2 - Software Livre

Este capítulo trata um pouco mais profundamente do assunto Software Livre, discorrendo brevemente sobre sua comunidade, sua definição, iniciativas baseadas em software livre no Brasil, algumas características marcantes do processo e como este procura garantir que um software seja construído adequadamente.

2.1 - A comunidade de software livre

A comunidade de software livre é composta por membros altamente colaborativos que mesmo estando envolvidos em um mesmo projeto, não se conhecem pessoalmente já que na maioria dos casos se encontram separados por grandes distâncias geográficas, interagindo apenas através de ferramentas disponíveis na Internet. Isso é verdade para a maioria dos casos, mas cabe lembrar que existem exceções, não é incomum que membros de um projeto deste tipo façam todos parte de uma única empresa, que pode vir inclusive a começar um projeto como comercial e mais tarde transformá-lo em um software livre. Podemos citar como exemplo deste último caso, sistemas famosos como o Mozilla, o compilador Java Jikes da IBM e o Java Development Kit da Sun [Godfrey00].

Outra característica muito importante desta comunidade é o fato de que a grande parte das pessoas envolvidas em projetos de software livre trabalha como voluntário e não recebe retorno financeiro. Vale observar que essa característica também não se aplica a todos os casos, já que existem vários projetos deste tipo que são feitos ou mesmo patrocinados por empresas (é comum grandes empresas como IBM e Sun distribuírem pessoal para certos projetos de software livre em que tenham interesse).

Os retornos obtidos por membros destes projetos incluem o reconhecimento dentro da comunidade de software, a melhoria do software em questão (já que

usualmente o membro é usuário deste software) e o fato de estar contribuindo para a expansão dos software livres no mercado atual.

O fato de estar fazendo um trabalho voluntário traz algumas implicações que acabam por se tornar diferenças cruciais entre um projeto de software livre e um projeto tradicional. Alguns destas diferenças estão relacionadas a:

- prazo de entrega: Como o membro está realizando um trabalho voluntário, ele se utiliza de seu tempo livre para trabalhar no projeto e não tem um compromisso com prazos de entrega.

- motivação dos desenvolvedores: Também pelo fato deste ser um trabalho voluntário, o projeto é algo que o programador tem interesse em fazer, mesmo que este interesse se manifeste apenas por uma determinada área do projeto e não pelo todo, ainda é o suficiente para que a motivação deste indivíduo seja alta. É provável também que ele venha a ter um bom conhecimento do domínio o qual mostrou interesse.

- período entre lançamento de novas versões: Como não há compromisso com prazos, o produto só será lançado quando seus desenvolvedores acharem que ele está suficientemente maduro, o que nem sempre ocorre em projetos tradicionais que sempre estão dependentes de prazos muita vezes apertados, o que pode vir a gerar o lançamento prematuro de um software.

- testes, manutenção, planejamento e documentação: Estas tarefas para evolução em um projeto de software livre nem sempre são consideradas tão interessantes quando programar, o que acaba por gerar uma falta de atividade nestas áreas. A qualidade do código em um projeto de software livre é mantida geralmente por massivos testes paralelos (grupo de usuários testando o software e reportando bugs encontrados), ao invés de testes sistemáticos [Godfrey00].

2.2 - Definição de Software Livre

Como já foi dito anteriormente, não existe uma definição exata do termo "Software Livre", mas geralmente a definição usada é a de que "Software Livre é um tipo de software que é distribuído de acordo com os termos estabelecidos pelo Open Source Definition" [Joseph02].

Este documento é mantido pela Open Source Initiative(OSI), e não deve ser entendido como uma definição de licença, e sim uma especificação de como o software deve ser distribuído. Para o produto ser considerado Software Livre, e capaz de receber um certificado OSI, este deve estar de acordo com todos os critérios da especificação sem exceção.

Os critérios são os seguintes, de acordo com a versão 1.9 do Open Source definition localizado em [OSI03]:

1. Redistribuição gratuita

A licença de distribuição não deve de maneira alguma restringir a nenhuma das partes interessadas de vender ou ceder o software como componente de uma distribuição de um software agregado contendo programas de várias fontes diferentes. A licença não deve cobrar direitos de propriedade ou outras taxas pela venda do programa.

Fundamentação: Ao forçar a licença a exigir a redistribuição gratuita, eliminamos a tentação de desfazer-nos de muitos ganhos de longo prazo, a fim de fazer algumas poucas vendas imediatas. Se não fizéssemos isso, haveria muita pressão para a perda de colaboradores.

