99

Software Livre

Igor Steinmacher, Igor Wiesel, Gustavo Pinto e Marco A. Gerosa

Após a leitura deste capítulo, você será capaz de:

- Entender o histórico do software livre;
- Saber o que diferencia software livre e não-livre;
- Compreender os passos e atividades envolvidos no processo de contribuição para projetos de software livre;
- Entender os benefícios do software livre para a sociedade.

1. Software Livre: da criação até os dias de hoje

O surgimento do software livre deu-se no início da década de 1960. Apesar do termo "software livre" não ter surgido naquela época, os primeiros computadores vendidos pela IBM já continham código fonte aberto que pudesse ser melhorado e modificado. Entretanto, o marco principal surge na década de 80 quando Richard Stallman, até então programador do MIT AI Lab, criou o projeto GNU em 1983. O projeto GNU se tornou um meio para possibilitar que as pessoas se "libertassem" do software não-livre, garantindo que os usuários finais pudessem ter as liberdades de executar, estudar, compartilhar e modificar o código fonte dos programas criados. Em 1983, Stallman começou o projeto GNU que tinha o objetivo de criar um sistema operacional completo. Dois anos depois, o mesmo Stallman divulgou o Manifesto GNU[1] e criou a Free Software Foundation (FSF)[2], uma organização que se dedicava a promover o movimento do software livre que estava surgindo. Este movimento foi um marco importante na história do software livre, porque até então, os sistemas eram predominantemente não-livres.

Em 1991, Linus Torvalds, um estudante de ciência da computação na Finlândia, implementou a primeira versão do núcleo do Linux (*Linux Kernel*). Logo, muitas pessoas começaram a colaborar com o Linux adicionando mais funcionalidades com o objetivo de criar um sistema operacional estável e funcional. O núcleo do Linux foi lançado em conjunto com os aplicativos GNU, com todo o código fonte licenciado com a segunda versão da licença GNU GPL. Nessa mesma década, muitos novos projetos de software livre surgiram, dentre eles o servidor Apache httpd, a linguagem Perl, o GNOME, o KDE e o navegador Mozilla.

Outro grande fato marcante na popularização e divulgação do movimento de software livre foi o lançamento do livro "The Cathedral and the Bazaar: "Musings on Linux and Open Source by an Accidental Revolutionary" (RAYMOND, 1997) por Eric Raymond, em 1997.

---

1 https://www.gnu.org/gnu/manifesto.pt-br.html

2 https://www.fsf.org/

Em seu livro, Reymond reflete sobre o desenvolvimento do Linux e sobre sua experiência à frente do projeto fetchmail.

O sucesso do texto de Raymond foi muito influente e acelerou a adoção comercial do modelo livre em inúmeras empresas de software como a IBM, a HP e a RedHat. Esta influência também incentivou a criação da Open Source Initiative (OSI)[3], em 1988, por Raymond e Bruce Perens. A OSI, se tornou uma entidade sem fins lucrativos que trabalharia para promover as ideias do desenvolvimento de software livre por razões técnicas e sugerindo o uso da expressão *open source* ao invés de *free software*. A principal motivação para a adoção de um novo termo foi a tentativa de introduzir o software livre no ambiente corporativo evitando a ambiguidade do termo *free*, que poderia significar tanto livre quanto gratuito, na língua inglesa.

Em 1999 foi fundada a Apache Software Foundation, uma importante entidade que criou e incentivou o desenvolvimento de projetos de software livre amplamente usado hoje na indústria, como por exemplo, o servidor HTTP, o banco de dados Derby e o arcabouço Hadoop. A partir de então grandes fundações e projetos de software livre surgiram, como por exemplo, a Openoffice.org também em 1999, o navegador Mozilla Firefox em 2003, o controle de versão Git em 2005 e o Android da Google, que atualmente é o mais popular sistema operacional para dispositivos móveis.

Hoje em dia, software livre é usado extensivamente em toda a pilha de desenvolvimento, desde bibliotecas essenciais até os aplicativos sofisticados de usuário final (web, móveis e desktop). Mesmo sistemas não-livre usam ou são executados em ambientes com software livre. O software livre é, definitivamente, uma força motriz no modelo de desenvolvimento de software atual. O 10º Levantamento Anual sobre o Futuro do Software Livre (BLACKDUCK, 2016) mostrou que 65% das empresas pesquisadas utilizam software livre para acelerar o desenvolvimento de aplicativos e 55% o utilizam como infraestrutura de produção.

