Aspectos do Trabalho Cooperativo no Desenvolvimento de *Software* Livre

Marcelo Sávio R. M. de Carvalho
IBM Brasil
msavio@br.ibm.com

Resumo. *Software livre é todo software cujo código fonte está disponibilizado publicamente, e seu conteúdo pode ser livremente modificado e redistribuído (e não sendo permitida a apropriação desse código). Uma conseqüência indireta desta liberdade é o aparecimento de comunidades de desenvolvimento que trabalham de forma descentralizada, através da Internet, desenvolvendo e mantendo os diferentes projetos de software livre. Essas comunidades, que a primeira vista parecem estar desorganizadas, produzem software de qualidade, com alta produtividade e satisfação entre os seus desenvolvedores e usuários. O objetivo deste artigo foi verificar os aspectos do Trabalho Cooperativo Suportado por Computador, CSCW (Computer Supported Cooperative Work), que suportam o desenvolvimento descentralizado de software livre, através do estudo de caso do sistema operacional Linux.*

Abstract. *Free software is all software whose code source is made public available and its content can be freely modified and redistributed (while not being allowed the appropriation of this code). An indirect consequence of this freedom is the dawn of development communities that work in a decentralized form, through the Internet, developing and keeping the different free software projects. These communities, that at first seem to be disorganized, produce quality software, with high productivity and satisfaction among its developers and users. The objective of this article was to verify the aspects of the Computer Supported Cooperative Work (CSCW) that supports the decentralized development of free software, through the study of case of the Linux operating system.*

Palavras-chave: *software livre, Linux, CSCW, cooperação, comunidades.*

1 Introdução

Apesar da definição apresentada acima, ainda existem controvérsias em relação ao que seja precisamente considerado *software* livre. Isso se dá pelo fato de que se trata de um assunto multifacetado, que contém questões relacionadas a assuntos como engenharia de *software*, propriedade intelectual, modelo econômico, política tecnológica, cooperação, liderança e motivação. Uma olhada mais de perto em alguns projetos de *software* livre revela, ainda, uma grande diversidade de objetivos, formas de participação e técnicas de implementação.

Ao analisar as atividades que suportam o desenvolvimento de *software* livre, do ponto de vista do trabalho cooperativo suportado por computador, é possível identificar um desafio comum a todos os projetos: a necessidade da criação e manutenção de um ambiente sociotécnico onde os participantes possam, cooperativamente, construir soluções para os problemas de interesse mútuo.

As práticas do desenvolvimento de *software* livre são importantes para a pesquisa em CSCW por dois motivos principais. Primeiro porque o desenvolvimento de *software* é tradicionalmente um processo de coordenação intensiva que tem atraído pesquisadores de CSCW (KRAUT, 1995; BUTTON, 1996) e segundo porque o processo de desenvolvimento de *software* livre é efetuado

por participantes geograficamente dispersos, com a ajuda de tecnologias colaborativas como o email, conversação on-line ou via *Web* e sistemas de controle de versão e de gerenciamento de configuração, tecnologias que, por sua vez, têm sido um aspecto central nas pesquisas de CSCW.

Embora não exista uma definição precisa do que seja um projeto de *software* livre, existe um consenso informal de que o *software* em conjunto com seus desenvolvedores, usuários e repositórios de código e documentos constitui um projeto, que desta forma abrange:

- Código fonte, que é o *software* propriamente dito, mas que existe em forma de inúmeras cópias entre todos seus usuários e repositórios de dados. *Software* livre, em geral, tem uma versão bem definida e distribuída a partir de um ponto central conhecido para o Projeto.
- Um grupo de desenvolvedores, que trabalha para codificar e corrigir o *software*. Os desenvolvedores, em todos os casos e sem exceção, trabalham cooperativamente através da Internet e podem ter status diferenciados dentro do projeto.
- Os usuários do *software*. Apesar de todo *software* ter usuários, sua participação em projetos de *software* livre é essencial, pois discutem inovações e apresentam erros encontrados com freqüência, incentivando os desenvolvedores a trabalhar por um produto melhor
- Os repositórios de documentos e códigos. Todo projeto tem algum *site* central que é uma referência para desenvolvedores e usuários buscarem códigos e informações atualizadas.

É interessante perceber que todo o processo de desenvolvimento e distribuição depende essencialmente do uso da Internet, e que esta é um pré-requisito para a existência de um projeto de *software* livre. Segundo o Sourceforge, que é o maior repositório de *software* livre e de código aberto disponível na Internet, existem em seus registros mais de cem mil projetos, em diferentes estágios de desenvolvimento, mantidos por mais de um milhão de participantes.