2. Código fonte

O programa deve incluir o seu código fonte e deve permitir a sua distribuição assim como a distribuição em forma compilada. Quando o produto não for

distribuído com o código fonte, deve haver uma forma claramente anunciada de obter o código fonte, sem custo, via Internet. O código fonte deve ser o recurso preferencial utilizado pelo programador para modificar o programa. Não é permitido ofuscar deliberadamente o código fonte (ato de embaralhar o código fonte). Também não são permitidas formas intermediárias como a saída de um pré-processador ou tradutor.

Fundamentação: É requerido acesso a um código fonte não ofuscado porque entendemos que não se podem melhorar os programas sem modificá-los. Como nosso objetivo é facilitar a evolução, exigimos que as mudanças sejam facilitadas.

3. Trabalhos derivados

A licença deve permitir modificações e trabalhos derivados e deve permitir distribuí-los sob os mesmos termos da licença do software original.

Fundamentação: simples habilidade de ver o código fonte não é suficiente para apoiar a revisão independente e a rápida seleção evolutiva. Para que a rápida evolução se concretize, as pessoas devem ser capazes de realizar experimentos e distribuir modificações.

4. Integridade do código fonte do autor

A licença somente pode restringir a distribuição do código fonte em forma modificada se ela permitir a distribuição de remendos (*patchs*) junto com o código fonte original com o objetivo de modificar o programa durante a compilação. A licença deve permitir explicitamente a distribuição de software compilado a partir do código fonte modificado. A licença pode exigir que trabalhos derivados tenham nomes ou números de versões diferentes do software original.

Fundamentação: Mesmo valorizamos a melhoria e o progresso, é entendido que os usuários tem o direito de saber quem é responsável pelo software que utilizam. Autores e mantenedores têm direitos recíprocos de saber o que devem apoiar e de proteger suas reputações.

Dessa forma, uma licença de software livre deve garantir que a fonte esteja sempre disponível, mas pode requerer que seja distribuída como código fonte original com remendos. Portanto, mudanças não oficiais podem ser colocadas à disposição, mas imediatamente distinguíveis do código fonte original.

5. Não descriminar pessoas ou grupos

A licença não deve discriminar nenhuma pessoa ou grupo de pessoas.

Fundamentação: A fim de maximizar o benefício do processo, a maior diversidade de pessoas e grupos deve ter acesso igualitário aos software livres. Portanto, é proibido a qualquer licença de software livre excluir qualquer pessoa do processo.

6. Não discriminar campos de interesse

A licença não deve restringir ninguém a utilização do programa em algum campo de interesse específico. Por exemplo, a licença não deve restringir a utilização do programa em algum negócio ou em atividades de pesquisa genética.

Fundamentação: A principal intenção desta cláusula é proibir armadilhas na licença que previnam a utilização comercial de software livres. É desejado que usuários comerciais integrem a comunidade e não se sintam excluídos da mesma.

7. Distribuição da licença

Os direitos vinculados ao programa devem ser aplicados a todos aqueles aos quais o programa é redistribuído, sem que haja necessidade que as partes levem a efeito uma licença adicional.

Fundamentação: Esta cláusula tem por objetivo proibir o fechamento do programa por formas indiretas como a exigência de uma declaração de não-divulgação.

8. Uma licença não deve ser específica para um determinado produto

Os direitos vinculados a um programa não devem depender de que o referido programa seja parte de uma distribuição de software específica. Se o programa é extraído daquela distribuição e usado ou distribuído nos termos da licença do programa, todas as partes para as quais o programa é redistribuído devem ter os mesmos direitos que são concedidos às pessoas que receberam o software original.

Fundamentação: Esta cláusula visa prevenir alguns tipos de armadilhas de licença.

9. Uma licença não deve restringir outros sistemas

A licença não deve impor restrições a outro software que é distribuído junto com o software licenciado. Por exemplo, a licença não deve insistir que todos os outros programas distribuídos no mesmo meio sejam software livres.

Fundamentação: As pessoas que querem utilizar ou redistribuir software livres tem direito de tomar suas próprias decisões sobre seu próprio software.

10. A licença deve ser neutra quanto à tecnologia

Nenhuma parte da licença deve ser atrelada a nenhuma tecnologia individual ou estilo de interface.

Fundamentação: Este critério visa especificamente licenças que requerem um gesto explícito por parte do usuário para que seja estabelecido um contrato entre o licenciador e o licenciante. Estes métodos podem conflitar com importantes meios de distribuição de software, como FTP, CD-ROM e espelhamento em sites.

2.3 - Licenças

Estando de acordo com estes 10 critérios, o produto é considerado elegível para obter o certificado OSI (OSI Certified).