Além disso, grandes empresas de software, bem conhecidas por serem restritivas quando se trata de publicar seus artefatos de código, têm recentemente não somente utilizado projetos de software livre em seus ambientes de trabalho (como forma de reúso de código), mas também têm adotado iniciativas de código aberto (seja através de hackathons ou financiando o desenvolvimento de projetos de software livre) (PINTO, 2018). É o caso de diversos produtos da Microsoft, Twitter, Facebook, Apple e Google. Um exemplo interessante é o projeto Swift, da Apple, que, na primeira semana de sua abertura, teve mais de 1.900 forks realizados e recebeu contribuições de mais de 200 desenvolvedores da comunidade externa em um mês.

---

3 https://opensource.org/

Software Livre no Brasil

No Brasil, o software livre também tem sua relevância e importância para a indústria, academia e governo. Na indústria, por exemplo, em 1995, a empresa Conectiva criou uma criou uma versão do GNU/Linux pronta para uso que se tornou reconhecida internacionalmente, sendo adquirida pela Mandrakesoft.

Na academia, Fabio Kon, orientado por Arnaldo Mandel, foi o primeiro brasileiro a programar dentro do kernel do Linux em sua pesquisa de mestrado (KON, 1994). Mais recentemente, em 2005, o CCSL-IME-USP (Centro de Competência em Software Livre do Instituto de Matemática e Estatística da USP) foi aprovado como projeto apoiado pela FINEP e hoje é um importante centro de competência que estimula o desenvolvimento de inúmeros projetos de software livre no Brasil. Também na academia, em 2007, a PUC-Rio em colaboração com a UFPB criou o GINGA - um middleware do Sistema Brasileiro de TV Digital de código aberto. Já a linguagem Lua, bastante utilizada em alguns nichos em todo o mundo, foi criada por um time de desenvolvedores do Tecgraf da PUC-Rio.

O governo brasileiro também tem papel importante no incentivo do uso de software livre. Na esfera federal existem iniciativas para o apoio à realização de eventos e portais que divulgam o software livre no Brasil. O mesmo acontece com estados, especialmente Rio Grande do Sul e Paraná. Além do aspecto governamental, a Associação Software Livre (ASL), uma associação civil sem fins lucrativos, reúne empresários, profissionais liberais, estudantes e servidores públicos, estabelecendo relações com os mais diversos setores da sociedade em torno do movimento de software livre. Fundada em 2003, a ASL é a entidade organizadora do Fórum Internacional Software Livre (FISL), que se tornou a maior conferência de software livre na América Latina.

2. Software Livre vs. Software Não-Livre (ou Proprietário)

A distinção entre software livre e os software não-livre, também conhecido como software proprietário, é relacionado à licença de software empregada. Para ser considerado software livre, um projeto de software precisa ser licenciado por (pelo menos) uma licença reconhecida por uma entidade promotora, como por exemplo a Open Source Initiative (OSI), assim evita-se a proliferação de licenças que são incompatíveis.

Importante mencionar que o software livre abrange licença de software, modelo de comunidade, ideologia e movimento social (JEWEL, 2010), sendo caracterizado por quatro liberdades:

- executar o programa, para qualquer propósito;
- estudar o programa e adaptá-lo para as suas necessidades;
- redistribuir cópias do programa de modo que você possa ajudar ao seu próximo;
- modificar (aperfeiçoar) o programa e distribuir as modificações, de modo que toda a comunidade se beneficie.

Dessa forma, qualquer software que viole qualquer uma das quatro liberdades mencionadas anteriormente, não podem ser considerados como software livre. Uma detalhada revisão sobre as licenças de software livre pode ser vista no capítulo [Capítulo Kon et al.].

