COMO SER LIVRE ?
ORIENTAÇÕES PARA TRANSFORMAR SEU SOFTWARE DE CÓDIGO FECHADO EM UM PROJETO DE SOFTWARE LIVRE

ERS

Novembro de 2006

São Paulo, SP – Brasil

“Eu não sou um orador carismático, e se tivesse a capacidade de sê-lo, eu não seria. Eu realmente não estou interessado em persuadir as pessoas. O que gosto de fazer é ajudá-las a persuadirem a si mesmas“

Noam Chomsky

Sumário

Lista de Figuras

Capítulo 1

Introdução

Meu desejo neste trabalho é apresentar e discutir processos, modelos e engenharia em um projeto de software livre, de forma a orientar, estimular e instruir autores e líderes desenvolvedores de softwares proprietários a transformar seus softwares em projetos de software livre disponibilizando-os para a comunidade. A motivação para este trabalho veio da própria filosofia da comunidade de software livre, onde o acesso a informação é livre e compartilhado. O pensamento foi trazer projetos que estão do lado de lá para o lado de cá, ou melhor dizendo, de licença proprietária para licença livre. Exemplos como o da Netscape que liberou o código do seu navegador e o beneficio disto foi a criação do Projeto Mozilla, casos como o do banco de dados Interbase da Borland que teve seu código aberto e com isso nos proporcionou o projeto do banco de dados Firebird que hoje segue seu próprio caminho, e por último o StarOffice da Sun que após a liberação do código fonte pela empresa foi transformado pela comunidade no Projeto OpenOffice que hoje chama-se BrOffice e é um dos maiores e mais conhecidos projetos de software livre.

Por se tratar de documento restrito, foram selecionadas algumas referências bibliográficas, de onde o conteúdo foi baseado, porém, apesar do conteúdo aqui contido não possuir caráter inovador optei por tratar as informações de forma a servir a um propósito prático e atual. Trouxe para o contexto imposto pelo tema trabalhos de grande valor em pesquisa e experiência, como o são o trabalho do Softex, Unicamp e Ministério da Ciência e Tecnologia "O Impacto do Software Livre e de Código Aberto na Indústria do Software do Brasil" e o ensaio "A Catedral e o Bazar" do gringo hacker, guru e programador de software a uns 30 anos Eric S. Raymond, que marcou o início da tomada de conhecimento do modelo de desenvolvimento de software livre através da análise prática do modelo de desenvolvimento criado por Linus Torvalds para desenvolver o Linux. Também não posso deixar de citar um importante documento e simples em seu

conteúdo, produzido pelo Projeto Via Digital chamado "Gestão de Projetos de Software Livre: Uma Abordagem de Práticas". Em anexo a esta monografia inseri dois documentos interessantes, o primeiro é a Licença da Creative Commons CC-GNU GPL e o segundo é um pequeno e objetivo documento, também do Projeto Via Digital chamado "Dicas e Diretrizes para se Iniciar um Projeto de Software Livre".

1.1 Objetivos

- Organizar informações relevantes acerca de como transformar softwares proprietários em softwares livres;
- Demonstrar meios e caminhos claros que possam se tornar um estímulo para esta transformação Software Proprietário $\Longrightarrow$ Software Livre;
- Explicar e demonstrar o contexto atual do Software Livre e suas vantagens para o usuário e para o desenvolvedor.

1.2 Desafio

O desafio será organizar e selecionar a relevência de cada informação tornada disponível neste trabalho. Sempre pensando que devemos ter na cabeça quem poderá utilizá-lo e colocar em prática as informações expostas aqui. Outro desafio será conseguir contextualizar e demonstrar as vantagens e benefícios que são descritos ao longo do que já foi criado e desenvolvido em termos de software livre.

1.3 Motivação

Minha motivação, talvez o mais óbvio, para a escolha e confecção deste tema é: Quanto mais software queremos ver livres e com o código aberto? Quantas pessoas eu gostaria que soubessem sobre estes Movimentos Livres e de Colaboratividade Planetária? O quanto mais melhor, claro. Portanto a motivação é simplesmente colaborar com o Movimento de Software Livre no sentido de termos mais conteúdo colaborativo e liberdade de conhecimento.

1.4 Resultados esperados

Eu realmente esperaria que este documento fosse útil e quem o leia seja introduzido nesta possibilidade de liberar um software proprietário sob uma

licença livre, ou adquira uma maior consciência, não especificamente do Movimento Software Livre mas do movimento de colaboração coletiva e liberdade de conhecimento, pois os princípios do Software Livre são sustentados por estes conceitos.

Capítulo 2

Contexto e Panorama Geral

Este capítulo trata de contextualizar e traçar um panorama do software livre e de código aberto.

2.1 Software Livre e Código Aberto

O movimento pelo desenvolvimento de SL foi oficialmente iniciado por Richard Stallman[1] em 1983 através do manifesto GNU. Segundo Stallman, Software Livre se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software. Mais precisamente, ele se refere a quatro tipos de liberdade para os usuários do software:

- A liberdade de executar o programa, para qualquer propósito;
- A liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades. Sendo o aceso ao código-fonte um pré-requisito para esta liberdade;
- A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo;
- A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie. Sendo o acesso ao código-fonte um pré-requisito para esta liberdade.

[1] http://www.stallman.org

O termo SL trouxe problemas de ambigüidade de interpretação, principalmente quando está colocado na língua inglesa, free software, pois a palavra free tanto pode ser interpretada no contexto de ausência de custo de aquisição quanto no contexto de liberdade. No sentido de minimizar estes problemas surge, em 1998, a OSI (Open Source Initiative)[2], que passou a utilizar o termo open source (código aberto (CA)) para fornecer recomendações práticas através de critérios específicos:

- Distribuição Livre: A licença não restringirá nenhuma vontade de vender ou dar o software como um componente de uma distribuição de software agregado

  que contém programas de várias fontes diferentes. A licença não requererá royalties ou outra taxa para venda.

- Código Fonte: O programa tem que incluir código de fonte e tem que permitir

  distribuição deste código, também, em forma compilada. Onde, quando um produto não for distribuído com código de fonte, deve haver meios de obter o código fonte através da Internet sem custos. O código fonte deve estar em um formato que possibilite que qualquer programador possa alterá-lo.

- Derivação de trabalho: A licença deve permitir modificações no software bem

  como a distribuição deste sob os termos da licença original.

- Integridade do código fonte do autor: A licença pode restringir a distribuição

  do código fonte em forma modificada somente se ela oferece arquivos de correção com o código fonte com o propósito de modificar o programa em tempo de construção. A licença deve dizer explicitamente se permite a distribuição do software com o código modificado.

- Não deve haver discriminação contra pessoas ou grupos: Todas as diversidades de pessoas ou grupos devem ser consideradas em condições iguais para conseguir o benefício máximo do processo.

- Não deve haver discriminação contra empreendedores: Não poderá haver discriminação contra pessoas que queiram fazer uso do programa em seu empreendimento.

[2]http://www.opensource.org

- Distribuição da licença: As regras presas ao programa devem ser aplicadas a todos os quais o programa fora redistribuído sem a necessidade de execução de uma licença adicional para estas pessoas.
- A licença não deve ser específica a um produto: As regras presas a um programa independem de qualquer distribuição. Se o programa for extraído de uma distribuição e usado, ou distribuído dentro dos termos da licença desta distribuição, todas as partes que foram redistribuídas devem ter as mesmas regras.
- A licença não deve restringir outros softwares: As distribuições de software de código aberto têm suas próprias regras sobre seus softwares.
- A licença deve ter neutralidade de tecnologia: A licença não deve predizer o uso de alguma tecnologia específica ou estilo de interface.

Em ambos os conceitos, SL e CA, as mesmas premissas são preservadas. São mantidas as liberdades de copiar, distribuir, estudar, modificar e melhorar o software. Porém, estas premissas, em certos casos, são mais restritivas em SL quando comparadas a CA. Principalmente no que diz respeito à redistribuição de modificações efetuadas no software, pois algumas licenças de CA permitem que as mudanças efetuadas em um software CA podem ser mantidas privadas. Estas restrições e regras são estabelecidas nas licenças veiculadas junto ao software, este assunto é alvo da próxima seção. Os movimentos software livre e código aberto podem ser vistos como duas vertentes políticas internas à comunidade de SL . A comunidade SL discorda de alguns princípios da comunidade CA, principalmente com relação a algumas licenças aceitas em software de CA, porém, concorda com as recomendações práticas desta comunidade. A comunidade SL é apenas contra o software proprietário, que proíbe o livre uso, redistribuição e modificação, ou requer qualquer tipo de permissão, ou é restrito de tal forma que não se pode efetivamente fazê-lo livre .

2.2 A Catedral e o Bazar

“A Catedral e o Bazar“ foi o ensaio escrito pelo programador de software Eric S. Raymond que tornou-se um divisor de águas para o modelo de desenvolvimento de software proprietário e software livre e de código aberto.

Suas idéias definiram o Bazar como o ambiente coletivo e colaborativo, meio anáquico e onde a hierarquia é conquistada atráves do mérito (meritocracia) e a

Catedral o ambiente fechado e para poucos desenvolvedores onde o software deve atingir uma certa perfeição para ser liberado, resultando em releases longos.

Para atingir estas conclusões Raymond analisa a atitude e postura de Linus Torvalds e o modelo de desenvolvimento de software que nasceu com o Linux. Ele assume o projeto de um cliente de email chamado Fetchmail e ele mesmo passa por todas as etapas de um projeto de software livre, desde sua concepção ao reconhecimento por parte dos usuários, sempre seguindo as regras, leis adaptadas ou criadas por Linus Torvalds.

"De fato, eu penso que a engenhosidade do Linus e a maior parte do que desenvolveu não foram a construção do kernel do Linux em si, mas sim a sua invenção do modelo de desenvolvimento do Linux. Quando eu expressei esta opinião na sua presença uma vez, ele sorriu e calmamente repetiu algo que freqüentemente diz: "Sou basicamente uma pessoa muito preguiçosa que gosta de ganhar crédito por coisas que outras pessoas realmente fazem." Prequiçoso como uma raposa. Ou, como Robert Heinlein teria dito, muito preguiçoso para falhar ... para mim Linus parece ser um gênio da engenharia, com um sexto sentido em evitar erros e desenvolvimentos que levem a um beco sem saída e uma verdadeira habilidade para achar o caminho do menor esforço do ponto A ao ponto B. De fato, todo o projeto do Linux exala esta qualidade e espelha a abordagem conservadora e simplificada de planejamento do Linus."

Suas contatações passam pelas seguintes impressões e práticas, abaixo.

"Eu disse que eu decidi usar este projeto (Fetchmail) para testar minha teoria sobre o que Linus Torvalds fez corretamente. Como (você pode perguntar) eu fiz isto? Desta forma:

- 1. Eu liberei cedo e freqüentemente (quase nunca menos que uma vez a cada dez dias; durante períodos de desenvolvimento intenso, uma vez por dia).
- 2. Eu aumentei minha lista de beta testers adicionando à ela todo mundo que me contatava sobre o fetchmail.
- 3. Eu mandei extensos anúncios à lista de beta testers toda vez que eu liberava, encorajando as pessoas a participar.
- 4. E eu ouvia meus beta testers, questionando-os sobre decisões de desenvolvimento e incitando-os toda vez que eles mandavam consertos e respostas.

2.3 Modelos de Negócio e Motivos para Desenvolvimento e Uso

Baseando-me num excelente documento produzido pela Associação para Promoção da Excelência do Software Brasileiro – SOFTEX acerca do impacto do software livre e de código aberto na indústria de software no Brasil demonstro rapidamente como é o modelo de negócios e a motivação para uso e desenvolvimento de SL/CA, atualmente (2006).

2.3.1 Modelo de negócios em SL/CA

Software livre faz parte da indústria de software. Por óbvio que isso possa parecer, é importante realçar a condição de pertença do SL/CA a uma indústria, à sua indústria, que é a indústria de software. Mesmo considerando o potencial revolucionário do SL/CA, é preciso identificar com clareza as estruturas que ele revoluciona. Como modelo de negócio, o SL/CA incorpora novas formas de se fazer dinheiro na indústria de software. Sendo assim, ele altera padrões de concorrência junto à indústria que o gerou. Portanto, para identificar modelos de negócios de SL/CA é preciso antes olhar como ele altera os modelos de negócios da indústria de software. Vale também responder às seguintes questões: o SL/CA acaba com os regimes proprietários da indústria? Quais? Ele engendra novos regimes tecnológicos e novos mercados? Quais? O que deve mudar na organização do mercado de software com o crescimento do SL/CA?

As respostas que são aqui dadas a essas questões dizem mais ou menos o seguinte: o SL/CA não acaba com os regimes proprietários, mas com alguns tipos de regimes proprietários, especificamente os que combinam baixa especificidade de aplicação (programas mais genéricos, normalmente comercializados como pacotes) com elevado interesse na reprodução (cópia desejável). Em paralelo, o SL/CA não engendra propriamente novos regimes tecnológicos, não na acepção dada por Nelson & Winter (1982) ao termo, ele engendra, sim, novos rumos para velhas trajetórias e novas trajetórias dentro de um mesmo regime tecnológico. Nessa reorganização de trajetórias a indústria de software vem assumindo novos contornos e certos padrões proprietários tendem a desaparecer, particularmente os utilizados em sistemas operacionais.

Negócios com SL/CA

A bibliografia especializada aponta vários modelos de negócios, que na verdade são mais formas variadas de se fazer dinheiro com SL/CA:

- Serviço integral: negócio baseado na venda do pacote físico (CD, booklets) e na venda de todo tipo de suporte ao software (treinamento, consultoria, pré-venda, desenvolvimento customizado, pós-venda etc).
- Criação de clientela (Loss leader): negócio não está baseado no SL/CA especificamente, mas este serve para criar hábitos e preferências que depois serão úteis para a introdução de software comercial proprietário baseado no SL/CA.
- Habilitando hardware (widget frosting): uso do software livre para drivers, interfaces ou mesmo sistema operacional visando à redução de custos e de preços do equipamento a ser comercializado.
- Acessórios: venda de itens físicos relacionados ao SL/CA (hardware compatível, livros, canecas, imagens etc).
- Oferta on-line: desenvolvimento e oferta de SL/CA em sistemas on line cujo acesso é autorizado mediante pagamento de uma taxa de associação. Além disso, este modelo também apresenta ganhos com propaganda.
- Licenciamento de marcas: criam-se e licenciam-se marcas associadas a SL/CA.
- Primeiro vender, depois liberar: abertura do código após amortização dos investimentos, criando clientela para novos desenvolvimentos associados ao programa aberto.