Para um software obter esse certificado, basta que este esteja de acordo com alguma licença consistentes com a Open Source Definition. Para obter a lista completa destas licenças, basta acessar a página da OSI [OSI03], que mantém uma lista constantemente atualizada com todas as licenças que já foram aprovadas.

Existem diversas licenças que se encaixam nesta categoria, algumas das mais conhecidas são:

- BSD License
- GNU General Public License (GPL)
- GNU “Library” or “Lesser” Public License (LGPL)
- IBM Public License
- Intel Open Source License
- MIT License
- Mozilla Public License 1.0 e 1.1
- Sun Public License

Dentre estas, as três primeiras licenças merecem um detalhamento maior:

Tanto a GPL quanto a LGPL foram criadas por Richard Stallman e são extremamente populares. A característica marcante do GPL é que esta é uma licença “viral” ou seja, qualquer usuário pode modificar como quiser um software

com essa licença, mas caso este em algum momento lance uma versão pública deste software, ele deve ser distribuído de acordo com os termos da GPL.

A LGPL é uma alternativa não viral da GPL. Esta foi criada visando o uso de bibliotecas em outros sistemas. Neste caso, se um software fizer uso de uma biblioteca que possua uma licença LGPL, e estiver apenas fazendo uso de suas funções externamente e não usando seu código dentro do software, então este software não precisa ser LGPL.

Vale notar que o criador de ambas essas licenças recomenda fortemente o uso da GPL em detrimento da LGPL mesmo para bibliotecas, já que segundo ele, essa licença resulta em uma vantagem para os desenvolvedores de software livre e sua comunidade, além de uma desvantagem para os outros desenvolvedores que não poderão fazer uso de sistemas com essa licença em projetos que não sejam livres.

A licença BSD é bem menos exigente que as anteriores, e faz apenas algumas exigências quanto à manutenção dos devidos créditos e as listas de condições do software original, além de exigir que nomes dos autores originais ou subseqüentes não sejam usados para promoção de sistemas derivados.

As três licenças acima citadas são muito populares, e somente no site SourceForge [SourceForge03], em Julho de 2003, dos 42585 projetos que usavam licenças certificadas pela OSI, 30250 (71,03%) usavam a licença GPL, 4495 (10,55%) usavam LGPL e 2945(6,91%) usavam a licença BSD.

Caso se deseje escolher uma licença aprovada pela OSI para um projeto de software livre, deve-se ler com cuidado os termos das licenças já aprovadas e escolher a que melhor se encaixa para o projeto.

A vantagens de se usar uma licença aprovada pela OSI em um projeto é o reconhecimento que estas licenças tem com a comunidade de software livre. Deste modo, mesmo que este projeto possua uma licença pouco conhecida, o fato desta

ser aprovada pela OSI, faz com que esteja de acordo com os 10 itens anteriormente citados, gerando, portanto um maior retorno por parte da comunidade.

Vários software livres de nome usam licenças aprovadas pela OSI, dentre estes podemos citar o Apache HTTP Server, BSD, Mozilla, PHP e MySQL (apesar de que também existe uma licença comercial para este).

2.4 - Software Livre no Brasil

No Brasil o movimento de software livre tem crescido expressivamente, conseguindo inclusive o apoio de universidades e órgãos governamentais, que enxergam no software livre um caminho economicamente viável para a democratização do acesso aos recursos da informática no país. Vários congressos e workshops com foco exclusivo em software livre estão sendo realizados pelo país, como exemplo podemos citar o Workshop Sobre Software Livre (WSL) que teve sua quarta edição realizada em junho de 2003 [WSL03].

A seguir apresentamos exemplos de iniciativas na área de software livre no Brasil:

2.4.1 - Projeto Rede Escolar Livre RS:

Um projeto pioneiro do governo do estado, com a participação da Secretaria Estadual de Educação e da Companhia de Processamento de Dados do Estado.

Este projeto visa disponibilizar o uso da informática nas escolas públicas estaduais do Rio Grande do Sul, possibilitando assim a inclusão dos alunos, professores, funcionários e da comunidade escolar no novo mundo que se apresenta através da tecnologia da informação [Rel01].

Um projeto piloto já foi feito inicialmente em cinco escolas da rede estadual, e o programa visa beneficiar todas as escolas da rede pública estadual

com mais de 100 alunos, disponibilizando para estes, laboratórios de informática com 10 microcomputadores ligados em rede local, e utilizando software livres com acesso à Internet. Entre os sistemas utilizados, estão o Linux e o conjunto de ferramentas do StarOffice. Como todos os sistemas utilizados são livres, estes podem ser copiados livremente em todas as máquinas, gerando uma economia estimada de cerca de R$ 40 milhões para o projeto.

