa abordagem, a versão mais nova do software é disponibilizada sob licença restrita, mas versões anteriores são disponibilizadas sob licenças livres. Assim, é possível criar uma vantagem competitiva para a versão restrita e, ao mesmo tempo, oferecer o software sob forma livre para a comunidade, que pode utilizá-lo livremente, mas sem acesso aos seus últimos recursos. Da mesma forma que o licenciamento dual, essa abordagem depende de uma posição favorável junto à comunidade, pois também depende da posse sobre os direitos de autor do software.

Um modelo ainda pouco explorado é a integração com produtos de hardware. Nesse tipo de cenário, o software muitas vezes é encarado como uma origem de custos e não como uma fonte de renda. Assim, a transformação de *drivers* de dispositivos ou software embarcado em software livre pode promover melhorias na qualidade final do produto com custos iguais ou menores que o possível com software restrito.

Um exemplo são os dispositivos de telefonia celular baseados nas plataformas *Maemo* e *Android* que, além de serem baseadas em software livre, incentivam a extensão do sistema através de novos aplicativos que podem ser instalados pelo usuário. Muitos desses aplicativos são software livre, o que talvez ocorresse com menor frequência se a plataforma em si fosse restrita.

Um outro modelo comum envolve o uso do software livre para a prestação de serviços diversos. Por exemplo, a google presta diversos serviços na web; Para isso, ela faz uso de diversos sistemas de software livre. De forma similar, empresas que oferecem serviços de hospedagem web normalmente são baseadas em servidores linux e utilizam amplamente programas como apache, php etc.

Embora seu foco não seja diretamente o software, o fato de essas empresas fazerem forte uso dele as torna membros importantes da comunidade. Elas comumente utilizam o software em vários ambientes diferentes, sob carga razoável e, graças a isso, normalmente identificam rapidamente falhas, sugerem funcionalidades e muitas vezes pagam pelo desenvolvimento de novas características que consideram importantes.

Provavelmente a mais comum e variada forma de exploração do software livre consiste na oferta de serviços agregados. Existe uma infinidade de empresas especializadas na oferta de cursos e treinamentos, serviços de consultoria, implantação, manutenção e suporte técnico, personalização, integração etc. Essa forma de exploração é relativamente simples, não depende de grande capital e é fundamental para o bom funcionamento de qualquer ambiente baseado em software.

Alguns dos serviços desse tipo podem ser oferecidos de maneira pontual, onde a cada serviço corresponde um contrato com prazo determinado, ou sob a forma de assinatura, onde o usuário paga um valor fixo mensal que cobre os custos com os serviços que vierem a ser necessários. Um exemplo de serviço baseado em assinatura é a Red Hat Network, com a qual o usuário gere seu parque de máquinas através de uma interface centralizada, monitorando o estado do sistema, verificando a necessidade de atualizações etc.

Outros modelos ainda podem vir a ser melhor explorados ou mesmo criados, dependendo da criatividade das empresas envolvidas. Essa criatividade trouxe para o navegador Firefox uma excelente fonte de renda através de um contrato com a Google, pelo qual a página inicial do navegador, por padrão, é a página de busca da Google. Um outro modelo que pode vir a ser mais explorado é o uso de franquias, que possibilitam a ação em âmbito local com base em estratégias, técnicas e marketing globais.

8.5.2. Alguns casos de sucesso

Como mencionado, existe um grande número de empresas fortemente vinculadas ao software livre, em geral atuando através de um ou mais dos modelos elencados anteriormente.

Um exemplo de empresa que coordena alguns projetos de software livre e, com isso, obtém uma posição favorável no mercado é a Sun Microsystems, recentemente adquirida pela Oracle por 7.4 bilhões de dólares. Provavelmente o projeto de software livre mais importante sob a coordenação da Sun é a plataforma Java, que consiste nas especificações da linguagem e tecnologias relacionadas, na marca "Java", na implementação da máquina virtual Java (JVM) e de suas bibliotecas-padrão. Embora a máquina virtual Java da Sun (JVM) e as bibliotecas que a acompanham originalmente não fossem software livre, suas licenças foram alteradas para a GPL versão 2 em 2007. Essa mudança ocorreu quando o Java já era reconhecido como uma ferramenta fundamental para diversas áreas da computação.