Nestas categorias percebem-se duas coisas importantes: os focos dessas formas de negócios em serviços e em programas embarcados. Vejamos agora como se pode analisar o SL/CA desde a ótica dos modelos de negócios da indústria de software. Os desenvolvedores de uma maneira geral e os empresários desenvolvedores em particular enquadram claramente seus mercados em três principais modelos: serviços de alto e baixo valor e produto customizável. Na verdade, essa preferência é mais acentuada no segmento dos empresários que dos desenvolvedores em geral.

Se cruzarmos agora as formas de se fazer dinheiro específicas de SL/CA com os modelos de negócios da indústria de software, então teremos a perspectiva de que os maiores impactos dão-se principalmente pelos modelos de “serviço integral“ e “habilitando hardware“ Embora “vender e liberar“ (que é complementar ao modelo “criação de clientela“) e “acessórios“ impactem na indústria, é nos dois primeiros modelos que se localizam os principais impactos. É significativo o impacto do modelo “serviço integral (cujos exemplos são Conectiva, Red Hat e

IBM) em praticamente todos os padrões de concorrência da indústria em geral. Da mesma forma, mas com mais especificidade, o modelo "habilitando hardware" abre oportunidades para software embarcado, exatamente porque reduz custos e amplia as possibilidades de comercialização de hardware.

2.3.2 Motivos para uso e desenvolvimento de SL/CA

A literatura sobre o tema "motivações de usuários e desenvolvedores de SL/CA" é vasta. Em geral, as opções de motivações apresentadas aos entrevistados envolvem aspectos técnicos, ideológicos, sociológicos e econômicos.

Figura 2.1: Razões de desenvolvimento e/ou distribuição de SL/CA, respondido por desenvolvedores.

Foram investigadas as motivações nas três principais fontes de informação deste estudo: painel, enquete com desenvolvedores e usuários e entrevistas com empresas. Na enquete com desenvolvedores, foram a eles colocadas questões específicas sobre motivações.

A grande maioria dos respondentes destacou opções relacionadas à capacitação, como "desenvolver novas habilidades" e "compartilhar conhecimento". Em

seguida aparece uma motivação de ordem técnica: “resolver problema sem solução com o software proprietário“. Segue-se uma motivação de natureza ideológica: “software não deve ser proprietário“. Vale ainda comentar duas motivações de natureza econômica: “melhor empregabilidade“ e “aplicações comerciais“, ambas com baixa colocação no ranking da enquete. Já para os usuários os resultados foram um pouco diferentes.

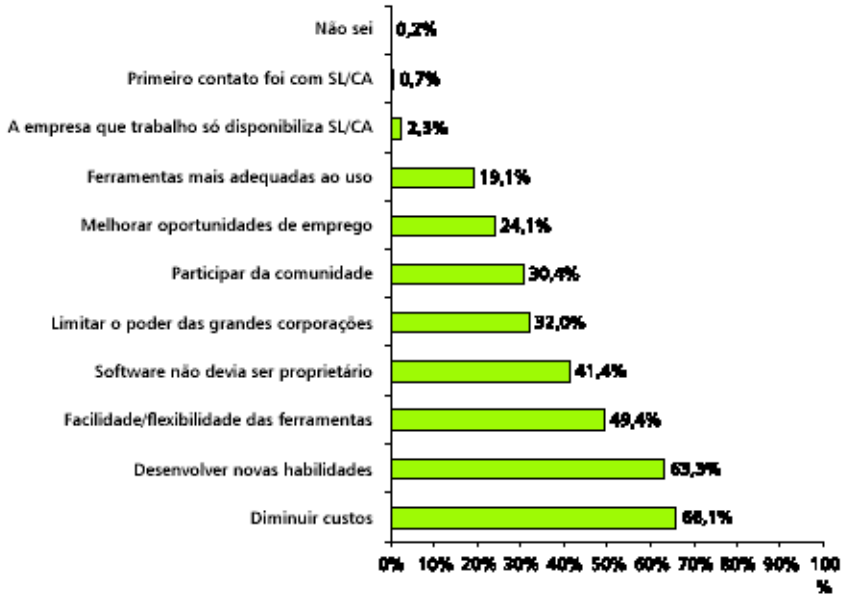

Figura 2.2: Razões de desenvolvimento e/ou distribuição de SL/CA, respondido por usuários.

“Reduzir custos“ e “desenvolver novas habilidades (capacitação)“ foram as razões com maior freqüência de respostas. As demais motivações são uma mistura de razões técnicas (“facilidade das ferramentas“) com ideológicas (“software não deve ser proprietário“ e “limitar o poder das grandes corporações“). É interessante notar que as razões ideológicas são mais fortes entre usuários que entre desenvolvedores, provavelmente porque estes têm no desenvolvimento de programas um elemento fundamental de sustentação financeira. Ainda que o componente ideológico seja importante, há razões de natureza prática que os tornam mais pragmáticos.

2.4 Casos de Sucesso

Neste próximo tópico são apresentados casos de sucesso de softwares que anteriormente proprietários liberaram seus códigos fonte sob uma licença livre. Estes softwares representam hoje apenas uma pequena parcela de mercado e área de atividade, porém possuem uma base instalada bastante representativa.

2.4.1 Mozilla

O lançamento do Netscape Communicator open source, que veio na altura do crescimento econômico tardio dos anos 90, foi elogiado pela comunidade com uma mistura de aclamação e ceticismo. Em alguns círculos, a liberação dos fontes por parte da Netscape foi vista como uma vitória para o movimento do software livre e uma oportunidade para a Netscape explorar o poder do desenvolvimento open source. Esta visão era particularmente popular entre usuários Linux e de outros programas livres. Outros observadores incluindo muitos da comunidade de negociantes não-open-source interpretaram o movimento como uma rendição da Netspace frente à ascenção crescente do legalmente-criticado navegador Internet Explorer da Microsoft.

A despeito da opinião pública, o desenvolvimento sobre a base de código antiga do Communicator provou ser mais complicado do que o esperado.

- A base antiga era grande e complexa.
- Ela tinha que ser desenvolvida simultaneamente em muitos sistemas operacionais e, portanto, lidar com as diferentes bibliotecas e idiossincrasias desses sistemas.
- Ela tinha as marcas de muitos ciclos rápidos de desenvolvimento em código fechado no chamado “Tempo Internet“. Esses ciclos curtos de desenvolvimento fizeram com que os programadores sacrificassem modularidade e elegância para conseguir implementar mais opções no programa no tempo limitado que tinham.
- Várias partes do código do Communicator nunca foram liberadas como código aberto por causa de acordos de licenciamento feito com terceiros.

Como resultado, a primeira versão de código aberto do Communicator nem mesmo compilava direito, e simplesmente não rodava. Isso se transformou em um enorme desafio para os desenvolvedores principais do Mozilla (muitos dos quais ainda estavam na folha de pagamento da Netscape), e um desafio ainda maior

para os desenvolvedores independentes querendo contribuir para a aplicação por si próprias.

Finalmente, os desenvolvedores principais do Mozilla concluíram que a antiga base de código não podia ser salva. Eles decidiram ignorá-la e reescrever todo o sistema do nada, uma decisão que levou um dos líderes do desenvolvimento da Netscape, Jamie Zawinski, a se demitir. O plano resultante incluiu, entre outras coisas, a criação de uma biblioteca de interface gráfica multi-plataforma completamente nova e um novo mecanismo de renderização de HTML.

Poucos observadores conseguiram prever o resultado. Em 7 de Dezembro de 1998 menos que dois meses depois do anúncio de 26 de Outubro de 1998 declarando que a base de código antiga do Communicator seria descartada a Netscape liberou uma versão de "demonstração" inicial baseada no novo mecanismo de renderização chamado Gecko. O Gecko já estava sob desenvolvimento por algum tempo dentro da Netscape sob o nome interno NGLayout ("Next Generation Layout"), e era notavelmente mais rápido e menor do que seu predecessor. Uma de suas características mais publicitadas na sua primeira versão era que ele cabia em um único disquete de 1.44 MB, um tamanho cerca de 10 vezes menor do que seus contemporâneos.

A rápida liberação da primeira versão do Gecko levou muitos a acreditarem que um navegador completo não estaria muito longe. Entretanto, a primeira versão do mecanismo de renderização estava longe de ser estável, e mais longe ainda de estar pronta para ser implementada em um navegador. Além disso, produzir um navegador com todas as características necessárias requeria muito mais do que apenas um mecanismo de renderização nascente: os desenvolvedores do Mozilla logo visionaram um projeto muito mais ambicioso do que um simples navegador. O novo Mozilla seria uma plataforma para aplicações Internet, com um interface gráfica de usuário completamente programável e uma arquitetura modular. Este Mozilla funcionaria tanto com um hospederio para clientes de e-mail, instant messaging, clientes NNTP, ou qualquer outro tipo de aplicação similares.
Por causa do esforço requerido por essa rescriva massiva, o projeto se distanciou rapidamente de suas metas projetadas. Nos anos que se seguiram, o ceticismo sobre o Mozilla cresceu, e muitos duvidaram que um navegador terminado veria a luz do dia. Entretanto, o projeto persistiu, continuando ininterrupto mesmo em face da aquisição da Netscape pela AOL e o fim do crescimento econômico excessivo e insustentável propiciado pela Internet nos anos 90.

Por volta de Junho de 2002, o projeto Mozilla havia produzido um navegador útil e conformante com os padrões web, que funcionava em múltiplos sistemas operacionais incluindo Linux, Mac OS, Windows e Solaris. A versão 1.0 do

Mozilla, liberada em 5 de Junho de 2002 foi elogiada por introduzir características que nem mesmo no seu rival, o Internet Explorer, possuía, incluindo um suporte melhor para as preferências de privacidade do usuário e algumas melhorias na interface gráfica. Adicionalmente, o navegador tornou-se a implementação de referência de facto dos vários padrões do World Wide Web Consortium, devido ao seu forte suporte aos mesmos. Versões atuais do Mozilla são altamente customizáveis e incluem características avançadas como gerenciamento de cookies, popups, senhas e imagens, e também o famoso "tabbed browsing".

Em 5 de Julho de 2003, a AOL anunciou que fecharia a sua divisão de navegadores, o que, em essência, era o Mozilla. Longe de ser o fim da platafoma, este foi o começo da Fundação Mozilla, formada por ex-veteranos da Netscape/Mozilla que tomaram sobre si a responsabilidade do desenvolvimento do projeto. Como consolação, a AOL prometeu investir 2 milhões de dólares da recém-formada fundação. Muitas pessoas já estavam esperando que isso acontecesse quando a AOL fechou um acordo com sua competidora, a Microsoft, em um negócio que permitia que a AOL embutisse o Internet Explorer em seus produtos pelos próximos 7 anos. A Netscape sempre foi vista como um item de barganha da AOL em relação à Microsoft.

A AOL demitiu a maior partes dos empregados e desenvolvedores da Netscape, alguns dos quais foram transferidos para outras divisões da empresa. Os símbolos da Netscape foram removidos dos prédios que ela ocupava, confirmando o que muitos tomaram como o fim da divisão. A AOL continuaria a manter a marca Netscape em seu portal, mas a companhia não mais pagaria desenvolvedores para trabalhar na base de código do projeto. Versões futuros do Netscape seriam apenas versões renomeadas do Mozilla, começando com a versão 7.2 (baseada, tentativamente, na versão 1.7) do Mozilla.
Mozilla, um produto que originalmente visava desenvolvedores ao invés de usuários finais, agora enfrenta o desafio do marketing para as massas. [3]

2.4.2 StarOffice/OpenOffice/BrOffice

A origem do BrOffice.org remonta a meados da década de 90, quando a empresa alemã Star Division criou um pacote de escritório chamado StarOffice e começou a distribuí-lo gratuitamente para as plataformas Windows e Linux. Em 1999, a Star Division foi adquirida pela empresa americana Sun Microsystems. Logo após lançar o StarOffice 5.2, em 13 de Outubro de 2000, a Sun Microsystems doou parte do código fonte do StarOffice para a comunidade de código aberto, tornando-se colaboradora e patrocinadora principal do recém lançado projeto

[3]Fonte: http://www.wikipedia.org

OpenOffice.org. A iniciativa ganhou o apoio de diversas organizações do mundo tecnológico como Novell, Red Hat, Debian, Intel, Mandriva, além das importantes contribuições de desenvolvedores independentes, ONGs e agências governamentais. Essa comunidade, formada por programadores e usuários do mundo inteiro, é quem desenvolve o pacote desde então. Todos fazendo com que o OpenOffice.org não seja apenas uma alternativa livre em suítes de produtividade, mas a melhor e a mais avançada solução de automação de escritórios. Além, é claro, de uma formidável comunidade colaborativa.

Durante todo esse tempo, são quase 50 milhões de downloads contabilizados, dos mais de 40 idiomas diferentes nos quais o OpenOffice.org está disponível. Ao ser disponibilizado sobre as plataformas GNU/Linux, Windows, Sun Solaris e Mac OS X (X11), entre outras, o OpenOffice.org rompeu a barreira da conectividade, integrando usuários dos mais variados perfis e estabelecendo o Software Livre como uma alternativa concreta no até então invariável mercado de aplicativos para usuários finais.
No Brasil, uma comunidade de voluntários se formou com a missão de adaptar o OpenOffice.org para o português brasileiro. Em feveiro de 2002, Raffaela Braconi, líder internacional da equipe do projeto L10N na época, repassou a função de coordenação da tradução para Claudio Ferreira Filho. Além de Claudio Ferreira, entre os primeiros colaboradores do projeto estavam César "Guanch" Melchior, Olivier Hallot e Gervásio Antônio. A esse grupo foi destinada a primeira grande tarefa do projeto, a tradução do glossário padrão, que daria o subsídio para a compilação das primeiras versões do OpenOffice.org em português do Brasil.