2.4.2 - Incubadora virtual de projetos de software livre

O Centro Universitário Univates, localizado no Rio Grande do Sul, disponibiliza seu ambiente de apoio ao desenvolvimento colaborativo de software livre através do portal CodigoLivre (antigamente chamado de CodigoAberto). O Univates utiliza uma versão baseada no software usado pelo famoso portal sourceforge [SourceForge03], para hospedar gratuitamente projetos de software livre brasileiros. Com 3 anos de existência, o CodigoLivre hospeda mais de 480 projetos, mantidos por mais de 3500 colaboradores, e sua estrutura está sendo movida para a Unicamp, que em conjunto com o Univates passará a administrar o ambiente [CodigoLivre03].

2.4.3 - HOSPUB

O ministério da Saúde, através do DATASUS, também está fomentando o desenvolvimento de aplicações para a área da saúde pública. Um dos resultados é o HOSPUB, que é um sistema integrado para informatização hospitalar.

HOSPUB é um sistema on line e multiusuário, desenvolvido em um ambiente operacional de banco de dados relacional, e tem por objetivo suprir as necessidades dos diversos setores/serviços existentes em uma unidade Hospitalar, para atendimento secundário e/ou terciário. Além disso, é uma ferramenta eficaz para prestar informações que possam subsidiar os diferentes níveis hierárquicos que compõem o SUS, seja no processo de planejamento, de operação ou de controle das ações em saúde.

O HOSPUB é de domínio público e encontra-se à disposição de qualquer interessado vinculado à rede assistencial do SUS. Vale lembrar entretanto que este software pode ser distribuído livremente, mas seu código não é aberto [HOSPUB03].

2.4.4 - SuperWaba

Criado por um brasileiro, SuperWaba é uma máquina virtual para handhelds que é conhecida e usada no mundo todo, esta alia a facilidade da linguagem Java com todo o potencial dos dispositivos móveis.

Guilherme C. Hazan, seu criador, se baseou em outra ferramenta famosa, o Waba. Esta ferramenta era bem difundida, e se propunha a levar ao mundo dos handhelds a simplicidade da linguagem Java. A sua maior qualidade era sua simplicidade, mas esta acabou por se tornar também o seu maior defeito, pois essa simplicidade em muitos casos implicava em limitações para a aplicação. Criado inicialmente apenas como algumas funções adicionais do Waba, o SuperWaba cresceu tanto que superou o programa original tanto em popularidade, qualidade e número de usuários. Até julho de 2003, essa máquina virtual tinha mais de 14000 usuários cadastrados, e era usada por diversas empresas de portes variados pelo mundo todo [SuperWaba03].

O SuperWaba é distribuído sob a licença Lesser General Public License (LGPL), sua distribuição é gratuita, e seu código fonte é aberto. A única diferença é que o autor recebe retorno financeiro da venda de tutoriais, *add-ons* e suporte do SuperWaba.

2.4.5 - Projeto Eclipse@Rio

A plataforma Eclipse é uma proposta de um consórcio de empresas que apóiam o uso de uma arquitetura aberta para uso na construção de ambientes integrados de desenvolvimento (IDEs) de forma que esta possa ser usada na criação de aplicações diversas como páginas web, programas Java, programas em C++ e Entreprise JavaBeans [Eclipse03].

Existe um grande interesse na plataforma por parte de empresas de grande porte; um sinal claro deste fato é o número sempre crescente de empresas que fazem parte do consórcio. Podemos citar como exemplo empresas como a IBM, Borland, Red Hat, Fujitsu, Ericsson e Oracle. A licença utilizada é a Commom Public License (apesar de que alguns componentes possam vir a ter uma licença diferente), que permite a livre distribuição do código fonte e trabalhos derivados do código original.

O projeto Eclipse@Rio é desenvolvido pelo Laboratório Teccomm [Teccomm03] da Pontifícia Universidade Católica do Rio de Janeiro, e foi um dos vencedores do concurso Eclipse Innovation Grants, realizado pela IBM no final de 2002 [Eclipse@Rio03]. Este projeto visa ajudar a disseminar a plataforma Eclipse a uma comunidade mais ampla de profissionais e pesquisadores. Essa disseminação é feita através de palestras e workshops gratuitos, a adoção da ferramenta em disciplinas obrigatórias e eletivas da graduação e pós-graduação e desenvolvimento de *plugins* gratuitos para a plataforma Eclipse.