O desenvolvimento de projetos como software livre se tornou amplamente reconhecido como uma forma eficaz de entregar software funcional. Apesar de iniciativas como a da RedHat (que financiam projetos de software livres há várias décadas) serem bem conhecidas, outras empresas, universidades e organizações públicas hoje financiam o

desenvolvimento de projetos de software livre. Assim, o desenvolvimento de projetos de software livre – que por muitos anos era considerado como uma atividade voluntária, em que desenvolvedores investem seu tempo livre para construir / projetar / testar / refatorar projetos de interesse – tem recentemente ganhado uma perspectiva mais empresarial. Como resultado, contribuidores de projetos de software livre agora são uma mistura de voluntários (que dedicam do seu tempo livre para contribuir) e empregados (que são contratados por empresas para contribuir).

À primeira vista, pode-se imaginar que migrar um projeto de software proprietário para um formato de software livre pode introduzir vários benefícios. Por exemplo, um projeto de software livre pode:

1. fomentar uma base de contribuidores externos;
2. gerar novas e inovadoras ideias; e
3. acelerar o passo de mudanças.

Um outro eixo importante que sustenta a pirâmide do software livre nos dias atuais se apoia em projetos de software livre que foram um dia projetos de software proprietário. Nos últimos anos, somente a Google já tornou mais de 900 projetos de software --- desenvolvidos inicialmente para atender demandas internas --- em formato de software livre, traduzidos em mais de 20 milhões de linhas de código fonte[4]. De forma similar, a Microsoft e a Apple já disponibilizaram como software livre alguns de seus produtos mais renomados. Uma rápida consulta no site de busca HackerNews --- um agregador de notícias para engenheiros de software --- reporta mais de 250 ocorrências de mensagens que indiquem a abertura de projetos proprietários para um formato de software livre[5].

Além do mais, existe a crença de que projetos de software livre populares tendem a atrair ainda mais contribuidores de código fonte, o que, por sua vez, pode aumentar a diversidade em um projeto de software, além de aprimorar as habilidades de resolução de problemas. Ademais, abrir um projeto de software proprietário pode ainda melhorar atributos de qualidade no projeto, como já vislumbrado pela "Lei de Linus", que diz que "Dados olhos suficientes, todos os erros são óbvios"; tal lei já foi empiricamente avaliada em outros estudos.

Os potenciais benefícios de tornar aberto um projeto proprietário, se alcançados, não vêm sem custo. Abrir o código fonte de um projeto de software proprietário também significa criar, manter e fomentar uma comunidade de desenvolvedores ao redor da tecnologia. Ademais, quando uma empresa de software decide abrir um de seus produtos, é provável que o time de desenvolvimento enfrente uma sobrecarga de trabalho devido aos custos de (1) refatorar o código fonte para torná-la mais compreensível; (2) configurar um sítio online, além de canais de comunicação para que novos desenvolvedores possam entrar em contato; além de (3) escrever documentação para apoiar novos desenvolvedores. Como

---

4 https://developers.google.com/open source/projects

5 https://hn.algolia.com/?query="is now open source", acessado em 23 de Junho de 2018

consequência, a decisão de abrir ou não um projeto de software proprietário deve ser tomada com cuidado.

3. Desenvolvimento Baseado em Comunidade

Muitos eram descrédulos que projetos de software livre seriam bem-sucedidos. Como projetos tocados de forma distribuída com trabalho voluntário e sem uma estrutura formal de comando e controle iriam para frente? Projetos como Linux mostraram que não só esse modelo é viável, mas que muitas vezes gera projetos mais bem-sucedidos do que outros conduzidos como software proprietário em um ambiente corporativo.

Como projetos de software livre surgem? Normalmente, um projeto nasce da necessidade de um desenvolvedor ou empresa. Alguém não encontrou um bom software que resolvesse seu problema e resolveu criar um. Como diz o ditado popular, a necessidade é a mãe da invenção. Daí, por que não disponibilizar de forma livre o software para a comunidade? Abrir o código como software livre possibilita que outros desenvolvedores que tenham o mesmo problema adotem o software. Eventualmente, esses desenvolvedores vão sentir necessidade de estender o software original para suas necessidades ou de corrigir problemas que os estejam atrapalhando. Tendo o código aberto, esses desenvolvedores podem modificar o software e contribuir de volta para o projeto. Aqui podemos observar dois fenômenos comuns em software livre: "*scratch your own itch*" (coçar sua própria coceira) e "*give back*" (retribuir).