A partir de então, além da tradução, o projeto OpenOffice.org.br passou a organizar e desenvolver funcionalidades específicas para a versão brasileira do pacote. Foram criadas as listas de discussão, o projeto de Documentação, o Rau-tu, o projeto Extras e finalizadas as traduções das aplicações e da ajuda do software. O período coincide, também, com a organização de comunidades de Software Livre espalhadas por todo o país. Pela sua popularidade e organização o projeto OpenOffice.org.br passou a ser uma das referências dentro do cenário do Software Livre brasileiro, disseminando a utilização do pacote de aplicativos para usuários, empresas, entidades governamentais e organizações em geral. O novo BrOffice.org Em 2004, no entanto, devido a problemas com a marca Open Office, registrada anteriormente por uma empresa do Rio de Janeiro, foi necessário trocar o nome da comunidade e do produto. Surgiu assim o BrOffice.org. No dia 25 de janeiro de 2006, foi anunciado oficialmente o lançamento da ONG BrOffice.org que passou a organizar as atividades da comunidade OpenOffice.org.br. Apesar da mudança de nome, o BrOffice.org continou representando o OpenOffice.org, com a garantia de todos os instrumentos

jurídicos de proteção à marca BrOffice.org.

A missão definida para a ONG alinhou-se às atividades da comunidade já em curso e inclui apoiar e desenvolver ações para fomentar a comunidade brasileira do BrOffice.org e seus projetos relacionados. Entre os objetivos da ONG BrOffice.org incluem-se a difusão do Software Livre e de Código Aberto, a sustentação do projeto BrOffice.org e a promoção do voluntariado. Além disso, a criação da ONG BrOffice.org permitiu ao projeto relacionar-se com outras figuras jurídicas na forma da lei, seja através de contribuições financeiras, de equipamentos ou recursos em geral ou, ainda, através de projetos contratados junto a ONG, desde que alinhados com a missão e objetivos definidos em seu estatuto.

O anúncio foi acompanhado por diversas modificações na estrutura do projeto. Além da formalização através da ONG, o portal do projeto e a configuração do servidor foram totalmente remodelados. Essas ações foram motivadas pela necessidade de prover a estrutura necessária para o desenvolvimento das versões brasileiras do pacote, com recursos diferenciados em relação ao OpenOffice.org original. O projeto BrOffice.org está pronto para o futuro. Mais aberto, mais funcional e mais interoperável. Continuaremos trabalhando para disponibilizarmos aplicações de qualidade e ampla utilização, acreditando no Software Livre e na força do trabalho colaborativo. Você também está convidado. Junte-se a nós e ajude a escrever parte dessa história! [4]

2.4.3 Firebird

O Firebird é baseado no código fonte do InterBase 6.0 que foi liberado como Open Source pela Borland em Agosto de 2000. A história do Interbase remonta aos idos de 1984, portanto, são cerca de 20 anos de experiencia com base de dados relacional no produto.

O Interbase, o Open Source, e o nascimento do Firebird

De acordo com Ann Harrison, o Interbase nasceu um dia debaixo de um chuveiro: reza a lenda que Jim Starkey estava a tomar ducha quando teve a ideia de criar uma base de dados relacional, que eventualmente se tornou o Interbase, e mais tarde, o Firebird. Jim Starkey criou uma pequena equipe com Don De Palma e Ann Harrison, e juntos criaram uma empresa, a Groton database Systems, que iniciou o desenvolvimento do Interbase. Estávamos em 1984. Em 1986, a empresa muda de nome para Interbase, e sai a versão 2 do Interbase. A Ashton-

[4]Fonte: http://www.broffice.org.br

Tate torna-se um dos investidores, e em 1988 surge a versão 3. Ainda neste ano, a Ashton- Tate reforçou a sua posição na companhia, e adquire 51% da mesma. Anos mais tarde, em 1991, já é detentora de todo o capital. A versão 4 do Interbase surge em 1994. No ano seguinte, a Borland já comercializa a versão 4 para Windows NT e Novell Netware, e a versão Delphi 1 Client/Server inclui o Interbase 4.0C. No mesmo ano surge ainda a primeira versão do InterClient.

Em 1997, a Borland cria uma subsidiária, a ISC, que se dedicará a expandir o negócio do Interbase. Ainda neste ano, a versão Delphi 3 Client/Server é comercializada com o Interbase 4.2. E ainda no final de 1997, em Dezembro, é lançada a versão 5. Em 1998, a Borland muda o nome para Inprise Corporation. Em Agosto, é lançado o Delphi 4, que inclui a versão 5.1.1. do Interbase. Nesse ano é lançada também a versão 5 para Linux. Em 1999, a versão do Borland C++ Builder 4 é comercializada com o Interbase 5.5. Começam os trabalhos no desenvolvimento da versão 6. Mas é em Dezembro de 1999 que tudo se altera: Bill Karwin, Paul Beach and Wayne Ostiguy despedem-se da ISC, e todos os trabalhos do Interbase são congelados. Um pouco por todo o lado começam os pedidos para não deixar o Interbase morrer: Helen Borrie cria a "lista SaveInterbase" para evitar a morte do Interbase; é criado o grupo IBDI (Interbase Developers Initiative) por Helen Borrie, Jason Wharton e Dalton Calford para salvaguardar os interesses dos programadores em Interbase. A 3 de Janeiro de 2000, a Borland anuncia que vai tornar Open Source o Interbase, e desafia outras empresas a seguir o mesmo rumo. A 14 de Fevereiro, Ann Harrison aceita o desafio de ser a nova presidente da ISC, tendo Jim Starkey como consultor técnico. Mas o Interbase 6 não é lançado na data prevista: 30 de Junho. Começa a confusão: a Inprise (mas não a ISC) lança ao público em 25 de Julho o código do Interbase 6. E nem tudo são rosas: este lançamento não traz a documentação, não traz as mais básicas ferramentas de testes, e... não são permitidas alterações ao código por ninguém externo à companhia! Em resumo: embora toda a gente tenha acesso ao código, este não está documentado, e ninguém pode alterar o mesmo, apenas "sugerir" alterações. Dias depois, a 28 de Julho, Ann Harrison demite-se. O negócio entre a ISC e a Borland morre. A Borland decide reter o controle do produto pelo interesse dos accionistas.

A 23 de Julho de 2000, um grupo de entusiastas decide criar a Firebird Tree no Sourge Forge: já que a Borland não permite alterações, existe agora um espaço aberto a todos. A história vai continuando com uns imprevistos que não caem nada bem: a 10 de Janeiro de 2001 a CERT anuncia uma falha na segurança do Interbase que existe desde 1994: dentro do código, existe um user e uma password pré-definidos que permitem o acesso a qualquer base de dados, com direitos de SYSDBA: uma situação nada "politically correct". São lançados

"patches" para corrigir este problema.

Em 14 de Março de 2001, a Borland regressa com versões comerciais pagas. Existem assim duas versões do IB 6 no mercado: a Open Source, à qual a Borland não presta qualquer tipo de apoio, e a versão comercial paga, a qual foi sujeita a testes intensivos por parte da Borland e à qual os utilizadores podem recorrer à Borland para suporte. Entretanto a equipe do Firebird vai fazendo o seu trabalho independentemente de todas estas confusões e desentendimentos. A equipe assume o seu trabalho muito profissionalmente: os bugs vão sendo corrigidos, as versões Beta vão saindo com bastante frequência. Esta equipe não tem qualquer vínculo comercial, pelo que tudo é feito com base apenas na qualidade do produto final: depois da versão beta surge a primeira "Release Candidate" (RC1) e uns tempos depois a "Release Candidate 2". Milhares de programadores por todo o mundo participam nos testes destas versões. A 4 de Dezembro de 2001 a Borland anuncia o Interbase 6.5. Que se desenganem os puristas nada do que existe na versão 6.5 está disponível na versão open source. A 12 de Março de 2002 é lançada a versão 1.0 do Firebird. O número de downloads desta versão, só nas primeiras semanas, atingiu as dezenas de milhares.[5]

2.4.4 Compiere

O Compiere é um sistema de CRM+ERP de código fonte aberto e livre (SL/CA) feito em Java que já existe no mercado desde 2000. Esse projeto foi patrocinado pela Goodyear da Alemanha em 2000. O Compiere é um dos projeto mais ativos no sourceforge com mais de um milhão de downloadas. O foco do produto é em pequenas e médias empresas. No entanto, ele usa o Oracle como banco de dados, visto que o desenvolvedor principal veio da Oracle. Isso é uma barreira para a realidade das pequenas e médias empresas brasileiras porque o Oracle tem um custo muito alto e assim acaba impossibilitando o uso do Compiere por empresas sem grandes possibilidades de investimentos na área de tecnologia de informação. Porem em 2003 a Finep, órgão ligado ao Ministério da Ciência e Tecnologia, lançou um edital para apoio em projetos de sotware livre. Uma empresa, em parceria com a Unicamp, enviou o projeto para adaptação do Compiere para usar o PostgreSQL. Esse projeto foi aceito e durou um ano, portanto hoje emdia temos o Compiere um sistema de CRM+ERP livre rodando sob banco de dados PostgreSQL também livre.[6]

[5] http://www.comunidade-firebird.org

[6] Fonte: http://www.compierebrasil.com.br e http://www.compiere.org

Capítulo 3

Modelo de Projeto de Software Livre

"O modelo SL/CA traz uma nova forma de desenvolver e licenciar software que está quebrando modelos tradicionais de apropriabilidade e de desenvolvimento tecnológico" Arthur Pereira Nunes – Softex/Unicamp/MCT

As informações deste capítulo foram em grande parte baseadas no trabalho "Gestão de Projetos de Software Livre: Uma Abordagem de Práticas", desenvolvido pelo Portal Via Digital – Caminho Inteligente para a Informatização Pública (http://www.viadigital.org.br).

3.1 Participantes

Em um primeiro momento, é importante entender quais são as pessoas que podem fazer parte de uma comunidade de SL, que papéis podem assumir e como podem contribuir para a realização de um projeto de SL. Partindo de uma observação em diversos projetos de SL , constatou-se que não existe uma terminologia comum para definir os participantes de uma comunidade de SL. Diferentes projetos, com diferentes perfis, podem fazer uso de um grupo grande de participantes com vários papéis definidos. Outros projetos podem fazer uso de um esquema simples de definição de papéis. Desta maneira, para prover um resultado mais amplo, enumeramos os possíveis papéis encontrados em análises realizadas sobre diferentes projetos de SL.

3.1.1 Usuários passivos

São usuários que apenas fazem uso do software sem contribuição direta para a comunidade, são atraídos principalmente pela qualidade dos projetos de SL e a

potencialidade de modificações conforme suas necessidades.

- Responsabilidades: fazem download e uso do software livre.
- Privilégios: acesso de leitura ao repositório de dados do projeto.

3.1.2 Usuários ativos

Assim como usuários passivos, são usuários de uma solução de software oferecida por alguma comunidade. No entanto, estes usuários têm um posicionamento de maior participação na comunidade, não fazendo apenas download e uso da ferramenta.

- Responsabilidades: participam de discussões nas listas e fornecem opiniões, principalmente, sobre funcionalidades do software.
- Privilégios: envio de mensagens em listas de discussões, acesso de leitura ao repositório de dados do projeto.

3.1.3 Relatores de bugs

Participam do desenvolvimento do software, contribuindo com a localização de problemas e a especificação destes problemas em sistemas destinados para esta atividade.

- Responsabilidades: descrever problemas encontrados no software, que tanto podem ser problemas encontrados durante o uso do software, quanto problemas encontrados no código fonte.
- Privilégios: envio de mensagens em listas de discussões, acesso ao sistema de rastreamento e acompanhamento de modificações para envio de reportes de erros, acesso de leitura ao repositório de dados.

3.1.4 Corretores de bugs

Enviam correções de bugs cadastrados na base de dados para membros de núcleo do projeto ou desenvolvedores ativos para que seja avaliada a qualidade da correção efetuada e se a mesma poderá ou não compor o projeto. Pessoas que se enquadram neste possuem certo conhecimento sobre código do projeto.

- Responsabilidades: recuperar problemas cadastrados no sistema de rastreamento e acompanhamento de mudanças do projeto, realizar as correções inerentes ao reporte de erro selecionado, enviar trechos corrigidos

para avaliação (podendo utilizar o sistema de rastreamento e acompanhamento de mudanças para isto).

- Privilégios: envio de mensagens em listas de discussões, acesso ao sistema de rastreamento e acompanhamento de modificações para envio de correções de reportes de erros, acesso de leitura ao repositório de dados.

3.1.5 Desenvolvedores periféricos

São desenvolvedores que contribuem de maneira ocasional para o projeto, não estando totalmente comprometidos com a comunidade. As pessoas que executam este papel entendem o funcionamento do projeto (em termos de implementação) e contribuem, principalmente, com implementações de funcionalidades isoladas para o projeto. Não costumam se envolver em longo período de tempo com o projeto.

- Responsabilidades: desenvolvimento de soluções pontuais para o projeto, participação em discussões sobre o desenvolvimento, adicionamento de novas funcionalidades para o projeto, envio de funcionalidades implementadas para membros mais antigos do projeto para avaliação.

- Privilégios: envio de mensagens em listas de discussões, acesso ao sistema de rastreamento e acompanhamento de modificações para envio de correções de reportes de erros, acesso de leitura ao repositório de dados.

3.1.6 Testadores

Realizam testes de software, visando aferir a qualidade do software construído pela comunidade. Podem realizar suas atividades através do uso de ferramentas para automação de testes.

- Responsabilidades: manutenção da qualidade do sistema através da realização de testes: construção e realização de testes de unidade, construção e realização de testes funcionais, realização de testes baseados em entradas e saídas.

- Privilégios: envio de mensagens em listas de discussões, acesso ao sistema de rastreamento e acompanhamento de modificações para envio de correções de reportes de erros, acesso de leitura e escrita ao repositório de dados.

3.1.7 Desenvolvedores ativos

São desenvolvedores que contribuem de maneira constante para o projeto, tendo responsabilidade por grande parte do código fonte construído para o mesmo. As pessoas que executam este papel têm grande conhecimento do funcionamento do projeto. Envolvem-se com o projeto durante períodos de tempo longos.