As palestras e workshops servem como ponto de encontro para que profissionais interessados venham a conhecer ou aprimorar seus conhecimentos na plataforma Eclipse, já a adoção da plataforma nas disciplinas visa familiarizar o aluno com uma ferramenta largamente utilizada no mercado de trabalho e acadêmico.O desenvolvimento de *plugins* busca a criação de um conjunto maior e mais detalhado de ferramentas para a plataforma. Todos os *plugins* são software livres, seus códigos fontes são abertos.

2.4.6 – Lua

Desenvolvida no Departamento de Informática da Puc-Rio [Puc03], mais precisamente no Tecgraf [Tecgraf03], pelos professores Roberto Ierusalimschy, Waldemar Celes e Luiz Henrique de Figueiredo, Lua é uma linguagem de programação poderosa e leve, esta foi projetada com o intuito de estender e facilitar as aplicações de outras linguagens mais pesadas, como C ou C++, às quais se mistura de modo a ordenar as ações que devem ser executadas pelos programas.

Embora não seja uma linguagem puramente orientada a objetos, ela fornece meta-mecanismos para a implementação de classes e heranças, e são justamente estes que fazem com que a linguagem seja mantida pequena, ao mesmo tempo em que a semântica seja estendida de maneiras não convencionais (o que é uma característica marcante de Lua). Lua está implementada como uma pequena biblioteca de funções C, escritas em ANSI C, que compila sem modificações em todas as plataformas conhecidas. Os objetivos da implementação são simplicidade, eficiência, portabilidade e baixo impacto de inclusão em aplicações [Lua03].

Essa linguagem é bastante conhecida e utilizada por programadores de todo mundo, mas curiosamente ela é pouco conhecida no Brasil. Desde sua criação, diversos projetos de porte variados foram e são feitos no Tecgraf utilizando a linguagem, mas Lua está presente em diversos outros projetos pelo mundo. Podemos citar como exemplo empresas de entretenimento como a LucasArts Entertainment e a Bioware Corp. A primeira fez uso da linguagem em jogos como Escape From Monkey Island e Grim Fandango, e a segunda a usou para fazer o script interno do seu famoso jogo Baldur's Gates.

Lua é um software livre desde seu nascimento quando usava uma licença própria, mas a partir da versão 5.0, a licença usada será a MIT License, que é extremamente flexível, permitindo que seu código fonte e produtos derivados sejam usados e distribuídos livremente, sejam em projetos comerciais ou software livres.

2.4.7 – Aulanet

AulaNet é um ambiente baseado numa abordagem groupware para o ensino-aprendizagem na Web que vem sendo desenvolvido desde junho de 1997. Inicialmente, este servidor de cursos a distância foi desenvolvido pelo conceituado Laboratório de Engenharia de Software (LES) da PUC-Rio [LES03].

No ano de 1998 iniciou-se a parceria entre a PUC-Rio [Puc03] e a EduWeb [EduWeb03] (então empresa residente da Incubadora da PUC-Rio) e neste mesmo ano foi feito o lançamento externo à universidade.

O AulaNet é distribuído gratuitamente para qualquer instituição, que possua interesse em utilizar a tecnologia para criar e manter cursos utilizando a Web como ferramenta. Os maiores interessados na ferramenta são grandes corporações, portais verticais e horizontais, instituições de ensino (universidades e colégios), órgãos governamentais, consultores e usuários da Internet em geral. Até Julho de 2003, a ferramenta já tinha sido baixada 4500 vezes e vem sendo usada em mais de 50 instituições no Brasil inteiro.

Apesar de ser distribuído gratuitamente, a ferramenta AulaNet assim como o HOSPUB, não disponibiliza o código fonte para o usuário. Neste caso o direito de desenvolver novas versões, adaptar novas funcionalidades e distribuir a ferramenta é exclusivo da empresa EduWeb.

2.5 - Características do processo de desenvolvimento de software livre

Inicialmente vale salientar que não existe um processo de desenvolvimento de software livre que seja considerado “padrão”, diferentes desenvolvedores e comunidades empregarão técnicas, métodos e ferramentas diferentes.

Embora estas diferenças existam, ainda assim é possível delinear nos diferentes projetos várias características comuns. Descreveremos aqui estas características, lembrando que mesmo estas podem vir a sofrer alterações significativas em alguns projetos mais específicos.

2.5.1 Estilos de software livres

Segundo Raymond [Raymond98] um projeto de software livre pode ter várias formas, mas dois estilos de desenvolvimento são considerados dominantes na comunidade: a catedral e o bazar.

2.5.1.1 A catedral

Sistemas desenvolvidos neste estilo são normalmente fruto do trabalho de um grupo pequeno de programadores ou mesmo de um único programador. Contribuições de código ou idéias vindas da comunidade são aceitas, mas estas não são a principal preocupação dos autores originais. Estes monopolizam grande parte das decisões de design, implementação e a data de liberação de determinada versão, o que faz com que o tempo entre o lançamento de versões se torne maior e a contribuição por parte da comunidade diminua.