2 O Modelo de Desenvolvimento do *Software* Livre

Quando o desenvolvimento é feito por uma equipe pequena e local, ou por uma comunidade de prática, onde os participantes estão fortemente ligados uns aos outros, o trabalho cooperativo é certamente mais fácil, pois, nessas comunidades, os membros podem aprender as novidades de forma mais eficiente, resolver seus problemas de maneira mais tranqüila, e ainda têm mais possibilidades de encontrar novas oportunidades de inovação. Em contraste, a coordenação de uma equipe numerosa e remota precisa ser inevitavelmente formal e o gerenciamento das contingências sempre é uma tarefa mais difícil (KRAUT, 1995).

Os projetos de desenvolvimento distribuído de *software* livre apresentam características comuns, quando analisados do ponto de vista de construção cooperativa. Segundo Scharff (2002), essas características comuns podem ser mapeadas em um *framework* que as organiza de forma descritiva em um modelo que enfatiza a construção cooperativa através de quatro componentes principais: As pessoas (participantes), o que eles estão criando (objeto produzido), os processos empregados (processos colaborativos) e os meios usados para comunicação e coordenação do projeto (tecnologias colaborativas), descritos a seguir:

2.1 Os Participantes

São os componentes principais de um projeto de *software* livre. Sem um grupo de participantes ativos não haverá *software* produzido ou este será apenas um pedaço de *software* sem mudanças. A participação em um projeto de *software* livre abrange, além da produção do código fonte em si, a contribuição em forma de idéias, *feedbacks*, correções e demonstrações. Assim, os participantes podem ser simpatizantes, usuários, desenvolvedores, testadores, documentadores ou evangelistas.

Geralmente são ativamente envolvidos em todas as fases de um projeto e trabalham de forma voluntária, motivados por uma mentalidade altruísta.

Um estudo realizado por Boston (2002) com uma população de líderes e desenvolvedores de projetos registrados no Sourceforge, revelou, dentre outros fatos, que entre os entrevistados:

- 72,6% disseram que, quando estão programando, perdem a noção do tempo por estarem motivados, sendo o tempo de sono considerado a principal contrapartida;
- 44,9% são motivados a trabalhar no projeto pelo desafio intelectual que representa;
- Apenas 11,1% revelaram como motivação principal desbancar o *software* proprietário;
- 70% dos participantes são voluntários e não são financeiramente recompensados (direta ou indiretamente) pelo seu trabalho no projeto;
- O tempo médio de trabalho gasto nos projetos foi de 14,9 horas por semana;
- 48,1% revelou que o maior benefício pessoal é o aumento do conhecimento;
- 83% se identificaram com a comunidade *hacker*;
- 45,4 % dos participantes são programadores com experiência média de 11 anos;
- A maioria afirmou esperar do líder do projeto não só a realização de tarefas práticas (criação da base de código original e contribuições constantes), mas uma boa visão de longo prazo, iniciativa para diálogo, e disposição para integrar contribuições.

2.2 O Objeto Produzido

É o resultado da criação e dos refinamentos sucessivos. Ainda que a produção de código fonte seja atividade central de um projeto de *software* livre, este não é o único produto final, pois também figuram neste componente toda a documentação, os relatórios de defeitos, as ferramentas de instalação e outros materiais de suporte. A versão do *software* recomendada para uso em produção é chamada de distribuição padrão.

2.3 Os Processos Colaborativos

São convenções, procedimentos, práticas, papéis e responsabilidades que regem as interações entre os membros de um projeto de *software* livre. Geralmente são difíceis de definir, pois são conhecimentos tácitos, que a comunidade pode nem perceber que adota, ainda que certamente estejam presentes e desempenhem um papel importante. Diferentes comunidades podem ter diferentes perspectivas em relação à construção cooperativa, seja enfatizando a correção de erros, promovendo novas funcionalidades ou criando referências de implantação.

Essas diferentes perspectivas afetam a natureza dos processos colaborativos e normalmente aparecem quando são endereçadas as questões dos participantes e as respectivas respostas da comunidade. São situações que remetem a novas situações de colaboração, como por exemplo:

- Quando uma comunidade é intolerante com perguntas simples ou assume uma postura defensiva perante uma sugestão, pode inibir os participantes;
- Se algum participante quer contribuir com uma correção ou melhoria, a comunidade deve ter processos para absorver e encaminhar estas questões;
- É necessário que a comunidade esteja preparada para atender às múltiplas contribuições em paralelo para vários pedaços ou módulo do *software* em questão;
- É preciso ter um processo de adoção de mudanças na distribuição padrão, pois se nenhuma contribuição for incorporada, os participantes podem se sentir discriminados no projeto.

Em um projeto de *software* livre, o ciclo de discussão, mudança e incorporação de mudanças, por definição não termina nunca, pelo menos enquanto houver o projeto.