- Responsabilidades: desenvolvimento de soluções para o projeto, participação em discussões sobre o desenvolvimento, adicionamento de novas funcionalidades para o projeto, acréscimo de funcionalidades implementadas ao projeto, revisão de código de outros desenvolvedores, integração de artefatos no repositório de dados.

- Privilégios: envio de mensagens em listas de discussões, acesso privilegiado ao sistema de rastreamento e acompanhamento de modificações para envio de correções de reportes de erros, acesso de leitura e escrita ao repositório de dados.

3.1.8 Documentadores

Escrevem documentos e manuais para o projeto. Estes artefatos podem fazer referência tanto ao projeto quanto à comunidade que o constrói. Em relação ao projeto são produzidos documentos como: manuais de utilização, instruções de instalação e configuração, apresentação das funcionalidades do software, etc. Em relação à comunidade são produzidos documentos como: informativos de como participar da comunidade ou contribuir com o projeto, políticas e regras utilizadas pela comunidade, padrões de codificação utilizados, etc.

- Responsabilidades: elaboração de documentos de referência para a comunidade e para o projeto, tornando estes artefatos disponíveis na página, ou no repositório de dados do projeto, na forma de tutoriais, FAQs, HowTos ou manuais.

- Privilégios: envio de mensagens em listas de discussões, acesso de leitura e escrita ao repositório de dados, acesso de escrita aos arquivos da página de acompanhamento do projeto.

3.1.9 Tradutores

Traduzem artefatos construídos pela comunidade de SL para outros idiomas. Estes artefatos incluem: o próprio software (ênfase à internacionalização da interface com o usuário), a página de acompanhamento do projeto e a

documentação pertinente ao sistema e/ou comunidade (manuais, FAQs, HOWTOs, tutoriais, regras da comunidade, etc.).

- Responsabilidades: tradução dos artefatos produzidos para o projeto segundo as estratégias estabelecidas para realização da tradução.
- Privilégios: envio de mensagens em listas de discussões, acesso de leitura e escrita ao repositório de dados, acesso de escrita aos arquivos da página de acompanhamento do projeto, acesso ao sistema de rastreamento e acompanhamento de modificações para envio de correções de reportes de erros.

3.1.10 Membros de núcleo

Fornecem a maioria das contribuições do projeto. Geralmente participam do projeto desde seu início, ou já possuem grande experiência nele, que pode ser obtida através do tempo e da consistência de suas interações com a comunidade. Na maioria dos casos, o número de membros de núcleo em um projeto não é grande (não chega a mais de dez pessoas). No entanto, em grandes projetos, como o caso do Mozilla, este grupo pode chegar a mais de vinte pessoas.

- Responsabilidades: coordenação das atividades dos desenvolvedores do projeto, definição de metas estratégicas para o projeto, desenvolvimento de código fonte, avaliação de contribuições dadas por outros desenvolvedores, decidir aspectos funcionais relevantes para o projeto, ajudar na definição de prioridades para o projeto, podem dar maior nível de privilégios para outros membros da comunidade.
- Privilégios: acesso privilegiado a todos os recursos, poder de decisão sobre assuntos relativos ao projeto.

3.1.11 Líderes de módulos/subsistemas

São pessoas responsáveis por módulos/subsistemas do software produzido por uma comunidade. Centralizam decisões tomadas nos subsistemas do software, decidindo que alterações ou funcionalidades podem ser somadas ao subsistema. Suas atividades diminuem a carga de responsabilidade jogada para os membros de núcleo e a liderança do projeto.

- Responsabilidades: suas responsabilidades concentram-se em relação ao subsistema do projeto sob sua liderança. Têm por responsabilidade: coordenação das atividades de desenvolvimento, recebimento e filtragem de

pedidos de correção de funcionalidades, decisão de que alterações são prioritárias e quais devem ser descartadas, podem dar maior nível de privilégios para desenvolvedores que contribuem para a evolução do subsistema sob sua responsabilidade.

- Privilégios: acesso privilegiado a todos os recursos, poder de decisão sobre assuntos relativos ao projeto.

3.1.12 Conselheiros/Patrocinadores

Alguns projetos de SL podem ser patrocinados por empresas ou organizações externas (como a IBM e a HP, que patrocinam diversos projetos de SL). Desta maneira, membros destas entidades podem participar da comunidade como conselheiros, participando ativamente do processo de decisão dos rumos do projeto, na medida em que interesses de uma empresa podem estar envolvidos com a evolução do software.

- Responsabilidades: discutir os rumos que o projeto deve tomar para satisfazer necessidades específicas de uma organização externa.
- Privilégios: acesso privilegiado a todos os recursos, poder de decisão sobre assuntos relativos ao projeto.

3.1.13 Líderes de Projeto

Pessoas que estão à frente dos projetos de SL. Líderes, geralmente, são pessoas que iniciaram um projeto de SL, porém esta liderança pode ser repassada a outras pessoas, fato que pode ocorrer caso o líder de um determinando projeto não possa executar de forma satisfatória suas funções. O seu trabalho torna-se evidente quando a comunidade está organizada sob o modelo de liderança centralizada (ditador benevolente). O que o faz centralizador das decisões tomadas para a evolução do projeto.

- Responsabilidades: gerência de pessoas, tarefas, recursos, prioridades e rumos de todo o projeto.
- Privilégios: acesso privilegiado a todos os recursos, poder de decisão sobre assuntos relativos ao projeto, poder de fornecer privilégios para outros participantes.

3.1.14 Membros de Comitê Administrativo

Membros de comitês/conselhos administrativos agem em conjunto no sentido de gerenciar a comunidade. Eles basicamente executam as mesmas funções de um líder único de projeto, no entanto dividem as responsabilidades entre si. Portanto devem existir políticas internas à comunidade para escolha destes membros e padrões para tomada de decisões na comunidade (tal como a decisão baseada em votos).

- Responsabilidades: gerência de pessoas, tarefas, recursos, prioridades e rumos de todo o projeto.
- Privilégios: acesso privilegiado a todos os recursos, poder de decisão sobre assuntos relativos ao projeto, poder de fornecer privilégios para outros participantes.

3.2 Ferramentas Utilizadas

O uso de ferramental para o desenvolvimento de software é uma das características mais marcantes em projetos de SL. De tal forma, o uso de um conjunto mínimo de ferramentas é importante para prover um ambiente de interação necessário ao desenvolvimento de projetos de SL. Assim, as ferramentas utilizadas podem ser classificadas em: ferramentas de comunicação, ferramentas de apoio ao processo, ferramentas de controle de qualidade e ferramentas para provimento de um ambiente colaborativo (integração de ferramentas destinadas ao desenvolvimento de SL em um único pacote).

3.2.1 Comunicação

Listas de discussão

Canais de comunicação utilizados pelas comunidades baseados na captura de mensagens enviadas pelos participantes e armazenamento das mesmas. As mensagens são dispostas em forma de texto simples sem formatação e estão acessíveis através da Internet, na forma de arquivos de discussões realizadas. Listas de discussão são as ferramentas primordiais em comunidades de SL, na medida em que podem ser usadas para diferentes finalidades: discussão de requisitos, votações, resolução de conflitos anúncios de novas versões, servir como documentação para novos usuários e desenvolvedores, etc.

↪ Exemplos de ferramentas:
– Mailman (http://www.gnu.org/software/mailman/)

– Sympa (http://www.sympa.org/)
– Majordomo (http://www.greatcircle.com/majordomo/)
↪ Papéis que fazem uso das listas de discussão do projeto:
– Todos.
↪ Restrições:
– Requer que os participantes estejam devidamente inscritos para postagem de mensagens.

Wiki

WikiWikiWeb, mais comumente conhecido por Wiki, é uma ferramenta colaborativa de edição de páginas Web, a qual é diretamente realizada através dos navegadores dos usuários. Este sistema permite que sejam criados e editados coleções de documentos inter-relacionados através de uma linguagem de script. A essência do sistema é permitir que qualquer um edite páginas através da Web para geração de conteúdo dinâmico. A maior vantagem do uso deste tipo de sistema está na ausência de necessidade de uso de um software no lado do cliente para edição de páginas HTML, toda esta tarefa é feita diretamente do navegador.

↪ Exemplos de ferramentas:
– TWiki (http://www.twiki.org/)
– WikiWeb (http://www.wikiweb.com/)
↪ Papéis que fazem uso de Wiki:
– Todos.
↪ Restrições:
– Em alguns casos os sistemas Wiki podem requerer algum controle de acesso e só permite a edição de conteúdo para usuários autorizados.

Ferramentas de mensagens instantâneas (Instant Messengers)

Ferramentas de mensagens instantâneas permitem que pessoas realizem comunicação, baseada em troca de mensagens textuais, em tempo real, através de uma rede de comunicação. Várias ferramentas podem ser utilizadas como clientes de mensagens instantâneas, nas comunidades de SL a mais popular entre elas é o uso de IRC (Internet Relay Chat). Desta maneira, as comunidades mantêm canais de bate-papo em servidores de IRC onde os participantes das comunidades trocam e amadurecem idéias sobre o projeto.

↪ Exemplos de ferramentas:
– mIRC (http://www.mirc.com/)
– Gaim (http://gaim.sourceforge.net)
↪ Papéis que fazem uso de Instant Messengers:

– Todos.
↪ Restrições:
– Não há restrição para uso destas ferramentas e participação de conversas em canais. Como os canais de IRC possuem moderadores, pode ocorrer o bloqueio da participação de algumas pessoas. No entanto, esse bloqueio geralmente ocorre em casos extremos, quando um participante agride outro, por exemplo. A regra geral é a permissão de participação de qualquer pessoa.

Página do projeto

Primordialmente as comunidades de SL utilizam ferramentas de apoio à comunicação, tais como listas de discussões, ou ferramentas de apoio ao desenvolvimento, tais como um sistema de gerência de configuração. No entanto, as páginas dos projetos, principalmente com a massificação da Web como meio de suporte à comunicação realizada por comunidades de SL, tornaram-se ferramenta essencial para difundir informações pertinentes às comunidades de SL: sumário do projeto, guias para os usuários, informações sobre membros fundadores e participantes da comunidade, detalhes sobre a licença do projeto, dicas para participar da comunidade, entre outros.

↪ Exemplos de ferramentas para construção de páginas:
– Bluefish (http://bluefish.openoffice.nl/)
– Nvu (http://www.nvu.com/)
↪ Exemplos de ambientes que permitem disponibilizar páginas de projetos de SL:
– SourceForge (http://sourceforge.net/)
– Tigris.org (http://tigris.org)
– CódigoLivre (http://codigolivre.org.br/)
↪ Papéis que fazem têm acesso de escrita aos arquivos das páginas do projeto:
– Tradutores, documentadores, membros de núcleo, líderes de subsistemas, membros, líderes de projetos, membros de comitê administrativo.
↪ Restrições:
– A edição do conteúdo das páginas é realizada por participantes com acesso de escrita às mesmas. O acesso à página do projeto é irrestrito.

3.2.2 Apoio ao processo de desenvolvimento

Gerência de configuração

As ferramentas de gerência de configuração garantem a realização de trabalho seguro e consistente em um ambiente onde o desenvolvimento é dado de forma descentralizada e paralela. Nos projetos de SL a ferramenta padrão para realizar

esta tarefa é o CVS (Concurrent Versions Systems), principalmente por se tratar de um sistema livre e possuir grande estabilidade e manutenção dada por sua comunidade.

↪ Exemplos de ferramentas:
– CVS (http://www.gnu.org/software/cvs/)
– Subversion (http://subversion.tigris.org/)
– Aegis (http://aegis.sourceforge.net/)
↪ Papéis têm acesso de escrita ao sistema de gerência de configuração utilizado no projeto:
– Desenvolvedores ativos, testadores, documentadores, tradutores, membros de núcleo, líderes de módulos, líderes de projeto, conselheiros.
↪ Restrições:
– Apenas os participantes autorizados pela liderança, ou por papéis que exercem liderança e possuem privilégios na comunidade, podem acessar o sistema de gerência de configuração utilizado no projeto.

Sistemas de visualização de arquivos em repositórios

Estes sistemas são utilizados para visualização de arquivos mantidos em repositórios de dados (como CVS e Subversion) a partir de um navegador Web, possibilitando a navegação nos diretórios presentes no repositório. Podem ser apresentadas versões específicas de arquivos, logs de alterações e diferenças entre estas versões. Este sistema possibilita o acompanhamento online de alterações efetuadas nos artefatos do projeto que estão armazenados em um repositório sob a administração de um sistema de gerência de configuração.

↪ Exemplos de ferramentas:
– ViewCVS (http://viewcvs.sourceforge.net/)
– CVSWeb (http://www.freebsd.org/projects/cvsweb.html)
– Bonsai (http://www.mozilla.org/bonsai.html)
↪ Papéis têm acesso ao sistema de visualização de arquivos em repositórios:
– Todos.
↪ Restrições:
– Não há restrições de acesso a este sistema.

Sistemas de rastreamento e acompanhamento de mudanças (tracking systems)

Os sistemas de rastreamento e acompanhamento de mudanças são contextualizados principalmente em termos de gerência de modificações pedidas e efetuadas no software, como também no acompanhamento de defeitos

encontrados e corrigidos. Neste sentido, estes sistemas são utilizados para manutenção e evolução do software produzido pela comunidade, principalmente pelo fato de permitir o acompanhar toda a evolução de um pedido de alteração no software. Estes sistemas também podem ser utilizados em processos de revisões, na medida em que permitem que anexos de arquivos sejam colocados para análise, facilitando o acesso a artefatos que estão passando por tal processo.

↪ Exemplos de ferramentas:
– Bugzilla (http://www.bugzilla.org/)
– GNATS (http://www.gnu.org/software/gnats/)
– Mantis (http://www.mantisbt.org/)
↪ Papéis que usam tracking systems para cadastrar bugs:
– Qualquer papel cujo indivíduo que o exerça esteja cadastrado neste sistema, desta forma ele assume o papel de relator de bugs.
↪ Papéis que usam tracking systems para atualização do estado de bugs:
– Desenvolvedores ativos, tradutores, testadores, membros de núcleo, líderes de módulos, conselheiros, líderes de projeto.
↪ Restrições:
– Os participantes que utilizam este sistema para consultas não precisam de nenhum tipo de autorização de uso, o uso para cadastrar bugs só é dado a participantes cadastrados no sistema.