2.5.1.2 O bazar

Ao contrário da catedral, este estilo é bem menos conservador, e trabalha com ciclos de desenvolvimento bem menores. Muitos projetos deste tipo ainda são controlados por um pequeno grupo de desenvolvedores, mas a contribuição de código é muito encorajada e inclusive esse fato é um dos motivos que leva o ciclo de desenvolvimento a ser tão pequeno. Este estilo de desenvolvimento faz uso intensivo de ferramentas de Groupware (bate-papos, listas de discussão, IRC) entre seus participantes como uma forma de estimular a troca de idéias. É comum também a existência de um repositório global de código onde as contribuições são feitas.

2.5.2 Início de um projeto

Um projeto normalmente surge devido a uma necessidade ou mesmo curiosidade de um grupo de desenvolvedores (ou mesmo como um projeto comercial dentro de uma empresa). Muitos projetos grandes começaram apenas com a vontade de aprender por parte dos autores originais, como foi o caso do Linux que surgiu da vontade de seu autor Linus Torvalds de criar um sistema operacional como o Unix que executasse em computadores PC 386.

Após o surgimento idéia original, cabe ao grupo de autores disponibilizar essa idéia para a comunidade de software livre e pedir ajuda para possíveis interessados. Caso exista alguma especificação ou código pronto, este é disponibilizado para a comunidade.

Como existe um interesse por parte dos desenvolvedores originais na ajuda que a comunidade pode oferecer, o modo como a divulgação da idéia é feita deve abranger o máximo número possível de colaboradores. Uma opção muito usada é a de hospedar o projeto em um dos serviços de hospedagem de projetos gratuitos que existem na Internet. Dentre eles destacamos o famoso SourceForge [SourceForge03], que oferece uma página para a hospedagem, ferramentas para a interação entre desenvolvedores, controle de versão (CVS) e serve como ponto de encontro virtual entre programadores interessados em software livre.

2.5.3 Requisitos em um projeto de software livre

Segundo [Massey01] existem três grandes fontes para obtenção de requisitos em um projeto de software livre: os próprios autores, os usuários e alguns padrões.

Os autores originais do projeto se encarregam de elaborar os requisitos iniciais do projeto; essa prática não é muito comum em sistemas comerciais, onde é normal ter uma pessoa ou equipe que não faz parte do grupo de desenvolvedores como responsável pelo levantamento dos requisitos. Em um projeto de software livre, como o autor está consciente do problema, é normal que este consiga fazer um *design* bom o suficiente para resolvê-lo, mas sempre existirá o problema de que requisitos levantados por uma pessoa ou um grupo pequeno de pessoas, nem sempre estarão de acordo com as necessidades da comunidade inteira.

Para resolver este problema existe outra fonte de obtenção de requisitos, esta sendo a comunidade de usuários do projeto. Quanto maior for a base de usuários, maior será o retorno obtido, este retorno vem das experiências destes usuários com o projeto, e pode vir na forma de comunicação de erros, sugestão de melhorias e dúvidas. Com esse retorno, é possível complementar os requisitos iniciais do sistema de forma que isso reflita num código muito mais robusto que o original.

A terceira forma de se levantar requisitos vem do fato que muitos projetos de software livres se iniciam do desejo de se implementar padrões já existentes. Desta forma a elicitação não ocorre, pois o requisito (o padrão) já está definido. Estranhamente, esta forma de se levantar requisitos é em muitos casos ignorada por projetos comerciais, que preferem criar padrões próprios que esta possa patentear, a usar um já existente no mercado. Um exemplo seria a tecnologia JDO (Java Data objects), que é uma especificação feita com ajuda do *Java Community Process* que serve para transformar modelos de domínio Java em dados persistentes. Apesar de existirem alguns poucos projetos comerciais que implementam essa tecnologia, ela ainda é ignorada por muitos outros que preferem fazer uso de uma especificação própria para esse fim.

Deve-se esclarecer que essas três fontes citadas por [Massey01] servem apenas para explicar em linhas gerais como a obtenção de requisitos funciona em um projeto de software livre. Essa obtenção não é trivial como pode vir a parecer.

2.5.4 Desenvolvimento paralelo

Após a disponibilização do projeto para a comunidade de software livre, este estará apto a receber contribuições de desenvolvedores do mundo todo. Cada um destes manterá foco na sua área de interesse e desenvolverá seu código em paralelo aos demais.