A técnica mais adotada nos projetos de *software* livre é a utilização de alguma forma de liderança. Normalmente um líder de projetos é uma pessoa (ou um pequeno grupo de pessoas) que participou da concepção da idéia ou de sua implementação inicial e que, principalmente, goza de respeito e reconhecimento por parte dos outros membros da comunidade. O papel desempenhado por um líder em um projeto de *software* livre pode variar muito porque depende fortemente da pessoa que exerce esta liderança, a começar pela própria definição desta função, referenciada por diversos nomes como "arquiteto-chefe", "mantenedor", "coordenador", etc. Em linhas gerais o líder provê direção e coerência para o projeto, resolve eventuais disputas e mantém o controle da distribuição padrão, diferenciando-se assim de um ambiente corporativo tradicional, cuja liderança normalmente está inserida em um contexto hierárquico de uma "cadeia de comando".

Outra técnica muito disseminada na organização de processos colaborativos de *software* livre diz respeito ao *framework* do projeto em si, que paraleliza a organização da comunidade e a arquitetura do *software* que está sendo desenvolvido. Entre as estruturas organizacionais típicas temos as decomposições funcionais hierárquicas em módulos fracamente acoplados e um núcleo central com módulos encaixáveis. Um importante objetivo de um *framework* desse tipo é a capacidade que oferece de, dado um problema específico, saber quais as mudanças que serão necessárias e onde precisam ser feitas.

A divisão de tarefas, diferente do modelo tradicional, é feita de forma democrática e baseada no interesse pessoal dos participantes, ainda que, quando muitas pessoas estejam trabalhando no desenvolvimento de uma mesma parte do sistema, possa haver uma delegação de tarefas visando evitar conflitos e que normalmente é explicitada através de uma lista de participantes e de subprojetos ativos, criando uma percepção informal de quem está trabalhando no quê. É importante frisar que cada participante contribui com o *software* livre por motivos pessoais e, normalmente, poucos se dedicam a fazer tarefas consideradas "chatas" (como documentação ou interfaces), ainda que sejam importantes.

Um conceito implícito do *software* livre que tem profundas implicações no processo colaborativo é o compartilhamento de informações. Qualquer um pode ter acesso, ler ou modificar qualquer código fonte no sistema. Este caráter público do código cria uma espécie de pressão social que motiva a produção de código de qualidade, pois todos os participantes querem que suas contribuições sejam reconhecidas pela comunidade (AOKI et al., 2001).

Mudanças em um projeto de *software* livre acontecem o tempo todo e o desafio é manter um *software* de qualidade sem ofender os membros da comunidade, pois críticas ou mudanças em um pedaço de código podem ser interpretadas como uma crítica pessoal. A submissão não só de erros e críticas, mas de correções e sugestões é importante para o desenvolvimento de uma comunidade.

2.4 As Tecnologias Colaborativas

As tecnologias colaborativas desempenham duas atividades essenciais ao processo de construção cooperativa. A primeira é dar aos participantes um canal de comunicação entre si e a segunda fornece mecanismos de armazenamento, distribuição e acompanhamento de versões de objetos produzidos pela comunidade. Ambas as atividades podem se basear em ferramentas síncronas ou assíncronas e de forma independente da sofisticação técnica e das questões temporais, ou seja, podem ser realizadas por contato físico face-a-face, videoconferência, *chat,* cartas postais, etc..

A maioria dos projetos de *software* livre é virtual, com participantes espalhados ao redor do mundo e, mesmo quando a maioria das pessoas de um projeto está em um único país, as dificuldades de agendar reuniões com presença física são consideráveis. Por essa razão a Internet é

um elemento indispensável na criação e desenvolvimento de projetos de *software* livre. Como os projetos de *software* livre tiveram sua origem e disseminação em paralelo com origem e disseminação da própria Internet, as principais tecnologias colaborativas utilizadas passam pelas ferramentas disponíveis nessa rede. Sobre esse aspecto Yamauchi (2000) oferece uma visão geral dos mecanismos de comunicação usados em projetos de *software* livre e afirma que, apesar da tendência acadêmica da pesquisa em CSCW apontar para a produção de ferramentas complexas para suportar trabalho cooperativo desta natureza, os projetos de *software* livre sobrevivem e comprovam a eficácia do uso de ferramentas e meios de comunicação simples e disponíveis há muito tempo na Internet, como o email, as listas de distribuição, os *newsgroups,* os *chats* e os *sites* repositórios de conteúdo.

Os mecanismos de armazenamento, distribuição e acompanhamento de objetos produzidos pela comunidade também utilizam ferramentas disponíveis na Internet. A distribuição padrão normalmente é disponibilizada em *website* conhecido, que funciona como repositório daquela versão (estática) até que seja substituída por outra mais nova.