Ferramentas de suporte ao lançamento de versões

As versões lançadas por comunidades de SL devem estar acessíveis para download através da Internet. Muitas destas versões devem, no entanto, estar em formatos que viabilizem a execução de seu download e deployment no ambiente do cliente. Para tal, existem ferramentas que são utilizadas para dispor os projetos construídos pelas comunidades em formatos úteis de distribuição. Estas ferramentas geram pacotes de software para diferentes plataformas operacionais, como Linux, Mac OS, Windows, BSD, entre outros. Além disso, existe a possibilidade de trabalhar com arquivos comprimidos para realizar a distribuição do projeto. Entre os tipos de formatos que estas ferramentas suportam estão: .rpm, .tar.gz, .tar.bz2, .zip, .pkg, .deb, etc.

↪ Exemplos de ferramentas:
– Gzip (http://www.gzip.org/)
– Gnu Tar (http://directory.fsf.org/tar.html)
– 7-Zip (http://www.7-zip.org/)
– RPM (http://www.rpm.org/)
↪ Papéis têm acesso ao uso de ferramentas para empacotamento e lançamento de versões:

– Líderes de módulos, membros de núcleo, líderes de projeto.
↪ Restrições:
– O uso destas ferramentas é restrito aos papéis que têm acesso ao servidor de arquivos do projeto e podem construir arquivos para disponibilizar versões do projeto.

3.2.3 Qualidade

Ferramentas de testes automatizados

Testes automatizados são realizados principalmente através do uso de frameworks para testes de unidade (tais como, JUnit, PHPUnit, CUnit), como também através de scripts automatizados de testes que podem vir juntos à distribuição do software. A utilização de testes automatizados garante que modificações efetuadas no software devam estar em concordância com tais testes. Por exemplo, os testes de unidade em um determinado componente de software que tenha sofrido modificações devem ser executados sem erros após as modificações realizadas. Obviamente, em alguns casos o código dos testes devem ser modificados também, porém o ponto principal é que estes testes estejam sempre funcionando em qualquer circunstância. Quando é exercido um controle de qualidade rígido (o que ocorre em projetos da Apache Software Foudantion), o não funcionamento dos testes escritos implica na reprovação da liberação de algum componente de software para compor o projeto.

↪ Exemplos de ferramentas:
– JUnit (http://junit.sourceforge.net/)
– PHPUnit (http://phpunit.sourceforge.net/)
– CUnit (http://cunit.sourceforge.net/)
– Tinderbox (http://www.mozilla.org/tinderbox.html)
↪ Papéis usam frameworks ou scripts para testes:
– Testadores.
↪ Restrições:
– Não há restrições para o uso de frameworks ou scripts de testes, qualquer pessoa que baixe o projeto pode escrever testes para ele. A restrição é dada apenas para a escrita dos testes construídos para o projeto no repositório de dados, que só é permitida a papéis com permissão para isto, os quais podem ser alcançados por meritocracia.

Ferramentas para construções do software (builds tools)

Construções de software são realizadas principalmente com o objetivo de integração dos vários módulos que compõem a aplicação. Neste sentido, são

utilizadas ferramentas que possibilitem a construção automática do software. Isto ocorre tanto do lado do cliente, para viabilizar questões de configurações e dependências de pacotes e bibliotecas externas, quanto do lado do servidor, para garantir que a versão esteja compilando corretamente após a efetivação de qualquer tipo de alteração no repositório de dados. Para um melhor controle de qualidade, as comunidades de SL utilizam as "construções noturnas". Estas construções são realizadas automaticamente e garantem diariamente que o software está compilando. Na ocorrência de problemas de compilação medidas devem ser tomadas no sentido de correção dos mesmos.

↪ Exemplos de ferramentas:
– Apache Ant (http://ant.apache.org/)
– GNU Make (http://www.gnu.org/software/make/)
↪ Papéis usam ferramentas para realização de construções de software:
– Todos (lado do cliente).
– Líderes de módulo, líderes de projeto, membros de núcleo (lado do servidor).
↪ Restrições:
– No lado do cliente qualquer um pode utilizar uma ferramenta deste tipo para construção do projeto. Já no lado do servidor as construções são feitas por papéis com acesso ao servidor de arquivos do projeto. Neste caso, podem ser configuradas para execuções previamente agendadas.

3.2.4 Colaboração e Gerência de projetos

Ambiente colaborativo de desenvolvimento

Como observamos, o desenvolvimento de projetos de SL pode fazer uso de diversos tipos de ferramentas para diferentes propósitos. Montar uma estrutura em um nível que ocorra o uso de um conjunto maior de ferramentas requer investimentos, principalmente pelo fato da existência de uma colaboração contínua entre os atores humanos e os recursos computacionais existentes em um projeto de SL. No sentido de fomentar a integração destes atores e facilitar a atividade de gerenciamento dos projetos de SL, existem ferramentas para o gerenciamento de ambientes colaborativos. Entre estas ferramentas destacam-se o SourceCast (utilizado para desenvolvimento de projetos do portal tigris.org e do projeto NetBeans) e o SourceForge (utilizado para suportar as comunidades do portal sourceforge.net). Ambos provêem um ambiente integrado de ferramentas para comunicação, apoio ao processo qualidade e gerência e acompanhamento dos projetos. Desta forma, a comunidade faz uso de todas as ferramentas disponíveis em um projeto de maneira integrada e padronizada. Igualmente, estes ambientes oferecem subsistemas para gerência e

acompanhamento do projeto como: número de visitas à página do projeto, número de downloads da ferramenta, quantidade de mensagens postadas nas listas de discussões, quantidades de commits realizados no repositório de dados, quantidade de entradas nos sistemas de rastreamento, entre outros.

↪ Exemplos de ferramentas:
– SourceCast (https://www.sourcecast.com/)
– SourceForge (https://sourceforge.net/)
↪ Papéis que usam ferramentas de colaboração para gerenciamento e acompanhamento de projetos:
– Lideres de projetos.
↪ Restrições:
– O uso destas ferramentas requer o cadastramento de um projeto de SL, incluindo: o nome do projeto, o líder, sua descrição, sua categoria (domínio de aplicação) e sua licença.

3.3 Visão Hierárquica de Papéis

Nas comunidades de SL, o crescimento dos participantes é frequentemente dado por um processo de meritocracia. A meritocracia tem por base o fundamento que a produção de trabalhos relevantes por um indivíduo implica em ganho de respeito, status e influência dentro de uma comunidade. Isto quer dizer que: quanto maior o envolvimento de uma pessoa dentro de um projeto maior será seu respeito e sua autoridade perante a comunidade que desenvolve este projeto.

No sentido de oferecer uma visão de maior organização de uma comunidade de SL, realizamos a nivelação dos papéis encontrados. Como parâmetros de referência para definir o nível em que cada papel se encontra na hierarquia de uma comunidade de SL utilizamos: o poder de tomada de decisão e o acesso aos recursos do projeto, ambos podem ser obtidos por meritocracia.

O poder de tomada de decisão indica quais participantes têm a competência de definir os rumos do projeto, dar ou retirar privilégios e/ou responsabilidades a outros membros da comunidade, enfim, têm a competência de gerenciar uma comunidade de SL. O acesso aos recursos indica quais participantes detêm maiores privilégios de acesso aos recursos disponibilizados para o desenvolvimento do projeto, estes recursos são: sistemas de gerência de configuração, sistemas de rastreamento e acompanhamento de mudanças, entre outras ferramentas que venham a ser utilizadas no desenvolvimento de um projeto de SL.

A seguir mostramos quais são os privilégios e poderes de decisão, em relação ao projeto, dos níveis hierárquicos propostos:

Nível 7

↪ Privilégios: acesso de leitura ao repositório de dados do projeto.
↪ Poder de decisão: nenhum.

Nível 6

↪ Privilégios: acesso de leitura ao repositório de dados do projeto, postagem de mensagens nas listas.
↪ Poder de decisão: nenhum.

Nível 5

↪ Privilégios: acesso de leitura ao repositório de dados do projeto, postagem de mensagens nas listas, uso da ferramenta de rastreamento e acompanhamento de mudanças para envio de bugs.
↪ Poder de decisão: nenhum.

Nível 4

↪ Privilégios: acesso de leitura ao repositório de dados do projeto, postagem de mensagens nas listas, uso da ferramenta de rastreamento e acompanhamento de mudanças para envio de correções de bugs e modificação do estado da correção.
↪ Poder de decisão: nenhum.

Nível 3

↪ Privilégios: acesso de leitura e escrita ao repositório de dados do projeto, postagem de mensagens nas listas, acesso de escrita aos arquivos da página do projeto, uso da ferramenta de rastreamento e acompanhamento de mudanças para envio de correções de bugs e modificação do estado da correção.
↪ Poder de decisão: podem participar de votações (dependendo da forma de organização da comunidade) para definir prioridades no projeto.

Nível 2

↪ Privilégios: acesso de leitura e escrita ao repositório de dados do projeto, acesso de administração nas listas de discussão, acesso de escrita aos arquivos da página do projeto, uso da ferramenta de rastreamento e acompanhamento de mudanças para envio de correções de bugs e modificação do estado da correção, acesso às ferramentas de geração de pacotes para lançamento de versões, podem realizar construções do projeto no lado do servidor através do uso de ferramentas de construção.

↪ Poder de decisão: definem prioridades para o projeto, decidem que contribuições serão integradas ao código do projeto, dão acesso de escrita aos recursos do projeto (repositório, página, etc.), tomam decisões de aspectos técnicos para evolução do projeto, participam de decisões junto à liderança do projeto.

Nível 1

↪ Privilégios: Total.
↪ Poder de decisão: Total.

Na Figura, logo abaixo, mostramos que participantes detêm maior privilégio de acesso aos recursos de uma comunidade, quanto menor o nível em que o papel se encontra, maior o privilégio de acesso de um participante.

É interessante notar que, qualquer papel em nível superior da hierarquia pode executar funções referentes a papéis de nível inferior, o que não ocorre de baixo para cima. Assim, um membro de núcleo pode relatar bugs, bem como pode corrigir bugs. Neste caso, ele não perde seus privilégios, apenas executa uma função que é de responsabilidade de um outro papel. Já papéis de nível inferior não têm privilégios, além daqueles definidos para ele, dentro do projeto.

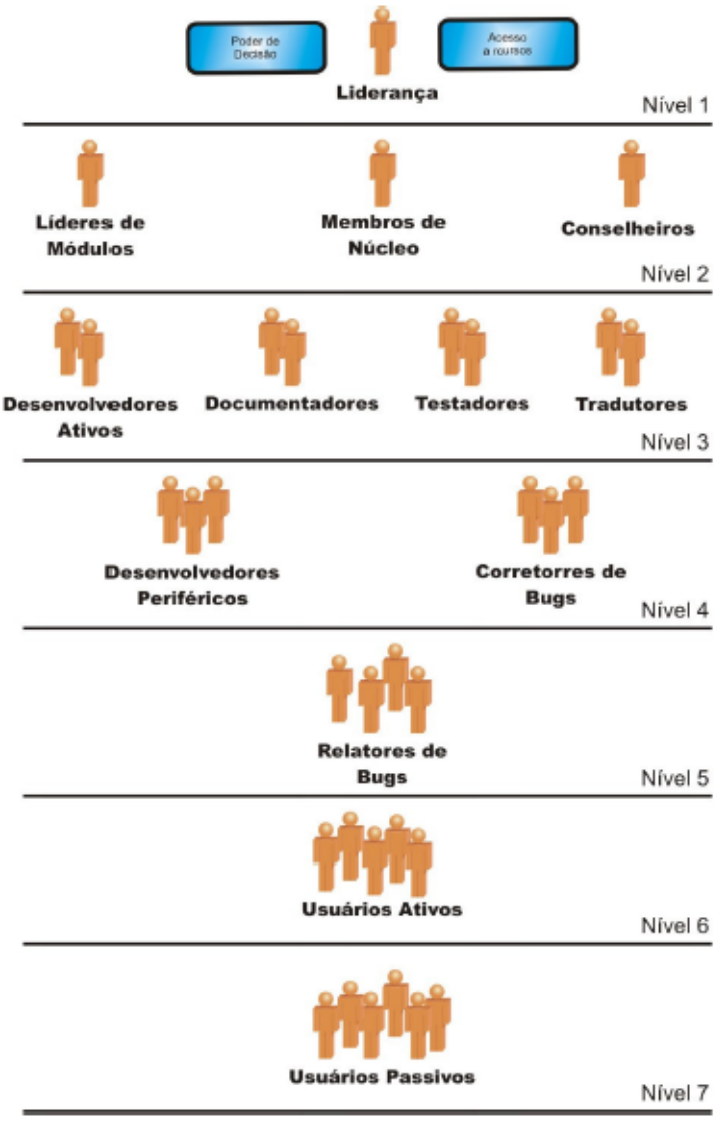

Figura 3.1: Organização hierárquica de uma comunidade de Software Livre

Capítulo 4

Licenças de Software

4.1 Introdução e Conceito

Licenças de software podem ser encaradas como documentos, de valor jurídico, que definem como o software pode ser utilizado. Quando se compra um software, o que na verdade se compra é o direito de utilização do mesmo, o comprador não se torna seu proprietário. Os verdadeiros proprietários são os autores, fornecedores ou detentores de patentes que possuem os direitos sobre o produto e determinam o que pode ou não ser feito com ele.

As licenças são baseadas em conceitos de propriedade intelectual, principalmente através de leis de copyright (direito de cópia). O conceito de propriedade intelectual está relacionado ao que o autor de um novo bem determina, dentro de limites socialmente aceitos e legalmente protegidos, bem como as condições sob as quais o bem pode ser usado por terceiros. Este conceito é utilizado para designar proteções a produtos diferentes como músicas, livros, filmes, processos industriais, técnicas científicas reguladas através de patentes assim como software. O conceito de copyright foi criado para proteger os autores e inventores e estabelecer um meio de recompensá-los e incentivar novos desenvolvimentos, sendo originalmente destinado a livros, de forma que o seu conteúdo fosse proibido de copiar.