Se o projeto atrair muitos interessados, forma-se uma comunidade de desenvolvimento. Seguindo a filosofia do software livre, tende-se a adotar um modelo de desenvolvimento aberto. Forma-se um time central de mantenedores que são responsáveis por revisar e aceitar as contribuições de desenvolvedores externos. Projetos de software livre são normalmente meritocráticos, então esse time é dinâmico - desenvolvedores externos muito ativos são convidados a entrar no time, enquanto alguns membros deixam de contribuir. No modelo de desenvolvimento aberto, toda a discussão sobre os rumos do projeto, as solicitações de mudança, a lista de bugs, os comentários feitos nas revisões das contribuições, entre outros, são disponíveis publicamente. Isso possibilita que desenvolvedores externos aprendam a dinâmica da comunidade e participem ativamente de qualquer um de seus processos. Além da meritocracia, essas comunidades são caracterizadas por princípios de igualdade e auto-organização.

E por que as pessoas participam de forma voluntária dessas comunidades? Além de coçar a própria coceira, implementando funcionalidades ou corrigindo bugs de interesse, estudos mostram que muitos participam para aprender. A disponibilidade de código fonte de sistemas reais junto com uma comunidade que dá suporte ao desenvolvimento é uma excelente oportunidade para aprendizado. Além do aprendizado, a participação em software livre deixa registros públicos, que são um grande diferencial na busca por emprego. Portanto, melhorar o currículo também é um grande atrativo para o software livre. Empresas valorizam essa experiência e de fato analisam o perfil do desenvolvedor em plataformas abertas, como o GitHub, onde atualmente está sediada a maior parte dos projetos de software livre ativos.

4. Contribuindo para um projeto de software livre

Existem diferentes formas de contribuir para projetos de software livre. O simples fato de enviar relatos de bugs encontrados, oferecendo detalhes e meios de reproduzir o problema, é uma maneira muito bem-vinda de contribuir. Escrever documentação, traduzir a interface gráfica, promover a comunidade e até responder questões de usuários em fóruns e listas de e-mails são maneiras relevantes de fazer parte de uma comunidade de software livre. Entretanto, quando pensamos em contribuir para um projeto de software livre tendo conhecimento técnico em desenvolvimento de software, o que vem à cabeça é contribuir com código fonte. Essa é a forma mais comum e que mais atrai contribuidores às comunidades de software livre.

Nesta seção o foco vai ser sobre esse tipo de contribuição: a contribuição com código fonte. Primeiramente, vamos explorar o fluxo mais comum quando se deseja contribuir para um projeto de software livre. Na Figura 1, buscamos generalizar as etapas de uma contribuição. Entretanto, como repetiremos algumas vezes, projetos são singulares, e cada um tem suas especificidades. A intenção de traduzir o processo de contribuição para um fluxo é facilitar o entendimento do que esperar durante a contribuição.