A Sun oferece diversos tipos de certificação em Java e tecnologias relacionadas, além de oferecer outros serviços baseados em Java. Ela também utiliza um mecanismo de licenciamento dual com o Java: Além da GPL, a JVM e suas bibliotecas são diponibilizadas sob uma licença restritiva, mas gratuita, e existem algumas pequenas diferenças entre essas versões. A origem dessas diferenças está no fato de que a Sun não é a detentora de todo o direito autoral da JVM, o que a impediu de relicenciar algumas partes do código sob a GPL.

Um dos aspectos que impediu a Sun de licenciar a JVM sob uma licença livre antes de 2007 foi a preocupação com a integridade da plataforma. A Sun tinha receio que o código fosse cindido para criar versões incompatíveis da plataforma e da linguagem Java. Como solução para esse problema, a Sun utiliza a marca registrada "Java", de sua propriedade. Se o código for modificado de forma a alterar a especificação da linguagem ou da plataforma, o distribuidor não pode mais usar o nome "Java" para essa versão modificada. Assim, a Sun garante que qualquer implementação de Java que utilize esse nome está de acordo com a especificação.

A Sun participa de outros projetos de software livre, como o OpenOffice e o GNOME. Esses projetos são usados como base para seus produtos licenciados de forma restrita: O StarOffice e o ambiente de janelas para o sistema Solaris.

Outra empresa com abordagem similar é a MySQL AB, que foi comprada em 2008 pela Sun Microsystems por 1 bilhão de dólares. A MySQL também baseia suas atividades no licenciamento dual e na oferta de serviços como treinamento e consultoria.

Uma empresa muito importante para o software livre é a Red Hat, que nasceu especificamente para prover serviços baseados em software livre e hoje atingiu um tamanho significativo, com contratos de distribuição com empresas com o a Dell e a IBM, além de ser uma das componentes do índice S&P 500. Atualmente, a Red Hat oferece serviços genéricos baseados em assinatura, como a Red Hat Network e serviços de suporte, além de serviços de consultoria, treinamento e certificação. Por seu papel como provedora de soluções para sistemas de porte relativamente grande, a Red Hat participa fortemente no financiamento do desenvolvimento do kernel Linux, de maneira a poder oferecer soluções estáveis e escaláveis para seus clientes.

De forma similar à Sun, a Red Hat faz uso de sua marca registrada para diferenciar seus produtos. O software livre distribuído pela Red Hat (em particular, as diferentes versões do Red Hat Enterprise Linux – RHEL) está vinculado a essa marca registrada, o que restringe a possibilidade de outras empresas de fornecerem produtos com essa marca no mercado. No entanto, é possível tomar o código-fonte de todos os produtos da Red Hat que são baseados em software livre e recompilá-los sem menções a essa marca. Essas novas versões são idênticas às originais em termos de funcionalidade, exceto pelo acesso aos serviços oferecidos pela empresa. Diversos grupos efetivamente realizam essa operação e distribuem os resultados livremente. O mais conhecido deles é o CentOS, que é equivalente ao RHEL.

A IBM atua em diversas áreas na computação, mas tem tido um grande foco no software livre, com fortes investimentos em diversos sistemas. Em particular, ela deu origem à Fundação Eclipse quando transformou o código-fonte do Eclipse em software livre, e continua tendo importante papel na manutenção da Fundação. Ela também apoia fortemente e há muito tempo a Fundação Apache.

A IBM explora os resultados obtidos por essas fundações, além de outros sistemas de software livre, principalmente através da oferta de serviços de consultoria e da venda de extensões sob licença restrita para esse software. Em alguns casos, como é o caso do servidor web apache, um produto livre é inteiramente relicenciado sob licença restrita (desde que esse relicenciamento seja permitido pela licença livre original).

O modelo baseado em serviços agregados não exclui as pequenas e médias empresas. A TrollTech é uma empresa de médio porte que desenvolve a biblioteca Qt, bastante popular entre desenvolvedores que precisam criar sistemas compatíveis com múltiplas plataformas. Ela operou durante muitos anos (quando era muito menor) explorando o modelo de licenciamento dual, até ter sido comprada por 153 milhões de dólares em 2008 pela Nokia.

Com a compra, a Nokia pretendia usar a Qt como componente central do sistema Maemo, um novo sistema operacional para telefones celulares baseado no Linux. A oferta de um sistema operacional de qualidade é um diferencial importante para aparelhos celulares, e o uso de software livre foi identificado como um meio de incrementar essa nova plataforma. A Nokia lançou alguns produtos e um aparelho baseados no Maemo; No entanto, graças a um recente acordo com a Microsoft, o Maemo deixou de ser o foco da companhia nessa área.