Cada vez mais projetos de *software* livre estão adotando ferramentas de controle de versão para gerenciamento do conteúdo do objeto produzido. Essas ferramentas são vistas como uma extensão natural do processo de desenvolvimento, permitindo que se possam realizar modificações paralelas de forma coerente e padronizada, através de um mecanismo automatizado para identificar e controlar as modificações realizadas nos arquivos de um projeto ao longo do tempo, garantindo a integridade e a rastreabilidade das modificações.

As ferramentas que documentam e controlam os erros encontrados (e suas correções) são muito importantes para qualidade e a produtividade dos projetos de desenvolvimento distribuído de *software*. Cada erro informado por um usuário participante, é registrado em um informe individual e cada informe possui um ciclo de vida que vai desde a sua criação, no momento em que é informado, até seu fechamento, quando o erro é reparado no objeto produzido. Essas ferramentas, que normalmente têm interface Web ou via *email* permitem, em geral, um bom gerenciamento dos erros, com capacidade de localização, confirmação e detecção de erros duplicados ou que não se apliquem à versão atual do objeto.

2.5 As Inter-relações

Nenhum dos componentes desse *framework* conceitual existe de forma isolada e assim como cada aspecto muda com o tempo, o mesmo acontece com as relações. A seguir estão listadas as inter-relações mais encontradas entre os componentes:

2.5.1 O Uso

O objeto produzido em um projeto de *software* livre é um pedaço de *software* de computador, que as pessoas irão baixar e usar em um contexto específico. O uso é um tipo de atividade reflexiva, na qual o usuário cria um modelo de percepção das limitações e pontos fortes do *software* em uso e identifica mudanças e características que acreditam que o *software* deveria ter para atender suas necessidades e preferências. Quando mais o *software* estiver relacionado com cotidiano do usuário, principalmente relacionado ao seu trabalho, mais chances existem desse usuário querer contribuir com críticas e sugestões para o desenvolvimento do *software*.

2.5.2 A Facilitação Social

O processo colaborativo emerge dos participantes ao mesmo tempo em que influencia suas ações. A facilitação social é a maneira com que os processos colaborativos encorajam as pessoas a se envolverem no desenvolvimento e evolução do projeto. Por exemplo, o processo colaborativo pode requerer que as pessoas que corrijam erros anexem seus nomes às correções. Em alguns grupos isso pode ser um grande incentivo à correção de erros porque o reconhecimento pessoal e os elogios motivam as pessoas a contribuir cada vez mais. Em outro exemplo, se um líder que precisa aprovar todas as mudanças nunca aceita as sugestões da comunidade, seus participantes perceberão que suas contribuições não estão recebendo o devido reconhecimento e poderão abandonar o projeto. Em suma, a facilitação social se refere às reações dos participantes ao contexto social do projeto.

2.5.3 A Facilitação Técnica

O processo colaborativo está intimamente relacionado com as tecnologias colaborativas empregadas. Algumas vezes as limitações da tecnologia influenciam no processo colaborativo em si. Por exemplo, as equipes que dependem da comunicação por *email*, muitas vezes têm dificuldade de encontrar mensagens importantes devido ao volume e a inerente falta de estrutura dos sistemas de correio eletrônico. Para compensar essa deficiência, o processo colaborativo pode especificar que quando a palavra "ANNOUNCE" aparece entre colchetes no início do assunto da mensagem, significa que se trata de um anúncio importante e que todos devem ler. Em alguns grupos são criados filtros automáticos que varrem as mensagens em busca dos identificadores de anúncios importantes que, uma vez identificados, são imediatamente postados em uma página de anúncios no *website* do grupo. Neste caso uma tecnologia colaborativa foi adaptada para reforçar um processo colaborativo de divulgação de anúncios importantes. A facilitação técnica se refere às conexões entre os processos colaborativos e as tecnologias que suportam esses processos.

2.5.4 Gerenciamento das Mudanças

O gerenciamento das mudanças está relacionado ao uso das tecnologias de comunicação, armazenamento e controle de versões e de erros da distribuição padrão. Como os objetos produzidos devem estar publicamente disponíveis e fáceis de serem acessados, a tecnologia colaborativa deve prover uma maneira confiável de se armazenar e disponibilizar esses objetos. A estrutura dos repositórios da distribuição padrão geralmente reflete as mudanças que ocorreram ao longo do tempo que, junto com os padrões de nomenclatura (numeração de versões) e com as tecnologias de controle de versão e de erros, ajudam no rastreamento das mudanças no objeto produzido, facilitando a volta para uma versão anterior caso problemas sejam encontrados após uma mudança na distribuição padrão.