Desenvolvimento paralelo ao invés de linear é uma característica muito forte do processo de desenvolvimento de software livre, esta característica é facilitada pela natureza modular da maioria destes sistemas.

Desenvolvimento paralelo pode implicar tanto em dois ou mais programadores estarem trabalhando em módulos distintos, ou estarem trabalhando no mesmo módulo. A primeira opção é comum, pois através das ferramentas de groupware disponíveis para o projeto, é possível saber em qual módulo se está realizando trabalho ou não, este tipo de desenvolvimento aumenta em muito a produtividade já que um módulo não precisa necessariamente influenciar em um

outro. A segunda opção ao contrário do que ocorre em projetos comerciais, também é comum.

Em um projeto comercial, distribuir várias pessoas para trabalhar em um mesmo módulo paralelamente não é financeiramente viável, mas como desenvolvedores de software livre fazem código por prazer, é comum que alguns trabalhem em módulos de seu interesse mesmo sabendo que já existe alguém o fazendo. Este último caso faz com que no final, apenas o melhor dos código seja colocado no projeto, o que contribui para o aumento de qualidade do produto final.

2.5.5 Retorno por parte dos usuários

Não é necessário que um determinado usuário seja um desenvolvedor para contribuir com um projeto de software livre. Na verdade, apenas uma pequena parcela de usuários de um projeto contribui efetivamente com código, mas uma parcela bem maior contribui com relatos de erros e sugestões de melhoria.

Este retorno dado pelos usuários é um ciclo constante de encontrar erro, reportar, corrigir e lançar a correção na próxima versão do produto. Quanto maior o número de usuários de um determinado projeto, maior é a eficiência desta seqüência, o que novamente nos leva ao fato de que a disponibilização adequada de um projeto pode influenciar muito em seu sucesso.

Em alguns casos de projetos de grande porte, a resposta a um erro mais grave pode ser quase instantânea, como no caso do ataque conhecido como "*Ping of Death*" que foi resolvido no Linux apenas algumas horas após sua descoberta [Joseph02]. Em um outro caso, foi reportado que 50% dos problemas enviados por usuários do Apache HTTP Server são resolvidos no intervalo de um dia, 75% são resolvidos em até 42 dias e 90% em até 140 dias [Mockus02].

2.5.6 Uso de ferramentas de Groupware

Como já foi dito anteriormente, o desenvolvimento de um software livre é feito de uma forma geral por comunidades distribuídas geograficamente. Estas comunidades usam a Internet para se comunicar e colaboram entre si para chegar em um objetivo comum, utilizando ferramentas de Groupware.

As ferramentas usadas são as mais simples possíveis, como ferramentas de correio eletrônico e listas de discussão para a troca de mensagens entre os desenvolvedores e usuários. Nestas listas é possível discutir qualquer assunto relacionado ao projeto, como adição de novas funcionalidades, correção de erros, novos requisitos; também é possível dar suporte aos usuários. É comum o uso de ferramentas de bate papos como o IRC (Internet Relay Chat) para discussões que requeiram mais dinamismo.

Outra ferramenta usada por quase todos os projetos, é a de controle de versão. A mais popular delas é o CVS que também é um projeto de software livre e é importantíssima para se controlar o repositório do projeto e suas versões.

Vale lembrar que existem serviços de hospedagem de projetos de software livre que já oferecem várias ferramentas de Groupware como ferramentas de listas de discussão e repositórios para o CVS, um exemplo seria o popular Sourceforge [SourceForge03].

2.5.7 Alta motivação dos participantes

O fato da contribuição de código ser voluntária cria um efeito interessante no projeto. Todo código enviado é feito por pessoas com interesse na área específica do código, e mesmo que a área em questão seja apenas uma pequena parte do todo, essa contribuição terá vindo de uma pessoa altamente motivada pelo interesse no sucesso do produto final.

Essa motivação pode vir da vontade de aplicar conhecimentos, aprofundar o estudo em determinada área ou mesmo da simples vontade de ter seu código

examinado por alguma autoridade da área (um ou mais dos administradores do projeto). O fato de apenas o melhor código enviado ser efetivamente colocado em alguma versão do projeto, faz com que haja um aumento na motivação para a criação de um código de maior qualidade.

Outro ponto interessante é o fato de muitas contribuições serem feitas por grandes nomes da área que muitas vezes são possuidores de altos salários nas empresas em que trabalham, e que mesmo assim gastam seu tempo livre desenvolvendo código gratuitamente. Esse fato nos leva a crer que o dinheiro nem sempre é um fator motivante para um desenvolvedor.