Um exemplo de pequena empresa operando com sucesso em torno do software

livre é a CACE Technologies. Ela foi fundada pelo principal desenvolvedor do software wireshark, e se baseia tanto na oferta de serviços quanto de extensões sob licenças restritas para esse software.

Um outro exemplo é a cooperativa CoLivre, da Bahia. Trata-se de uma cooperativa de profissionais que oferecem serviços baseados nas diversas soluções livres desenvolvidas por eles ou com as quais eles colaboram diretamente no desenvolvimento. Uma dessas soluções é o Noosfero, uma plataforma para redes sociais, que vem ganhando espaço no mercado e que conta com o apoio de uma comunidade de desenvolvedores externos à CoLivre.

A área de treinamento é fundamental para a difusão do software livre, e várias empresas têm tido sucesso em abordar esse mercado. No Brasil, duas empresas bem conhecidas atuando nessa área são a Impacta e a 4Linux.

Finalmente, existem diversas empresas que baseiam suas operações em software livre. Um exemplo são os provedores de acesso à Internet e hospedagem de sítios web, como a Insite, Locaweb, UOL e outras. Outro exemplo é a Paggo, que utiliza fortemente software livre para o desenvolvimento de seus produtos.

8.6. O futuro do software livre

O software livre se apresenta atualmente como um modelo já consolidado e viável em um número crescente de aplicações e ambientes. No entanto, restam incertezas jurídicas e dificuldades, por exemplo com patentes de software, possíveis problemas de interoperabilidade com plataformas e protocolos criados e baseados em implementações não-livres e a necessidade de demonstrar a viabilidade econômica do modelo em mais casos e contextos. Alguns desses desafios estão vinculados às barreiras criadas pelo modelo e cultura do software restrito.

Um exemplo bem atual, que se apresenta como um desafio importante para a comunidade de software livre, é a necessidade de criar mecanismos para garantir que as licenças de software livre sejam respeitadas. Em particular, esse tipo de problema tende a se agravar em se tratando de sistemas embarcados ou plataformas móveis (como telefones celulares) baseadas em ambientes como o Android e suas aplicações. Problemas e a violação das licenças vêm sendo denunciados e discutidos na Internet[41] e nas comunidades.

A atual tendência em direção à computação em nuvem (*cloud computing*) e o sucesso de redes sociais baseadas em software restrito também apontam para um caminho pouco desejável do ponto de vista do software livre. Por sua característica centralizadora, esse tipo de solução se apresenta como um risco para a autonomia dos usuários (pessoas físicas, empresas e governos) em relação à sua interação com seus dados e com a tecnologia que os administra.

Justamente por causa dessa ameaça, diversos esforços vêm sendo realizados no sentido de desenvolver soluções livres para esses tipos de aplicação. O consórcio OW2, por exemplo, administra um projeto para o desenvolvimento de soluções livres para

[41] http://mjg59.livejournal.com/

computação em nuvem, o *Open Source Cloudware Initiative*, com o intuito de garantir padrões abertos e interoperabilidade em ambientes em nuvem.

De forma similar, a interoperabilidade entre redes sociais vem sendo um tópico de crescente interesse[42]. Novas redes sociais capazes de interagir entre si e baseadas em software livre (como GNUSocial, StatusNet e Noosfero) podem abrir espaço para a melhor produção e disseminação do conhecimento e da informação sem a necessidade de depender soluções restritas sob o controle de uma única empresa. Um outro projeto relacionado e de grande interesse é o *Freedombox*[43], focado na proteção à privacidade.

Por fim, no âmbito dos negócios, em particular no Brasil, é necessário criar mais formas de incentivo financeiro à produção de software livre e ao empreendedorismo baseado no software livre. Esse tipo de apoio é fundamental para o crescimento e consolidação da indústria de software no Brasil de maneira geral e do software livre em particular.

A despeito desses desafios, o crescimento do software livre, abarcando cada vez mais áreas diferentes da computação, é inevitável. A exploração do software livre neste momento é uma oportunidade única para empresas e pessoas de buscar atividades econômicas sustentáveis que também colaboram para o bem-estar social.

Referências

Abrahamsson, P., Salo, O., Ronkainen, J. e Warsta, J. (2002). Agile software development methods. Relatório técnico, VTT Technical Research Center of Finland.