2.5.5 As Contribuições

Contribuição são os processos nos quais os indivíduos propõem melhorias ou extensões no projeto através de mudanças relacionadas à distribuição padrão do objeto produzido. A forma como uma contribuição acontece varia de projeto para projeto, mas certamente envolve todos os aspectos e componentes do framework. Participantes criam mudanças, que passam por alguns processos colaborativos onde o grupo decide aceitar, refinar ou rejeitar as contribuições. Durante esses processos, a tecnologia colaborativa coordena a comunicação e provê o suporte técnico para rastrear as múltiplas versões de uma mudança. Se uma contribuição é aceita, passa então a ser incorporada ao objeto produzido e atualizam-se todas as referências. Assim, apesar das contribuições serem atividades desempenhadas por participantes, são fortemente influenciadas pelo objeto, pela tecnologia e pelo contexto social dos processos colaborativos.

3 O projeto de desenvolvimento do Linux

O Linux é um *kernel* (cerne ou núcleo) de sistema operacional livre, multitarefa, preemptivo e multiplataforma (suporta mais de dez arquiteturas de *hardware* diferentes) e foi iniciado em 1991 por Linus Torvalds, então estudante da Universidade de Helsinque (Finlândia). Moon (2000) faz uma análise histórica do desenvolvimento do *kernel* e coloca claramente a importância do desenvolvimento em comunidade, fornecendo estatísticas históricas da participação de desenvolvedores externos:

O primeiro *release* da primeira versão do *kernel* possuía um pouco mais de três mil linhas de código, distribuídas em 88 arquivos escritos em linguagens C e Assembly. Atualmente o *kernel* está no *release* 6 da versão 2, cujo tamanho ultrapassa a quantidade de dois milhões e meio de linhas de código, distribuídos entre milhares de arquivos.

O *kernel* ainda hoje tem em Torvalds seu expoente máximo, que toma as principais decisões e define a filosofia geral do projeto. Entretanto, cada *release* tem sempre um mantenedor responsável, que aprova cada aperfeiçoamento e garante que não haja versões conflitantes. Em julho de 2003, Torvalds passou, pela primeira vez, a trabalhar oficialmente e integralmente dedicado ao projeto do Linux, como contratado do *Open Source Development Laboratories* (hoje *Linux Foundation*), uma entidade voltada para a disseminação do Linux.

3.1 Comunicação e Percepção

A lista de distribuição de emails *linux-kernel mailing list* (LKML) é o fórum central de discussão entre os participantes do projeto de desenvolvimento do Linux. Ainda que não haja uma hierarquia definida e que todos trabalham de acordo com interesses e aptidões pessoais, sabe-se que entre Torvalds e a comunidade de participantes existe um grupo de consultores técnicos, informalmente chamados de "*lieutenants"* ("tenentes"), que são participantes que ganharam o status de programadores competentes e com expertise de comprovada importância para o projeto através de anos de participação. Apesar desse grupo não existir de maneira formal, sua composição é aceita pela comunidade como uma espécie de "seleção natural", ainda que as informações e opiniões sobre quem faz (ou deveria fazer) parte do grupo divergem entre os participantes do projeto (KUWABARA, 2000).

Quando um novo participante entra na lista LKML, assim como acontece em outras listas na Internet, é sempre convidado a ler a documentação de perguntas e respostas mais freqüentes, que nesse caso, entre suas questões principais, esclarece algumas regras implícitas que regem o bom funcionamento da lista.

Apesar de ser um projeto virtual, com participantes espalhados ao redor do mundo, a comunidade de desenvolvimento do Linux se reúne pessoalmente em eventos anuais informais como o *Kernel Summit* para trocar experiências, combinar as agendas e próximos passos.

3.2 Controle de Versões e Erros

O projeto do desenvolvimento do Linux sempre foi, levando-se em conta o tamanho do objeto produzido e do número de participantes envolvidos, um dos projetos mais minimalistas em relação à infra-estrutura de desenvolvimento. Até a versão 2.4, incrivelmente, apenas listas de discussão e emails eram utilizadas por sua comunidade de desenvolvimento.

No de final de 2002, no entanto, foi lançado o *Kernel Bug Tracker* sistema para controle de erros. Outro recente projeto importante na manutenção da qualidade é o *Kernel Janitor*, que objetiva

"limpar" constantemente o código-fonte do kernel através da busca e eliminação de erros em trechos antigos através do reaproveitamento de correções recentes.

A implantação de uma ferramenta automática de controle de versão (e integração) do objeto produzido ainda demorou a acontecer no Linux, pois Torvalds sempre se opôs à idéia, uma vez que somente confiava tal tarefa aos mantenedores credenciados (SHAIKH, 2003). Com o número, cada vez maior, de mudanças sugeridas pelos participantes, entretanto, o projeto precisou adotar uma ferramenta de controle de versões.