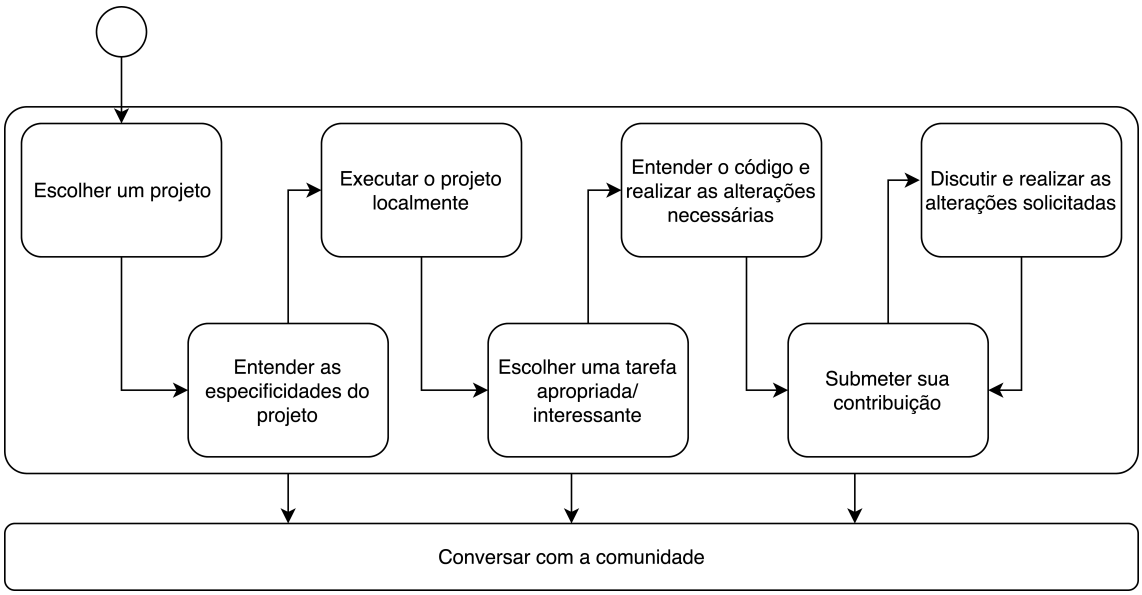

Figura 1. Processo de Contribuição para um Projeto de Software Livre

O primeiro passo é escolher o projeto. Uma dica interessante é buscar por um projeto que seja familiar. Pense em algo que você usa, que seja popular e que lhe dê prazer. Outro ponto é buscar por algo que seja desenvolvido em linguagens e tecnologias que você conheça e que não causem dificuldades em um primeiro momento.

Ao encontrar um projeto que interesse (ou um conjunto), o próximo passo é entender o projeto. Tenha em mente que toda comunidade de software livre é diferente. Pesquise a página do projeto para encontrar o repositório do código-fonte, o *issue tracker* (onde os usuários relatam e discutem tarefas e bugs) e os meios de comunicação disponíveis (listas

de discussão, fóruns, canais de IRC). Outra possibilidade para encontrar informações sobre o projeto é consultar o OpenHub[6], que é uma comunidade online e um diretório público de projetos de software livre.

Essas são informações básicas que você usará durante a sua jornada. Busque entender a forma como a comunidade interage, padrões de código utilizados, ferramentas preferidas, protocolos e habilidades necessárias. Alguns projetos oferecem diretrizes para novos contribuidores. Por exemplo, projetos hospedados em ambientes sociais de codificação (GitHub, GitLab, BitBucket, etc.) costumam manter um arquivo chamado CONTRIBUTING.md para orientar os desenvolvedores. Tendo mapeado o projeto, prepare-se para iniciar a jornada para entrar no projeto.

O próximo passo é **executar o projeto** localmente. Essa atividade envolve, em geral, baixar o código do repositório, instalar ou definir dependências necessárias e compilar/executar o projeto. É importante ressaltar que, apesar de parecer uma atividade simples, ela envolve diversos detalhes e pode ser cheia de obstáculos. Alguns projetos têm necessidades ou processos específicos. De forma geral, para executar o projeto é necessário realizar baixar uma cópia para sua máquina local (mais informações sobre o este modelo estão disponíveis online[7]). Além disso, projetos também podem definir ambientes de desenvolvimento (IDEs), plataformas ou sistemas operacionais específicos. Caso siga encontrando problemas após várias tentativas, *comunique-se com a comunidade*, identificando-se, detalhando suas tentativas e descrevendo (em detalhes) seu problema.

Para que sua jornada se torne interessante, é preciso que você consiga encontrar uma tarefa apropriada e interessante. Isso nem sempre é algo trivial e pode requerer algum tempo. A primeira etapa aqui é, basicamente, ir ao *issue tracker* e buscar por tarefas abertas. Alguns projetos seguem uma boa prática de etiquetar as tarefas fáceis ou apropriadas para novos contribuidores (ex. *good-for-beginner, first-timers-only, easy, newcomer, good-first-bug*). A leitura das diretrizes de contribuição dos projetos pode auxiliar a encontrar tais tarefas. Outra dica é buscar entender o que a tarefa está pedindo e analisar as habilidades que possam ser necessárias, para identificar se a tarefa é apropriada ao seu conjunto de habilidades e conhecimentos. Se tiver dúvidas, comunique-se com a comunidade. Em seguida, é importante tentar reproduzir o problema (caso seja um bug). Ao reproduzir, depure o código para entender o que está acontecendo. Por fim, nunca deixe de avisar a comunidade que você está trabalhando - comunicação é chave em projetos de software livre.