Castells, M. (2006). A Era da Intercomunicação. *Le Monde Diplomatique Brasil*, Agosto 2006.

Corbucci, H. (2011). Métodos ágeis e software livre: um estudo da relação entre estas duas comunidades. Dissertação de Mestrado, Instituto de Matemática e Estatística da Universidade de São Paulo. Disponível em: `https://github.com/hugocorbucci/dissertacao-mestrado`.

Determann, L. (2006). Dangerous liaisons - software combinations as derivative works? *Berkeley Technology Law Journal*, 21(4).

DiBona, C., Ockman, S. e Stone, M., editores (1999). *Open Sources: Voices from the Open Source Revolution*. O'Reilly & Associates, Inc., Sebastopol, CA, USA.

Fowler, M. (2000). The new methodology. (Original version). Disponível em: `http://martinfowler.com/articles/newMethodologyOriginal.html`.

Ghosh, R. A. et al. (2006). Economic impact of open source software on innovation and the competitiveness of the information and communication technologies (ICT) sector in the EU. Relatório de Estudo. Disponível em: `http://www.flossimpact.eu` [Acesso em 29 mar 2011].

[42] http://status.net/2010/07/13/what-is-the-federated-social-web

[43] http://www.freedomboxfoundation.org/

González-Barahona, J. M., Pascual, J. S. e Robles, G. (2009). *Introduction to Free Software*. Fundación per a la Universitat Oberta de Catalunya, third edition edição. Disponível em: http://materials.cv.uoc.edu/continguts/PID_00148365/index.html.

Grad, B. (2002). A personal recollection: IBM's unbundling of software and services. *IEEE Annals of the History of Computing*, 24:64–71.

Jones, P. (2003). The GPL is a license, not a contract, which is why the sky isn't falling. Disponível em: http://www.groklaw.net/article.php?story=20031214210634851 [Acesso em 16 mai 2011].

Laurent, A. M. S. (2004). *Understanding Open Source & Free Software Licensing*. O'Reilly, Sebastopol.

Moglen, E. (2001). Free software matters: Enforcing the GPL, I. Disponível em: http://emoglen.law.columbia.edu/publications/lu-12.html [Acesso em 16 mai 2011].

Open Source Observatory and Repository. License Compatibility and Interoperability [página web]. (s.d.). Disponível em: http://www.osor.eu/legal-questions-1/licence-compatibility-and-interoperability [Acesso em 05 fev 2011].

PRESIDÊNCIA DA REPÚBLICA (1996). Lei Número 9.279, de 14 de maio de 1996. Disponível em: http://www.planalto.gov.br/ccivil/leis/l9279.htm [Acesso em 29 mar 2011].

Raymond, E. S. (1997). The cathedral and the bazaar. Disponível em: http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar [Acesso em 17 out 2008].

Rosen, L. (2005). *Open Source Licensing: Software Freedom and Intellectual Property Law*. Prentice Hall, New Jersey.

Simon, I. e Vieira, M. S. (2008). O rossio não rival. Disponível em: http://www.ime.usp.br/~is/papir/RNR_v9.pdf [Acesso em 16 mai 2011].

Stallman, R. (2002). Software patents – obstacles to software development. Palestra. Disponível em: http://www.cl.cam.ac.uk/~mgk25/stallman-patents.html [Acesso em 29 mar 2011].

Stallman, R. M. (1990). The GNU manifesto. In *Computers, ethics, & society*, pp. 308–317. Oxford University Press, Inc., New York, NY, USA. Disponível em: http://portal.acm.org/citation.cfm?id=77685.77850.

Sun Microsystems (2006). Free and open source licensing.

Tapscott, D. e Williams, A. D. (2006). *Wikinomics: How mass collaboration changes everything*. Portfolio – Penguin Books, New York.

The Open Source Initiative - OSI. History of the OSI [página web]. (1999). Disponível em: http://www.opensource.org/history [Acesso em 29 mar 2011].

Torvalds, L. e Diamond, D. (2002). *Just for Fun: The Story of an Accidental Revolutionary*. HarperCollins Publishers.

Warsta, J. e Abrahamsson, P. (2003). Is open source software development essentially an agile method? Disponível em: http://agile.vtt.fi/docs/publications/2003/2003_oss.pdf.

Wasserman, A. I. (2011). How the internet transformed the software industry. *Journal of Internet Services and Applications*, 2(1).

Weber, S. (2004). *The Success of Open Source*. Harvard University Press, Cambridge.