3.3 A Padronização das Distribuições do Linux

Nos primeiros anos de existência do Linux, Torvalds simplesmente disponibilizava a versão estável do *kernel* e alguns comandos bem básicos. O usuário interessado em usar o sistema, entretanto, tinha que, além do baixar o *kernel* pela Internet, também arranjar todos os demais programas complementares, compilar tudo e acertar os arquivos de configuração para então poder proceder com a instalação. Como isso é trabalhoso até mesmo para um hacker iniciado no assunto, surgiram as distribuições, baseadas na idéia de se disponibilizar os programas principais pré-compilados, de tal forma que o usuário só precisasse pegá-los (em CD-ROM ou via Internet) e instalá-los em sua máquina.

Hoje existem dezenas de empresas e instituições que mantém centenas de distribuições e as principais diferenças entre elas são o sistema de empacotamento do software, a estrutura dos diretórios e algumas bibliotecas básicas que contém funções básicas para o funcionamento do sistema operacional. Quando uma nova versão dessa biblioteca é lançada, algumas distribuições a adotam imediatamente, enquanto outras, mais conservadoras, aguardam um pouco. Nesse meio-tempo, alguns programas podem funcionar em umas distribuições e não em outras.

A diversidade de distribuições de Linux, que foi providencial na disseminação do sistema em seus primeiros anos de vida, teve como conseqüência uma falta de padronização que terminou por criar problemas para a comunidade Linux. Entre outras questões, durante muito tempo era difícil obter versões pré-compiladas de aplicativos, exceto se tivessem sido preparadas especificamente para a distribuição de Linux que o usuário estivesse utilizando, o que exigia um conhecimento da instalação de programas a partir da compilação de seu código-fonte, o que se constituía numa grande barreira ao uso e adoção do Linux.

Com a entrada de participantes de maior porte no mercado criado em torno do Linux surgiu a necessidade de convergência em torno de um padrão, sem restringir a possibilidade de diferenciação entre as distribuições - mantendo assim a liberdade, sem permitir a fragmentação do Linux em uma série de alternativas incompatíveis entre si, como ocorreu com o sistema operacional UNIX. Assim, em 2001, foi criado o *Linux Standard Base* (LSB), que visa desenvolver e promover um conjunto de padrões para aumentar a compatibilidade entre as distribuições de Linux. O LSB é dividido em três componentes principais: a especificação do sistema, as ferramentas de teste e um exemplo de distribuição para referência. Apesar do padrão LSB ter sido adotado pelos muitos participantes do mercado Linux, incluindo as principais distribuições, os desenvolvedores de aplicativos e as empresas de suporte, ainda é possível encontrar aplicações que somente funcionam em determinadas distribuições do Linux. Os esforços de padronização são coordenados pela *Linux Foundation*.

4 Desafios para o Software Livre

O *software* livre vem ganhando espaço sistematicamente no cenário mundial e talvez esse interesse esteja crescendo mais rápido do que a consciência sobre a filosofia na qual é baseado, e isto pode

conduzir a problemas. Como já foi dito na introdução deste texto, trata-se de um assunto multifacetado, que contém questões relacionadas a diversos assuntos (engenharia de *software*, propriedade intelectual, etc.). Do ponto de vista dos aspectos relacionados à construção cooperativa, Rothfuss (2002) indica que esses fatores são decorrentes, principalmente, da falta de organização formal, e podem ser resumidos conforme a seguir:

4.1 Redundância de Esforços

A coordenação em projetos de *software* livre é relativamente pouca. Grupos independentes muitas vezes desenvolvem tarefas similares em paralelo sem conhecer o trabalho de seus pares, gerando um consumo adicional de recursos. Não há dúvida que o efeito colateral benéfico disso é o aumento do número de opções a escolher e que as diferentes alternativas melhoram a qualidade final do *software*. Entretanto estes trabalhos paralelos dificultam a escolha de alternativas por parte dos usuários, o que pode terminar em conflitos de posicionamento no mercado, gerando discussões apaixonadas, quase religiosas, com potencial de provocar "rachas" e enfraquecer o movimento de disseminação do *software* livre.

4.2 Dificuldades de Comunicação

Ainda que o inglês seja o principal idioma usado nos projetos de *software* livre, certamente não é o idioma nativo de todos os participantes. Isso pode acarretar em diversos problemas de entendimento, que tendem a piorar conforme os projetos aumentem de número de participantes ao redor do mundo. Outro problema de comunicação, que também está relacionado à expansão, diz respeito ao volume e à relevância do conteúdo das mensagens, que ainda são a base da comunicação dos grupos de desenvolvimento. É comum perceber em grupos de discussão, cada vez mais, os participantes reclamando da falta de etiqueta e boas maneiras em meio a uma profusão de informações inúteis. Isso certamente é menor nos grupos moderados, mas é inegável que a relação sinal-ruído na Internet não é das melhores atualmente e está piorando cada vez mais.