Software livre também é licenciado, conforme fora comentado, a licença deve permitir os direitos de distribuição, execução, aprendizado e modificação do software,elas garante direitos aos usuários do software. Já as licenças baseadas em copyright "defendem" o autor do software, restringindo o uso do software conforme termos especificados na licença veiculada junto ao software. Enfim, as licenças baseadas em copyrigh visam proteger os criadores do software, enquanto as licenças de SL visam garantir os direitos dos usuários do software, gerando um conceito oposto ao conceito de copyrigh, o copyleft. No sentido esclarecer os

diversos tipos de software, a Free Software Foundation (FSF)[1], que é uma fundação de fomento ao uso e desenvolvimento de SL, apresenta um glossário de categorias de software em função de restrições e direitos que suas licenças apresentam.

4.1.1 Software Livre (Free Software)

Software que vem com permissão para qualquer um copiar, usar e distribuir, com ou sem modificações, gratuitamente ou por um preço. Em particular, isso significa que o código fonte deve estar disponível.

4.1.2 Código Aberto (Open Source)

Termo utilizado por algumas pessoas para dizer mais ou menos a mesma coisa que software livre. Código aberto, no entanto, pode ser distribuído segundo licenças não aprovadas pela FSF.

4.1.3 Domínio Público

Software no domínio público é software não protegido por leis de copyright. Este tipo de software pode ser alterado e utilizado por qualquer um, no entanto, algumas cópias ou versões modificadas podem não ser livres.

4.1.4 Software protegido com copyleft

O software protegido com copyleft é um software livre cujos termos de distribuição não permite que pessoas que redistribuem o software incluam restrições adicionais quando modificam o software. Isto significa que toda cópia do software, mesmo que tenha sido modificada, precisa ser software livre. O copyleft diz que qualquer um que distribui o software, com ou sem modificações, tem que passar adiante a liberdade de copiar e modificar novamente o software, garantindo que todos os usuários têm liberdade.

4.1.5 Freeware

Termo usado para software que permite sua redistribuição, porém não permite que modificações sejam feitas, o código fonte do software não está disponível. Os termos freeware e SL são diferentes, o seu uso como sinônimo é incorreto.

[1] http://www.fsf.org

4.1.6 Shareware

Software que vem com permissão para redistribuir cópias, no entanto limita o uso do software a um tempo pré-determinado. Para maior parte dos sharewares, o código fonte não está disponível, inviabilizando a modificação do programa, também não vêm com permissão para fazer uma cópia e instalá-la sem pagar uma licença.

4.1.7 Software proprietário

Software proprietário é aquele que não é livre ou semi-livre (Possuem algumas restrições, podendo ser utilizados para fins não lucrativos). Seu uso, redistribuição ou modificação é proibido, ou requer pedido de permissão, ou é restrito de tal forma que não se pode efetivamente fazer seu uso de forma livre. A maior parte dos softwares comerciais está caracterizada como software proprietário, embora não sejam a mesma coisa. Existem softwares comercias que também são livres, o que não ocorre com software proprietário.

4.2 Principais licenças de software livre

A FSF qualifica uma licença de acordo ela ser uma licença de SL (ser copyleft), ou seja, ser compatível com a GNU GPL e não cause problemas práticos particulares. Por sua vez a OSI permite licenças que não estejam enquadradas nas exigências pedidas pela FSF, como exemplo disto podemos citar a licença Plan 9 (Lucent Public License), que não concede o direito de realizar mudanças privativas, o que é um problema particular para esta licença ser uma licença compatível com GNU GPL, segundo a FSF. Principais licenças encontradas em projetos de SL são: GNU GPL, GNU LGPL, BSD License, Apache Software License, Mozilla Public License.

4.2.1 GNU GPL (General Public License)

Licença usada na maioria dos projetos de SL. É uma licença com copyleft, ou seja, permite a redistribuição se forem mantidas as garantias de liberdade para quem recebe cópias do software, bem como obriga que qualquer modificação realizada no software seja livre.

4.2.2 GNU LGPL (Less General Public License)

Licença de SL compatível com a licença GPL, porém é uma licença com copyleft parcial, pois permite que software sob esta licença seja utilizado em outros

produtos sem que estes sejam necessariamente livres. Geralmente são utilizadas para bibliotecas, que podem ser usadas em software proprietário. Apenas modificações realizadas nos pacotes sob a licença LGPL devem permanecer livres, o produto ao qual o pacote está ligado é isento de ser software livre.

4.2.3 BSD License (Berkeley Software Distribution License)

Licença permissiva, não é copyleft, ou seja, permite que versões modificadas sejam distribuídas de forma não-livre. A licença original do BSD faz uso de uma cláusula de propaganda, obrigando cópias redistribuídas manter um aviso visível reconhecendo o uso de software desenvolvido pela Universidade da Califórnia em Berkeley, ou por seus colaboradores. Esta cláusula trouxe um problema prático e tornou esta licença incompatível com a GNU GPL, este problema foi corrigido com a licença modificada do BSD.

4.2.4 ASF (Apache Software License)

Licença permissiva que não exige copyleft, esta licença possui algumas exigências que a tornam incompatível com a GNU GPL. Atualmente a Apache Software License está na versão 2.0 e permanece incompatível com a GNU GPL por possuir certas terminologias de patentes que não são requeridas pala GNU GPL.

4.2.5 MPL (Mozilla Public License)

Licença de software livre que não exige copyleft forte, possui algumas restrições complexas que a tornam incompatível com a GNU GPL. Um módulo coberto pela GPL e um módulo coberto pela MPL não podem ser misturados. A MPL 1.1 possui um dispositivo que permite que um programa (ou partes dele) ofereça outras licenças como alternativa, entre estas alternativas estão a GNU GPL e licenças compatíveis com ela, isto faz com que a MPL 1.1 seja compatível com a licença GNU GPL. Geralmente, o autor define que licença que será utilizada para o projeto. O autor do projeto possui o poder de modificar a licença original a qualquer momento. O licenciamento também pode ser realizado por uma licença definida pelo autor do projeto, conforme seja a necessidade de criação de uma licença para tal. Uma comunidade pode iniciar o desenvolvimento de um software sob a licença GPL, que pode ser modificada, no futuro, para uma outra licença. Igualmente, pode haver partes do software sob diferentes licenças, o que é usado como estratégia de alguns produtos para possibilitar a agregação do produto de SL a produtos não-livres. Este caso ocorre no licenciamento projeto OpenOffice.org, que faz uso de duas licenças, a LGPL e SISSL (Sun Industry

Standards Source License). Além dessas duas licenças, é usada uma outra licença para documentação incluída no produto e na página do projeto, a PDL (Public Documentation License).

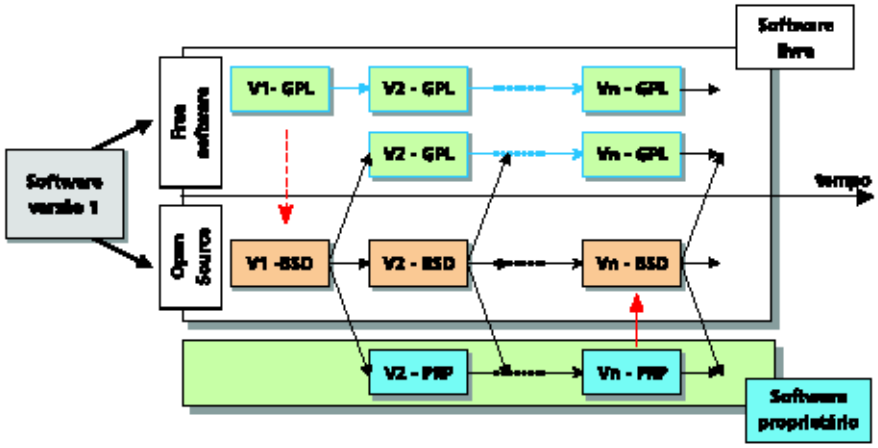

Figura 4.1: Possíveis evoluções do licenciamento de um software livre e de um software proprietário

Conclusão

"A melhor coisa depois de ter boas idéias é reconhecer boas idéias dos seus usuários. As vezes a última é melhor." Eric S. Raymond

O Movimento do Software Livre é a expressão de um conceito de liberdade aplicado ao mundo do software, este mesmo conceito evoluiu e hoje em dia já está sendo aplicado em outras áreas, como na produção de bens culturais, através do projeto Creative Commons[2] onde livros, vídeos, filmes, artigos, músicas podem ter a sua licença específica e livre, deixando de utilizar assim o copyright que somente faz restringir e proibir. Clamo que esta liberdade se espalhe, pois ela trás uma série de benefícios alterando nosso pensamento e comportamento. Outro movimento de conscientização e luta que nos trará maior liberdade e descentralização tecnológica diz respeito a liberação do Espectro Radioelétrico com vistas a tornar o nosso computador um meio não somente de comunicação mas também de telecomunicação, softwares para isso já existem, o que está nos impedindo são somente os modelos de negócios existentes e as reservas de mercado fomentadas pelo governo. Bom, é isso ...

[2]http://creativecommons.org.br/

Referências Bibliográficas

[1] Eric S. Raymond. **A Catedral e o Bazar**. Disponivel em: <http://www.catb.org/ esr/writings/cathedral-bazaar/>. Acesso em: 29 agosto 2006.

[2] Jose Carlos de Araújo Almeida Filho. **Introdução ao Estudo das Leis 9609/98 (Lei de Software) e 9610/98 (Lei de Direitos Autorais) e sua Importância no Novo Instituto Jurídico**. Disponivel em: <http://www.politicahoje.com/politica/aula02%5B2%5D.pdf>. Acesso em: 06 novembro 2006.

[3] **PSL Brasil**. Disponível em: <http://www.softwarelivre.org>. Acesso em: 06 novembro 2006.

[4] **Softex**. Disponível em: <http://www.softex.br>. Acesso em: 06 novembro 2006.

[5] Softex/Unicamp/MCT. **O Impacto do Software Livre e de Código Aberto na Indístria do Software do Brasil**. Disponivel em: <http://golden.softex.br/portal/softexweb/uploadDocuments/pesquisa-swl.pdf>. Acesso em: 06 novembro
2006.

[6] Via Digital. **Gestão de Projetos de Software Livre: Uma Abordagem de Práticas**. Disponivel em: <http://www.viadigital.org.br/docs/Praticas.pdf>. Acesso em: 06 novembro 2006.

[7] **Wikipedia**. Disponível em: <http://pt.wikipedia.org>. Acesso em: 06 novembro 2006.

Apêndice A

Licença CC-GNU GPL

Licença CC-GNU GPL

Figura A.1: Creative Commons

Licença Pública Geral do GNU (GPL) [General Public License]

Versão 2 [1], Junho de 1991 Direitos Autorais Reservados (c) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite [conjunto] 330, Boston, MA [Massachusetts] 02111-1307 USA [Estados Unidos da América]

É permitido a qualquer pessoa copiar e distribuir cópias sem alterações deste documento de licença, sendo vedada, entretanto, qualquer modificação.

Introdução

As licenças da maioria dos *softwares* são elaboradas para suprimir sua liberdade de compartilhá-los e modificá-los. A Licença Pública Geral do GNU, ao contrário, visa garantir sua liberdade de compartilhar e modificar *softwares* livres para assegurar que o *software* seja livre para todos os seus usuários. Esta Licença Pública Geral é aplicável à maioria dos *softwares* da Free *Software* Foundation [Fundação do *Software* livre] e a qualquer outro programa cujos autores se comprometerem a usá-la. (Em vez dela, alguns outros *softwares* da Free *Software* Foundation são cobertos pela Licença Pública Geral de Biblioteca do GNU). Você também poderá aplicá-la aos seus programas.

Quando falamos de *Software* Livre, estamos nos referindo à liberdade, não ao preço. Nossas Licenças Públicas Gerais visam garantir que você tenha a liberdade de distribuir cópias de *Software* Livre (e cobrar por isso se desejar), que receba código-fonte ou possa obtê-lo se desejar, que possa modificá-lo ou usar partes dele em novos programas livres; finalmente, que você tenha ciência de que pode fazer tudo isso.

Para proteger seus direitos, necessitamos fazer restrições que proíbem que alguém negue esses direitos a você ou que solicite que você renuncie a eles. Essas restrições se traduzem em determinadas responsabilidades que você deverá assumir, se for distribuir cópias do *software* ou modificá-lo.

[1] Disponível em http://creativecommons.org/licenses/GPL/2.0/legalcode.pt

Por exemplo, se você distribuir cópias de algum desses programas, tanto gratuitamente como mediante uma taxa, você terá de conceder aos receptores todos os direitos que você possui. Você terá de garantir que, também eles, recebam ou possam obter o código-fonte. E você terá a obrigação de exibir a eles esses termos, para que eles conheçam seus direitos.

Protegemos seus direitos através de dois passos: (1) estabelecendo direitos autorais sobre o *software* e (2) concedendo a você esta licença, que dá permissão legal para copiar, distribuir e/ou modificar o software.

Além disso, para a proteção de cada autor e a nossa, queremos ter certeza de que todos entendam que não há nenhuma garantia para este *Software* Livre. Se o *software* for modificado por alguém e passado adiante, queremos que seus receptores saibam que o que receberam não é o original, de forma que quaisquer problemas introduzidos por terceiros não afetem as reputações dos autores originais.

Finalmente, qualquer programa livre é constantemente ameaçado por patentes de software. Queremos evitar o risco de que redistribuidores de um programa livre obtenham individualmente licenças sob uma patente, tornando o programa, com efeito, proprietário. Para impedir isso, deixamos claro que qualquer patente deve ser licenciada para o uso livre por parte de qualquer pessoa ou, então, simplesmente não deve ser licenciada.

Os exatos termos e condições para cópia, distribuição e modificação seguem abaixo.

TERMOS E CONDIÇÕES PARA CÓPIA, DISTRIBUIÇÃO E MODIFICAÇÃO.