2.5.8 Freqüente lançamento de novas versões

Devido a dinâmica do processo de desenvolvimento, comunicação, distribuição e suporte, a freqüência da liberação de novas versões de um produto de software livre é bem maior que a de um comercial. Existem casos de projetos que em determinados períodos liberam mais de uma nova versão por dia, como o Linux no período de 1991 [Joseph02].

Este item é bem mais perceptível em projetos de software livre feito no estilo de desenvolvimento do bazar e menos no estilo da catedral.

2.6 - Os quatro atributos que um software deve possuir

Independente do estilo do projeto (bazar ou catedral), ou se este é um software livre ou não, de acordo com Sommerville [Sommerville92] um software produzido adequadamente deve possuir quatro atributos, a saber:

- **manutenabilidade** - um software que necessite ter uma expectativa de vida alta, deve ser bem documentado e escrito, de modo a facilitar mudanças no código quanto estas forem necessárias.

- **confiança** - um software deve fazer o que lhe é esperado pelos usuários, não sendo aceitável que este falhe mais do que o permitido pelas suas especificações.

- **eficiência** - não deve ser feito mal uso dos recursos computacionais do sistema, como memória e CPU.

- **uma interface apropriada com o usuário** - deve-se ter em mente o nível de conhecimento do usuário final para se realizar a elaboração da interface. Esta interface deve ser feita de forma que o usuário consiga tirar máximo proveito do software.

A forma de desenvolvimento não muda em nada o fato de que o software livre também deve atender esses quatro atributos.

Um software livre procura satisfazer estes quatro atributos com algumas de suas características mais fortes; desenvolvimento paralelo, retorno por parte dos usuários, alto nível de motivação dos participantes e lançamento freqüentes de versões.

Manutenabilidade é um tópico de grande importância para um software livre, pois um software deste tipo está em constante evolução, portanto um desenvolvedor deve sempre programar pensando em facilitar mudanças futuras no software, seja esta feita por ele ou outro membro do projeto. Este tópico será tratado com mais detalhes no capítulo 3.

A confiança de um software depende muito do que este propõe em suas especificações, certos sistemas de uso crítico não podem falhar de maneira alguma, um exemplo de sistemas que não podem falhar são quaisquer um que ao falhar possa causar risco a vidas humanas (como sistemas que controlam aparelhos de hospital), já outros sistemas cujas especificações sejam menos exigentes, podem suportar um número maior de falhas sem comprometer seu funcionamento satisfatório.

Para procurar garantir confiança, os desenvolvedores de software livres avaliam o retorno por parte dos usuários. Caso no projeto ou em alguma versão deste, os usuários reportem um número de falhas acima do permitido nas especificações, é feito um esforço por parte da equipe do projeto em contornar essa situação no próximo ciclo de desenvolvimento.

Eficiência de um software consiste em fazer bom uso dos recursos computacionais do sistema. Saber utilizar satisfatoriamente os recursos disponíveis é uma das qualidades de um bom desenvolvedor, o quão importante é essa economia para o projeto dependerá das especificações deste, já que existem projetos que necessitam de cada bit disponível (como projetos em hardwares com menos capacidade computacional, como Palms e celulares), e outros nem tanto.

Uma das características do desenvolvimento de software livre que ajuda em relação a este atributo, é o desenvolvimento em paralelo entre os membros do projeto. Como podem existir mais de um desenvolvedor mexendo em uma mesma parte do código, o responsável pelo projeto escolherá apenas o melhor destes códigos para colocar na versão oficial do projeto, fazendo desta forma que caso seja necessário para o projeto, apenas os códigos mais eficientes sejam usados. Esta “disputa” entre códigos também ajuda a melhorar a qualidade geral destes, já que os desenvolvedores saberão que apenas o melhor deles será utilizado, e farão um esforço extra para que sejam seus códigos os escolhidos.

Ao se analisar o último atributo, deve-se ter em mente que para se produzir uma interface que seja apropriada, é necessário saber o nível de conhecimento técnico do usuário. Caso o usuário seja uma pessoa que não esteja familiarizada com o processo usado no software, deve-se ter um cuidado para que a interface deixe esse processo o mais claro possível. Construir uma interface é sempre um grande desafio para a equipe do projeto, pois deve-se achar o equilíbrio entre a clareza nas informações e a simplicidade dos elementos exibidos.

Para se chegar a uma interface que seja satisfatória para o usuário, os desenvolvedores de software livre novamente fazem uso extensivo do retorno gerado pelos usuários, pois analisando estes, poderá se saber se esta interface foi bem aceita, se ela é de difícil compreensão ou se falta algo nesta. De posse deste retorno, correções serão feitas nas próximas versões do software, visando um melhor nível de satisfação para um maior número de usuários.