4.3 Senso de Prioridade

Em projetos de *software* livre, dificilmente as decisões importantes e profundas são tomadas com a agilidade necessária. Isso acontece devido à natureza distribuída do processo, que aliada a uma liderança tênue, pode impossibilitar a resolução de uma prioridade ou distorcê-la em direção às tendências pessoais de algum participante influenciador. Uma necessidade de mudança mais profunda normalmente desencadeia uma seqüência interminável de argumentos favoráveis e contrários, uma vez que ninguém pode forçar sua opinião aos demais e todos se sentem no direito de opinar, mesmo aqueles que não têm o conhecimento ou a preparação necessária para analisar suas considerações à luz de outras questões.

4.4 Proliferação de "grupos fechados"

Os projetos de *software* livre não têm regras formais ou convenções escritas. As relações entre seus participantes são regidas por um conjunto de regras consuetudinárias – não escritas, baseadas em costumes – criando uma "netiqueta" (regras de etiqueta na Internet). Dessa forma, os novos membros têm que gradualmente aprender as regras informais e, de certa forma, são medidos pelo sucesso em reagir às dicas passadas pelo grupo. As comunidades, principalmente as mais antigas, são entrosadas e conseguem melhorar a comunicação entre os membros através do

compartilhamento de contextos culturais, porém acabam por dificultar ainda mais a integração do grupo com os recém chegados. À medida que os participantes competem por atenção e reconhecimento de talento, essas barreiras são danosas ao projeto, principalmente aos menores, que ainda se estruturam. O desenvolvimento de *software* livre é um processo de aprendizado, onde as partes envolvidas contribuem para (e aprendem da) comunidade e esse equilíbrio é importante. Edwards (2000) chamou este fenômeno de "Comunidades Epistêmicas".

4.5 Falta de Foco

De acordo com estudos cerca de 70% dos participantes de *software* livre gastam 10 horas ou menos por semana em trabalhos relacionados ao projeto e essas horas são, em sua maioria, gastas durante à noite ou nos finais de semana (BERLECON, 2002). O baixo nível de participação pode introduzir problemas de comunicação, dado que muitos participantes têm dificuldade em acompanhar todos os desenvolvimentos realizados no projeto. Essas circunstâncias certamente dificultam o foco dos participantes no desenvolvimento do projeto. Normalmente o trabalho é conduzido aos pedaços e finalizado apenas após um longo período. O esforço dispensado em estar a par de todas as questões é geralmente tão grande que sobra pouco tempo para as contribuições efetivas. Muitos participantes perdem o interesse ou são absorvidos por outros comprometimentos externos, deixando muitos projetos pela metade.

4.6 Dependência de Pessoas-Chave

Muitos projetos de *software* livre dependem de poucas pessoas. Taurion (2004) afirma que 10% dos participantes contribuem com 78% do código; o segundo décimo, com 12%, o terceiro, com 3%; os outros, com menos de 1% cada. Existem algumas explicações plausíveis para esse fenômeno. A mais óbvia é que o nível de conhecimento necessário para se analisar e entender um código-fonte de um sistema grande requer treinamento e experiência. O esforço em adquirir esse conhecimento não é efetuado pela maioria dos participantes. Outra explicação está relacionada ao nível de reconhecimento que os participantes esperam receber por suas contribuições. Apenas um número pequeno de participantes acaba se tornando amplamente reconhecido por suas contribuições, o que tende a fortalecer ainda mais o papel dos contribuidores principais. Essa dependência pode se tornar um risco se uma pessoa-chave não puder continuar seu trabalho no projeto por algum motivo. Talvez seja impossível reconstruir todo seu conhecimento implícito a partir de seus objetos (código-fonte e documentação).

4.7 Escassez de Lideranças Competentes

O sucesso de um projeto de *software* livre depende de uma boa e carismática liderança. Além das qualidades intrínsecas da perspectiva da Engenharia de *Software*, o líder de projeto precisa endereçar questões de comunicação, marketing, juízo político e motivação. A liderança normalmente é feita por persuasão, pois não há um mandato oficial hierarquicamente estabelecido. O líder é avaliado não só por seus conhecimentos técnicos, mas por sua visão e habilidade em se comunicar com os co-participantes. (BOSTON, 2002). Ainda que haja muitos participantes com conhecimentos técnicos, as exigências são tão seletivas que se torna difícil preencher todas as posições de liderança com pessoal qualificado. O sucesso do Linux se deve muito mais à excelente liderança exercida por Torvalds do que por suas habilidades técnicas. A escassez de novos e bons líderes é uma questão muito sensível para o futuro do *software* livre, conforme atestado pelo próprio Torvalds (SEARLS, 2003).