1. Esta Licença se aplica a qualquer programa ou outra obra que contenha um aviso inserido pelo respectivo titular dos direitos autorais, informando que a referida obra pode ser distribuída em conformidade com os termos desta Licença Pública Geral. O termo "Programa", utilizado abaixo, refere-se a qualquer programa ou obra, e o termo "obras baseadas no Programa" significa tanto o Programa, como qualquer obra derivada nos termos da legislação de direitos autorais: isto é, uma obra contendo o Programa ou uma parte dele, tanto de forma idêntica como com modificações, e/ou traduzida para outra linguagem. (Doravante, o termo "modificação" inclui também, sem reservas, a tradução). Cada licenciado, doravante, será denominado "você".

   Outras atividades que não a cópia, distribuição e modificação, não são cobertas por esta Licença; elas estão fora de seu escopo. O ato de executar o Programa não tem restrições e o resultado gerado a partir do Programa encontra-se coberto somente se seu conteúdo constituir uma obra baseada no Programa (independente de ter sido produzida pela execução do Programa). Na verdade, isto dependerá daquilo que o Programa faz.

2. Você poderá fazer cópias idênticas do código-fonte do Programa ao recebê-lo e distribui-las, em qualquer mídia ou meio, desde que publique, de forma ostensiva e adequada, em cada cópia, um aviso de direitos autorais (ou copyright) apropriado e uma notificação sobre a exoneração de garantia; mantenha intactas as informações, avisos ou notificações referentes a esta Licença e à ausência de qualquer garantia; e forneça a quaisquer outros receptores do Programa uma cópia desta Licença junto com o Programa.

   Você poderá cobrar um valor pelo ato físico de transferir uma cópia, e você pode oferecer, se quiser, a proteção de uma garantia em troca de um valor.

3. Você poderá modificar sua cópia ou cópias do Programa ou qualquer parte dele, formando, dessa forma, uma obra baseada no Programa, bem como copiar e distribuir essas modificações ou obra, de acordo com os termos da Cláusula 1 acima, desde que você também atenda a todas as seguintes condições:

   a. Você deve fazer com que os arquivos modificados contenham avisos, em destaque, informando que você modificou os arquivos, bem como a data de qualquer modificação.

   b. Você deve fazer com que qualquer obra que você distribuir ou publicar, que no todo ou em parte contenha o Programa ou seja dele derivada, ou derivada de qualquer parte dele, seja licenciada como um todo sem qualquer custo para todos terceiros nos termos desta licença.

   c. Se o programa modificado normalmente lê comandos interativamente quando executado, você deverá fazer com que ele, ao começar a ser executado para esse uso interativo em sua forma mais simples, imprima ou exiba um aviso incluindo o aviso de direitos autorais (ou copyright) apropriado, além de uma notificação de que não há garantia (ou, então, informando que você oferece garantia) e informando que os usuários poderão redistribuir o programa de acordo com essas condições, esclarecendo ao usuário como visualizar uma cópia desta Licença. (Exceção: se o Programa em si for interativo mas não imprimir normalmente avisos como esses, não é obrigatório que a sua obra baseada no Programa imprima um aviso).

      Essas exigências se aplicam à obra modificada como um todo. Se partes identificáveis dessa obra não forem derivadas do Programa e puderem ser consideradas razoavelmente como obras independentes e separadas por si próprias, nesse caso, esta Licença e seus termos não se aplicarão a essas partes quando você distribui-las como obras separadas. Todavia, quando você distribui-las como parte de um todo que constitui uma

obra baseada no Programa, a distribuição deste todo terá de ser realizada em conformidade com esta Licença, cujas permissões para outros licenciados se estenderão à obra por completo e, conseqüentemente, a toda e qualquer parte, independentemente de quem a escreveu.

Portanto, esta cláusula não tem a intenção de afirmar direitos ou contestar os seus direitos sobre uma obra escrita inteiramente por você; a intenção é, antes, de exercer o direito de controlar a distribuição de obras derivadas ou obras coletivas baseadas no Programa.

Além do mais, a simples agregação de outra obra que não seja baseada no Programa a ele (ou a uma obra baseada no Programa) em um volume de mídia ou meio de armazenamento ou distribuição, não inclui esta outra obra no âmbito desta Licença.

4. Você poderá copiar e distribuir o Programa (ou uma obra baseada nele, de acordo com a Cláusula 2) em código-objeto ou formato executável de acordo com os termos das Cláusulas 1 e 2 acima, desde que você também tome uma das providências seguintes:

   a. Incluir o código-fonte correspondente completo, passível de leitura pela máquina, o qual terá de ser distribuído de acordo com as Cláusulas 1 e 2 acima, em um meio ou mídia habitualmente usado para intercâmbio de software; ou,

   b. Incluir uma oferta por escrito, válida por pelo menos três anos, para fornecer a qualquer terceiro, por um custo que não seja superior ao seu custo de fisicamente realizar a distribuição da fonte, uma cópia completa passível de leitura pela máquina, do código-fonte correspondente, a ser distribuído de acordo com as Cláusulas 1 e 2 acima, em um meio ou mídia habitualmente usado para intercâmbio de software; ou,

   c. Incluir as informações recebidas por você, quanto à oferta para distribuir o código-fonte correspondente. (Esta alternativa é permitida somente para distribuição não-comercial e apenas se você tiver recebido o programa em código-objeto ou formato executável com essa oferta, de acordo com a letra b, acima).

      O código-fonte de uma obra significa o formato preferencial da obra para que sejam feitas modificações na mesma. Para uma obra executável, o código-fonte completo significa o código-fonte inteiro de todos os módulos que ela contiver, mais quaisquer arquivos de definição de interface associados, além dos *scripts* usados para controlar a compilação e instalação do executável. Entretanto, como uma exceção especial,

o código-fonte distribuído não precisa incluir nada que não seja normalmente distribuído (tanto no formato fonte como no binário) com os componentes principais (compilador, kernel e assim por diante) do sistema operacional no qual o executável é executado, a menos que este componente em si acompanhe o executável.

Se a distribuição do executável ou código-objeto for feita mediante a permissão de acesso para copiar, a partir de um local designado, então, a permissão de acesso equivalente para copiar o código-fonte a partir do mesmo local será considerada como distribuição do código-fonte, mesmo que os terceiros não sejam levados a copiar a fonte junto com o código-objeto.

5. Você não poderá copiar, modificar, sublicenciar ou distribuir o Programa, exceto conforme expressamente estabelecido nesta Licença. Qualquer tentativa de, de outro modo, copiar, modificar, sublicenciar ou distribuir o Programa será inválida, e automaticamente rescindirá seus direitos sob esta Licença. Entretanto, terceiros que tiverem recebido cópias ou direitos de você de acordo esta Licença não terão suas licenças rescindidas, enquanto estes terceiros mantiverem o seu pleno cumprimento.

6. Você não é obrigado a aceitar esta Licença, uma vez que você não a assinou. Porém, nada mais concede a você permissão para modificar ou distribuir o Programa ou respectivas obras derivativas. Tais atos são proibidos por lei se você não aceitar esta Licença. Conseqüentemente, ao modificar ou distribuir o Programa (ou qualquer obra baseada no Programa), você estará manifestando sua aceitação desta Licença para fazê-lo, bem como de todos os seus termos e condições para copiar, distribuir ou modificar o Programa ou obras nele baseadas.

7. Cada vez que você redistribuir o Programa (ou obra baseada no Programa), o receptor receberá, automaticamente, uma licença do licenciante original, para copiar, distribuir ou modificar o Programa, sujeito a estes termos e condições. Você não poderá impor quaisquer restrições adicionais ao exercício, pelos receptores, dos direitos concedidos por este instrumento. Você não tem responsabilidade de promover o cumprimento por parte de terceiros desta licença.

8. Se, como resultado de uma sentença judicial ou alegação de violação de patente, ou por qualquer outro motivo (não restrito às questões de patentes), forem impostas a você condições (tanto através de mandado judicial, contrato ou qualquer outra forma) que contradigam as condições desta Licença, você não estará desobrigado quanto às condições desta Licença. Se você não

puder atuar como distribuidor de modo a satisfazer simultaneamente suas obrigações sob esta licença e quaisquer outras obrigações pertinentes, então, como conseqüência, você não poderá distribuir o Programa de nenhuma forma. Por exemplo, se uma licença sob uma patente não permite a redistribuição por parte de todos aqueles que tiverem recebido cópias, direta ou indiretamente de você, sem o pagamento de royalties, então, a única forma de cumprir tanto com esta exigência quanto com esta licença será deixar de distribuir, por completo, o Programa.

Se qualquer parte desta Cláusula for considerada inválida ou não executável, sob qualquer circunstância específica, o restante da cláusula deverá continuar a ser aplicado e a cláusula, como um todo, deverá ser aplicada em outras circunstâncias.

Esta cláusula não tem a finalidade de induzir você a infringir quaisquer patentes ou direitos de propriedade, nem de contestar a validade de quaisquer reivindicações deste tipo; a única finalidade desta cláusula é proteger a integridade do sistema de distribuição do *Software* Livre, o qual é implementado mediante práticas de licenças públicas. Muitas pessoas têm feito generosas contribuições à ampla gama de *software* distribuído através desse sistema, confiando na aplicação consistente deste sistema; cabe ao autor/doador decidir se deseja distribuir *software* através de qualquer outro sistema e um licenciado não pode impor esta escolha.

Esta cláusula visa deixar absolutamente claro o que se acredita ser uma conseqüência do restante desta Licença.

9. Se a distribuição e/ou uso do Programa for restrito em determinados países, tanto por patentes ou por interfaces protegidas por direito autoral, o titular original dos direitos autorais que colocar o Programa sob esta Licença poderá acrescentar uma limitação geográfica de distribuição explícita excluindo esses países, de modo que a distribuição seja permitida somente nos países ou entre os países que não foram excluídos dessa forma. Nesse caso, esta Licença passa a incorporar a limitação como se esta tivesse sido escrita no corpo desta Licença.

10. A Free *Software* Foundation poderá de tempos em tempos publicar novas versões e/ou versões revisadas da Licença Pública Geral. Essas novas versões serão semelhantes em espírito à presente versão, mas podem diferenciar-se, porém, em detalhe, para tratar de novos problemas ou preocupações.

    Cada versão recebe um número de versão distinto. Se o Programa especificar um número de versão desta Licença que se aplique a ela e a "qualquer versão posterior", você terá a opção de seguir os termos e condições tanto daquela

versão como de qualquer versão posterior publicada pela Free *Software* Foundation. Se o Programa não especificar um número de versão desta Licença, você poderá escolher qualquer versão já publicada pela Free *Software* Foundation.

11. Se você desejar incorporar partes do Programa em outros programas livres cujas condições de distribuição sejam diferentes, escreva ao autor solicitando a respectiva permissão. Para *software* cujos direitos autorais sejam da Free *Software* Foundation, escreva para ela; algumas vezes, abrimos exceções para isso. Nossa decisão será guiada pelos dois objetivos de preservar a condição livre de todos os derivados de nosso *Software* Livre e de promover o compartilhamento e reutilização de software, de modo geral.

EXCLUSÃO DE GARANTIA

11. COMO O PROGRAMA É LICENCIADO SEM CUSTO, NÃO HÁ NENHUMA GARANTIA PARA O PROGRAMA, NO LIMITE PERMITIDO PELA LEI APLICÁVEL. EXCETO QUANDO DE OUTRA FORMA ESTABELECIDO POR ESCRITO, OS TITULARES DOS DIREITOS AUTORAIS E/OU OUTRAS PARTES, FORNECEM O PROGRAMA "NO ESTADO EM QUE SE ENCONTRA", SEM NENHUMA GARANTIA DE QUALQUER TIPO, TANTO EXPRESSA COMO IMPLÍCITA, INCLUINDO, DENTRE OUTRAS, AS GARANTIAS IMPLÍCITAS DE COMERCIABILIDADE E ADEQUAÇÃO A UMA FINALIDADE ESPECÍFICA. O RISCO INTEGRAL QUANTO À QUALIDADE E DESEMPENHO DO PROGRAMA É ASSUMIDO POR VOCÊ. CASO O PROGRAMA CONTENHA DEFEITOS, VOCÊ ARCARÁ COM OS CUSTOS DE TODOS OS SERVIÇOS, REPAROS OU CORREÇÕES NECESSÁRIAS.

12. EM NENHUMA CIRCUNSTÂNCIA, A MENOS QUE EXIGIDO PELA LEI APLICÁVEL OU ACORDADO POR ESCRITO, QUALQUER TITULAR DE DIREITOS AUTORAIS OU QUALQUER OUTRA PARTE QUE POSSA MODIFICAR E/OU REDISTRIBUIR O PROGRAMA, CONFORME PERMITIDO ACIMA, SERÁ RESPONSÁVEL PARA COM VOCÊ POR DANOS, INCLUINDO ENTRE OUTROS, QUAISQUER DANOS GERAIS, ESPECIAIS, FORTUITOS OU EMERGENTES, ADVINDOS DO USO OU IMPOSSIBILIDADE DE USO DO PROGRAMA (INCLUINDO, ENTRE OUTROS, PERDAS DE DADOS OU DADOS SENDO GERADOS DE FORMA IMPRECISA, PERDAS SOFRIDAS POR VOCÊ OU TERCEIROS OU A IMPOSSIBILIDADE DO PROGRAMA DE OPERAR COM QUAISQUER OUTROS PROGRAMAS), MESMO QUE ESSE TITULAR, OU

OUTRA PARTE, TENHA SIDO ALERTADA SOBRE A POSSIBILIDADE DE OCORRÊNCIA DESSES DANOS.

FINAL DOS TERMOS E CONDIÇÕES

Como Aplicar Estes Termos para Seus Novos Programas. Se você desenvolver um programa novo e quiser que ele seja da maior utilidade possível para o público, o melhor caminho para obter isto é fazer dele um *Software* Livre, o qual qualquer pessoa pode redistribuir e modificar sob os presentes termos. Para fazer isto, anexe as notificações seguintes ao programa. É mais seguro anexá-las ao começo de cada arquivo-fonte, de modo a transmitir do modo mais eficiente a exclusão de garantia; e cada arquivo deve ter ao menos a linha de "direitos autorais reservados" e uma indicação de onde a notificação completa se encontra.