A partir desse momento, espera-se que você esteja preparado para começar a colocar a mão na massa. É necessário então **entender o código e realizar as alterações necessárias**. Utilize funcionalidades de busca de sua IDE ou do seu sistema operacional para apoiá-lo, bem como ferramentas de depuração. Para implementar suas alterações, lembre-se de seguir os padrões e boas práticas empregados pelo projeto. Se ainda não encontrou, volte a estudar o projeto e entender como isso deve acontecer – mais uma vez, se precisar, comunique-se com a comunidade pelos canais disponíveis.

---

6 https://www.openhub.net/

7 https://help.github.com/articles/fork-a-repo/

Com a solução implementada, chegou a hora de **submeter sua solução**. Antes de mais nada, volte a analisar as diretrizes de contribuição e verifique se existem itens que são necessários à sua submissão (inclusão de testes, modificação de *release notes*, aceitar o contrato de licença etc.). Então, verifique como é o processo de envio de suas alterações, que podem ser específicos para cada projeto. Mais uma vez, se o projeto está hospedado em plataformas sociais, como o GitHub, o processo deve ser o de *pull-request*[8], que consiste em enviar sua solução a partir de seu repositório privado para o repositório compartilhado do projeto. Este pode não ser o fim. Seu código será agora revisado (por ferramentas e revisores do projeto), e suas alterações podem ou não ser aceitas. Caso existam questionamentos com relação ao seu código, ou algum problema a ser corrigido, é chegada a hora de **discutir e realizar as alterações solicitadas**. Se não entender, pergunte. Se houver dúvidas com relação à sua solução, esclareça. Pode haver vários ciclos de revisão. Não desista. É o momento em que a comunidade interage com você para refinar a contribuição. Você vai aprender bastante e o projeto receberá um código de melhor qualidade.

Cuidado! Barreiras à frente

Como projetos de software livre são, geralmente, um esforço coletivo de uma comunidade de desenvolvedores, contribuir para um projeto de software livre é muito mais do que apenas lidar com os desafios técnicos. A contribuição envolve também aspectos não-técnicos, como habilidades sociais e aprender técnicas e ferramentas relacionadas ao processo de contribuição.

Em pesquisas recentes (STEINMACHER et al., 2015; BALALI et al., 2018), diversos desses desafios, ou barreiras, são identificadas e detalhadas. Como dito anteriormente, elas podem ter origem nos diversos aspectos do projeto, podendo ser classificadas como:

- **Barreiras pessoais**: relacionadas às características pessoais dos contribuidores (ou dos membros do projeto). Incluem timidez, medo de julgamento, falta de proatividade, falta de interesse.
- **Barreiras interpessoais**: dizem respeito ao relacionamento social entre indivíduos da comunidade e novos desenvolvedores, incluindo falta de habilidades de comunicação (línguas), baixa taxa de resposta aos novos desenvolvedores, mensagens ríspidas, problemas com diferentes culturas etc.
- **Barreiras técnicas**: problemas relacionados diretamente ou causados pela tecnologia (linguagem de programação, arcabouços e ferramentas utilizados pelo projeto). São exemplos de barreiras desse tipo: dificuldade de entender o código, problemas para configurar o ambiente de execução local, complexidade arquitetural etc.
- **Barreiras de processo**: são aquelas impostas pela organização, por procedimentos ou práticas internas, podendo incluir dificuldade para identificação de tarefas apropriadas para novos desenvolvedores, tempo de revisão de código prolongado, dificuldade para entender padrões de código etc.

Ao se propor a contribuir, esteja ciente de que é possível que você enfrente algumas dessas barreiras. Esteja preparado, e seja persistente. Software livre depende de contribuidores novos e é um excelente modo de aprender como funcionam os processos de engenharia de software no mundo real.

[8] https://help.github.com/articles/creating-a-pull-request/

5. Benefícios para a Sociedade

Importante mencionar que o software livre vai além de ser uma forma de licenciar software. A ideia de software livre está ligada à forma de compartilhar e utilizar conhecimento, formando uma comunidade em torno de um produto que pode ser consumido livremente. É comum pensarmos em software livre, de maneira simplista, como um software gratuito. Mas, como descrito no início deste capítulo, as liberdades que tornam um software livre vão além de sua gratuidade. A Figura 2 apresenta, em uma nuvem de palavras, alguns desses benefícios que serão aqui discutidos.