4.8 Questões Legais

A propriedade intelectual, com seus *copyrights*, patentes, marcas registradas e segredos comerciais é certamente um dos campos mais obscuros e esotéricos da esfera jurídica, principalmente quando aplicada ao *software* e seus algoritmos. É muito difícil saber se determinado método para se resolver um problema de software já foi patenteado ou não, e a interpretação dessas leis varia conforme o país. Ainda que esse problema exista para todo o mercado de *software* (livre e proprietário), no modelo de desenvolvimento de *software* livre, distribuído pelo mundo e sem organização formal, existe um maior risco potencial de se infringirem essas leis. Apenas como exemplo, um estudo recente conduzido pela *Open Source Risk Management* mostrou que apenas o *kernel* do Linux potencialmente viola quase trezentas duzentas patentes. Esse tipo de preocupação legal, que também varia conforme o país pode prejudicar o futuro do *software* livre.

5 Conclusão

Este artigo procurou mostrar o modelo de desenvolvimento do *software* livre através da perspectiva do Trabalho Cooperativo Suportado por Computador. Essa análise ajudou a revelar um pouco mais do contexto no qual *software* livre está inserido e reforçar o argumento de que esse assunto não pode ser explicado somente por uma disciplina ou visão, mas sim através de um conjunto delas.

Através da apresentação de um *framework* conceitual, complementada com um estudo de caso do desenvolvimento do sistema operacional Linux, foram apresentados os principais elementos e inter-relações que existem em um projeto típico, com participantes, processos, ferramentas e objetos produzidos em um modelo cooperativo de trabalho.

O estudo mostrou também que o fenômeno do *software* livre, apesar de ainda estar em formação, vem crescendo bastante nos últimos anos e que sua expansão (e futuro) dependem da estabilização de alguns fatores hoje apresentados como desafios.

Referências

AOKI, A., Hayashi, K., Kishida, K., Nakakoji, K., Nishinaka, Y., Reeves, B., Takashima, A., & Yamamoto,Y. *A case study of the evolution of Jun: an object-oriented open-source 3D multimedia library*. In Proceedings of the 23rd international conference on *software* engineering (ICSE 01) (pp. 524–533). Toronto, Canadá, 2001.

Boston Consulting Group. Hacker Survey release 0.73, 2002. Disponível em www.osdn.com/bcg/BCGHACKERSURVEY-0.73.pdf . Visitado em Abril de 2007.

BERLECON, Research. *FLOSS. Free/Libre and Open Source Software: Survey and Study*. European Commission, 2002. Disponível em http://www.infonomics.nl/FLOSS/. Visitado em Abril de 2007.

BUTTON, G., Sharrock, W. *Project Work: The Organisation of Collaborative Design and Development in SoftwareEngineering*, Computer Supported Cooperative Work: The Journal of Collaborative Computing, 5, 1996, 369-386

EDWARDS, Kasper. *Epistemic Communities, Situated Learning and Open Source Software* Development Department of Manufacturing Engineering and Management, Technical University of Denmark, 2000.

KRAUT, R. E. and Streeter, L. *Coordination in SoftwareDevelopment*, Communications of the ACM, 38(3),1995, 69-81

KUWABARA, Ko. *Linux: A Bazaar at the Edge of Chaos*. FirstMonday 5(3), 2000. Disponível http://www.firstmonday.org/issues/issue5_3/kuwabara/. Visitado em Abril 2007.

MOON, J., SPROULL, L. *Essence of Distributed Work: The Case of Linux Kernel*. First Monday. 2000. Disponível em http://www.firstmonday.org/issues/issue5_11/moon/. Visitado em Abril de 2007.

ROTHFUSS, Gregor. *A Framework for Open Source Projects*, Universität Zürich, Institut für Informatik, 2002. Disponível em http://greg.abstrakt.ch/docs/OSP_framework.pdf . Visitado em Abril de 2007.

SCHARFF, E. D. *Open Source: A Conceptual Framework for Collaborative Artifact and Knowledge*. University of Colorado, 2002

SEARLS, Doc. *Linus & the Lunatics*. Transcrição de palestra no *Linux Lunacy Geek Cruise 2003.* Disponível em http://www.linuxjournal.com/article/7272. Visitado em Abril de 2007.

SHAIKH, M. Cornford, T. 2003. *Version Management Tools: CVS to BK in the Linux Kernel*, disponível em http://opensource.mit.edu/papers/shaikhcornford.pdf . Visitado em Abril 2007.

TAURION, Cezar. *Software* Livre - Potencialidades e Modelos de Negócios. Brasport, 2004.

YAMAUCHI, Y., Yokozawa, M., Shinohara, T., Ishida, T. *Collaboration with Lean Media: How Open Source Succeeds*. In: Proceedings of CSCW, 2000. ACM Press. p. 329-338.