> <uma linha para informar o nome do programa e uma breve idéia do que ele faz.> Direitos Autorais Reservados (c) <nome do autor> Este programa é *Software* Livre; você pode redistribuí-lo e/ou modificá-lo sob os termos da Licença Pública Geral GNU conforme publicada pela Free *Software* Foundation; tanto a versão 2 da Licença, como (a seu critério) qualquer versão posterior. 135 Este programa é distribuído na expectativa de que seja útil, porém, SEM NENHUMA GARANTIA; nem mesmo a garantia implícita de COMERCIABILIDADE OU ADEQUAÇÃO A UMA FINALIDADE ESPECÍFICA. Consulte a Licença Pública Geral do GNU para mais detalhes.
>
> Você deve ter recebido uma cópia da Licença Pública Geral do GNU junto com este programa; se não, escreva para a Free *Software* Foundation, Inc., no endereço 59 Temple Street, Suite 330, Boston, MA 02111-1307 USA. Inclua também informações sobre como contatar você por correio eletrônico e por meio postal.

Se o programa for interativo, faça com que produza uma pequena notificação como esta, quando for iniciado em um modo interativo:

> Versão 69 do Gnomovision, Direitos Autorais Reservados (c) ano nome do autor. O Gnomovision NÃO POSSUI QUALQUER TIPO DE GARANTIA; para detalhes, digite 'show w'. Este é um *Software* Livre e você é bem-vindo para redistribuí-lo sob certas condições; digite 'show c' para detalhes.

Os comandos hipotéticos 'show w' e 'show c' devem mostrar as partes apropriadas da Licença Pública Geral. Naturalmente, os comandos que você utilizar poderão ter outras denominações que não 'show w' e 'show c'; eles poderão até ser cliques do *mouse* ou itens de um menu – o que for adequado ao seu programa.

Você também pode solicitar a seu empregador (se você for um programador) ou sua instituição acadêmica, se for o caso, para assinar uma "renúncia de direitos autorais" sobre o programa, se necessário. Segue um exemplo; altere os nomes:

> A Yoyodyne Ltda., neste ato, renuncia a todos eventuais direitos autorais sobre o programa 'Gnomovision' (que realiza passagens em compiladores), escrito por James Hacker.
>
> <Assinatura de Ty Coon> 1º de abril de 1989, Ty Coon, Presidente

Esta Licença Pública Geral não permite a incorporação do seu programa a programas proprietários. Se seu programa é uma biblioteca de sub-rotinas, você poderá considerar ser mais útil permitir a ligação de aplicações proprietárias à sua biblioteca. Se isso é o que você deseja fazer, utilize a Licença Pública Geral de Biblioteca do GNU, ao invés desta Licença.

Apêndice B

Dicas e Diretrizes para se Iniciar um Projeto de Software Livre

Dicas e Diretrizes para se Iniciar um Projeto de Software Livre

1. Introdução

O objetivo deste documento é mostrar como iniciar um projeto de SL na visão de gestão de projetos. Desta forma, realizamos um estudo no sentido de fornecer dicas de como pode ser iniciado um novo projeto de SL.
Assim, construímos um esquema passo a passo de como futuros gerentes de projetos podem integrar-se num ambiente de desenvolvimento de SL. Este documento está dividido de forma a enfatizar os seguintes aspectos:

- Aspectos úteis para iniciar um projeto de SL;
- Licenciando um projeto de SL;
- Montagem da infra-estrutura para desenvolvimento do projeto;
- Definição dos papéis que fazem parte do projeto;
- Documentação.

2. Aspectos Úteis para Iniciar um Projeto de SL

No momento em que se decide dar início a um projeto de SL, deve-se considerar alguns pontos que podem ajudar o autor (que certamente, mas não necessariamente, será o líder do projeto) de um projeto. Considerando que o início de um projeto é a fase onde existe o maior número de casos de insucesso, é interessante realizar um pequeno "checklist" para se assegurar de quais são os passos relevantes para iniciar um desenvolvimento com maior segurança. Desta forma, descrevemos alguns passos importantes para guiar um gerente no momento em que ele escolhe o modelo de desenvolvimento de SL para seu projeto.

2.1. Procure projetos semelhantes

Na medida em que existem outras comunidades com o desenvolvimento de um determinado projeto em estágio amadurecido maior, pode ser mais fácil (e rápido) contribuir para este projeto ao invés de iniciar um do zero. A participação em uma comunidade de SL já estabelecida retira do propenso participante as responsabilidades relativas à gerência do projeto. Assim, esforço exigido para coordenação é excluído caso esta decisão seja tomada.
Informações sobre projetos de SL podem ser facilmente encontradas em diversos meios fornecidos através da Internet.

- SourceForge (http://sourceforge.net/) Atualmente é o maior portal de projetos de SL do mundo. Milhares de pessoas de todo o mundo cadastram seus projetos neste portal. Nele podem ser feitas consultas à sua base de dados para se verificar a existência de projetos de SL. As buscas podem ser realizadas através do nome do projeto ou pela categoria (banco de dados, comunicação, editores de texto, segurança, etc.).

- Freshmeat.net (http://freshmeat.net/) Portal destinado a informar usuários de SL sobre lançamentos de versões de projetos de SL. Oferece um mecanismo de busca interno que pode ser utilizado para verificar a existência de projetos de SL cadastrados em sua base de dados. Ao contrário do SourceForge, este portal não oferece um ambiente para desenvolvimento de projetos de SL, apenas oferece informações sobre projetos de SL.

- Savannah.gnu.org e Savannah.nongnu.org. O portal savannah.gnu.org é destinado ao desenvolvimento, distribuição e manutenção específica de software GNU. Já savannah.nongnu.org é destinado ao desenvolvimento de SL que não necessariamente é software do projeto GNU. Ambos os portais oferecem ferramentas de busca a projetos cadastrados em suas bases de dados.

- Google (http://www.google.com/) Poderosa ferramenta de busca na Internet. Apesar de não ser uma ferramenta destinada diretamente ao desenvolvimento de SL, podem ser encontrados diversos projetos através de seu uso. Estes são alguns exemplos de ferramentas onde pode-se pesquisar a respeito de projetos de SL. De maneira que, encontrar projetos de SL semelhantes ao projeto que alguém se proponha a implementar pode um grande economizar esforço de implementação, pois, pode-se participar de um projeto em andamento ao invés de iniciar um novo projeto do zero.

3. Licenciando um projeto de SL

As licenças de SL têm como principal função a proteção dos usuários de SL, mantendo o direito de utilização do software distribuído sobre seus termos. A escolha de uma licença é uma das atividades mais importantes no contexto do desenvolvimento de SL, pois é a licença que define as características e restrições de uso, modificação e distribuição do software. Embora exista a possibilidade de mudanças na licença de um SL, é importante considerar que podem ocorrer prejuízos com estas mudanças no futuro. Membros da comunidade podem se sentir traídos ou explorados em seu trabalho quando uma licença muda de GPL para uma licença BSD, por exemplo.

3.1.1. Como licenciar um SL

Antes de licenciar um SL é necessário escolher uma licença de SL. Não está no escopo deste trabalho uma discussão detalhada sobre as licenças de SL e suas características, maiores informações sobre licenças podem ser encontradas em:

- http://www.gnu.org/licenses/
- http://www.opensource.org/licenses/

Assim, iremos nos concentrar em como uma licença de SL pode ser aplicada a um determinado produto de software.

- Passo 1: Crie um arquivo de texto separado para as distribuições do software (código-fonte e binária), colocando uma cópia completa da licença utilizada pelo software neste arquivo. Isso pode ser feito através de um arquivo chamado license.txt, que viria anexado ao programa.
- Passo 2: Para cada arquivo de código-fonte que compõe o software coloque um aviso no topo sobre a licença que está sendo utilizada no programa e onde ela pode ser encontrada.

```
As linhas iniciais podem descrever o programa, seu autor
e para quem os direitos de
cópia estão resertvados
Criado em <data de criação>
Criado por <nome do autor>
Copyright (C) 2005 <que detém direito de cópia>, Todos os
direitos reservados.
As linhas subseqüentes mostram informações sobre a licença
em que o software está
sendo distribuído e como obter informações sobre ela
Este programa é Software Livre; você pode redistribuí-lo e/ou
modifica-lo sob os termos da GNU Licença Pública Geral como
publicada na Free Software Foundation; tanto na versão 2 da licença,
ou (caso seja sua opção) qualquer versão atualizada.
Este programa é distribuído na esperança de que seja útil,
mas SEM QUALQUER GARANTIA; nem mesmo incluir garantia de
LUCRATIVIDADE ou
APLICABILIDADE PARA UM PROPÓSITO ESPECÍFICO. Veja a GNU Licença
Pública Geral para maiores detalhes.
Você deve ter recebido uma cópia da GNU Licença Pública Geral
 anexada a este programa; caso contrário, escreva para a
```

```
Free Software Foundation, Inc., 51 Franklin Street, Fifth floor,
 Boston, MA, 02110-1301, USA.\\
A execução destes dois passos é suficiente para informar que
um software está sob uma licença de SL.
```

4. Montagem da Infra-estrutura para Desenvolvimento do Projeto

A montagem de um ambiente de desenvolvimento de SL requer a utilização de algumas ferramentas que irão ajudar na integração da comunidade no sentido de desenvolver um produto de software. Desta maneira levantamos as seguintes ferramentas, que consideramos essenciais, para prover um ambiente "mínimo" para desenvolver SL.

Página do projeto Todo projeto de SL deve possuir uma página de acompanhamento do projeto, onde são dispostas informações gerais sobre:

- O andamento do projeto;
- Acesso a informações e documentos (geralmente em formato HTML)
- Acesso direto para download do projeto
- Acesso a listas de discussão;
- Informar sobre eventos e notícias a respeito do projeto;
- Disponibilizar documentação sobre meios de como contribuir para o projeto.

Listas de discussão As listas servem como um meio de comunicação entre os membros de um projeto de SL. Através delas os membros de um projeto podem discutir sobre o desenvolvimento, o uso, o lançamento de novas versões e qualquer outro tópico relevante ao projeto.

Sistema de gerência de configuração O ambiente de desenvolvimento de projetos de SL é descentralizado, várias pessoas contribuem paralelamente com a evolução do projeto. De maneira que existe a necessidade de se prover um ambiente que armazene os artefatos construídos pelos membros do projeto e realize a gestão das alterações sobre estes artefatos. Assim, este ambiente é fornecido por um sistema de gerência de configuração.

Sistema de rastreamento e acompanhamento de mudanças Estes sistemas são utilizados, principalmente, para efetuar o controle de alterações e correções e problemas relatados em um software. Através de um sistema deste tipo pode ser realizada a manutenção do projeto de maneira controlada, onde os erros podem ser priorizados e as versões futuras do projeto podem ser destinadas à sua correção.

Ferramentas para empacotamento do projeto A forma de disponibilizar os resultados da construção de um projeto de SL é importantíssima. Facilitar o acesso ao projeto de uma maneira simples é essencial para viabilizar seu uso. Desta maneira, entram em uso as ferramentas que permitem o empacotamento de um projeto (que pode ser composto por diversos tipos de arquivos) em uma unidade de software simples de baixar e configurar localmente. Para sistemas Windows pode ser utilizado o formato .zip, para sistemas *NIX pode ser utilizado o formato .tar, que ainda podem ser comprimidos no formato .tar.gz. Os pacotes binários podem vir com extensões .rpm, .exe, ou .deb, por exemplo.

4.1. Onde encontrar um ambiente com este ferramental incluso?

Na Internet existem portais destinados a fomentar o desenvolvimento de projetos de SL através de um ambiente colaborativo de desenvolvimento que integra todas estas ferramentas (e outras mais). Estas ferramentas provêem um ambiente para gerenciamento e desenvolvimento dos projetos disponibilizando um conjunto de ferramentas propícias para tal atividade e sem nenhum custo para os desenvolvedores. O mais famoso, e utilizado, ambiente nesse estilo é o SourceForge.net. No Brasil também existe um grande portal, com objetivos similares ao SourceForge, o Código Livre.

5. Definição dos Papéis que Fazem parte do Projeto

É importante definir quais são os papéis que serão utilizados no projeto e na medida em que o projeto tem seu andamento, alocar membros para executarem estes papéis. No documento de práticas construído para o FLO-PREF, estão definidos vários tipos de papéis que podem ser instanciados em um projeto de SL. A complexidade do projeto pode ser utilizada para definir uma maior, ou menor, granularidade de papéis em um projeto.

6. Documentação

Disponibilizar documentos sobre o projeto também é importantíssimo para a sobrevivência de um projeto. Há várias formas de documentar um projeto de SL, como também há vários tipos de pessoas interessadas em diferentes formas de documentos no contexto de um projeto de SL. Como nosso objetivo não é abordar o público-alvo para o qual a documentação é produzida, vamos nos ater a aspectos práticos de produção de documentação. Tanto desenvolvedores quanto usuários esperam obter algum tipo de documentação relativa ao projeto, onde, deve existir pelo menos os seguintes recursos voltados à documentação:

Arquivo LEIAME Documento de texto contendo alguns tópicos referentes à instalação, compilação, configuração... Enfim, contém informações mínimas para fazer com que o programa execute na máquina de um usuário. Não deve ser um arquivo grande e cansativo, deve conter apenas informações essenciais para utilização do software.

Arquivo INSTALL Deve ser um arquivo contendo apenas informações de como fazer a construção e instalação do programa. Mostra quais são os comandos básicos utilizados para instalação e quais são as opções de uso destes comandos.

Arquivo CHANGELOG É um arquivo que informa as mudanças, as correções e funcionalidades adicionadas que foram efetuadas no programa entre releases diferentes. É interessante colocar estas informações, também, na página do projeto, pois facilita aos usuários o seu acesso e os ajuda na decisão de utilizar a nova versão, ou esperar por modificações mais relevantes.

FAQ Um FAQ (Frequent Asked Questions) é utilizado para colocar questões que são repetidamente feitas a respeito do projeto em um único local, de forma a evitar replicação de informações. Uma política interessante para se montar um FAQ é quando uma mesma questão for postada mais de duas vezes em listas de discussões é útil pô-la no FAQ. Todos os documentos citados acima podem estar em formato HTML e acessíveis através da navegação na página do projeto. Outros documentos serão criados durante a evolução do projeto, tais como manuais ou tutoriais.