Figura 2. Software Livre e seus potenciais benefícios para a sociedade

O maior benefício vem das origens do movimento e está relacionado ao desenvolvimento do conhecimento, que depende do compartilhamento livre de ideias. É a possibilidade de "enxergar mais longe por estar nos ombros de gigantes". Pode-se resumir esse benefício como **democratização do conhecimento**, que pode ser visto como base para todos os outros benefícios do software livre.

Por ter o código disponível livremente, é possível aprender, ensinar, encontrar problemas, reusar soluções, otimizar uma solução, melhorar a qualidade etc. Como dito por Kon et al. (2012), "*restrições de acesso ao código fonte do software podem se traduzir em dificuldades para governos, empresas e cidadãos, já que o software traz consigo regras e decisões tomadas em função de interesses diversos*".

Do ponto de vista do desenvolvimento de software em si, o software livre possibilita a **reutilização de trechos de código**, ou componentes, em diferentes aplicações e contextos. Como colocado por Eric Raymond, "*Os bons programadores sabem o que escrever; os grandes sabem o que reescrever (e reusar)*". Do ponto de vista de custo, isso é interessante, já que diferentes empresas podem se beneficiar e, caso necessário, melhorar uma solução com base no que já está disponível. A adoção de software livre fomenta ainda a **competição entre fornecedores**, que podem utilizar-se das mesmas soluções para oferecer serviços aos

usuários finais. O que passa a ocorrer é a concorrência baseada nos serviços oferecidos, e nas garantias que as empresas entregam.

Outro potencial benefício do software livre para a sociedade tem relação à **adoção pelo governo**. Diferentemente do que se imagina num primeiro momento, custo é apenas um dos benefícios. Pensemos, por exemplo, em um sistema de automação de escritório (suíte de aplicativos *office*). Um governo pode optar por uma solução proprietária, de uma empresa multinacional, ou uma opção livre. Vamos explorar o que acontece caso opte pela alternativa proprietária. Provavelmente, o dinheiro das licenças estará empregando alguns desenvolvedores fora do país, que gastarão seus fundos no exterior. Além disso, o código fonte é uma caixa-preta, sendo conhecido apenas pela empresa que detém seus direitos. O governo não tem a **possibilidade de auditoria** e de **adequação para necessidades específicas** (personalização). Caso opte pela alternativa livre, o custo inicial é reduzido, mas não zerado (como alguns poderiam esperar). O governo ainda precisará arcar com custos de instalação e, provavelmente, de suporte. Entretanto, as opções de suporte podem ser locais, empregando mão-de-obra dentro do estado/município/país, gerando mercado, conhecimento, oportunidades e renda locais. Na possível necessidade de auditoria, personalização ou melhoria da solução, é possível fazê-los contando com o suporte local, negociando as necessidades específicas e tendo o conhecimento sobre o produto desenvolvido.

Ainda sobre solução para governo, a construção de aplicações em diferentes esferas pode criar uma comunidade interessada em aprimorar e compartilhar soluções. Com isso, o governo detém o conhecimento sobre a aplicação, podendo treinar mão-de-obra especializada e compartilhar a solução entre diferentes órgãos do governo. Alguns casos de sucesso existem no Brasil, como, por exemplo, o portal de correio eletrônico e colaboração Expresso Livre[9], que é a customização da ferramenta alemã E-GroupWare, realizada pela CELEPAR, empresa de desenvolvimento de software do Governo do Paraná. No momento da escrita deste livro, o portal estava sendo utilizado por mais de 30 órgãos de governo, em diferentes esferas (municipal, estadual e federal) da administração direta e indireta, nos três poderes. O governo federal do Brasil oferece um catálogo de sistemas de software em um portal[10] que atende às necessidades de diferentes esferas, em que diferentes órgãos podem se beneficiar, gerando economia e fomentando o ecossistema nacional.

Software livre também pode trazer benefícios para a **educação** em áreas relacionadas ao desenvolvimento de software. A disponibilidade de código fonte e artefatos de documentação de soluções que são amplamente adotadas no mercado possibilita que professores, estudantes e pessoas interessadas em aprender sobre desenvolvimento de software tenham acesso a exemplos reais. Além disso, projetos de software livre são ambientes de colaboração intensa, em que uma comunidade interage a fim de construir software. A participação nesse tipo de projeto possibilita interagir com sistemas reais, problemas reais e com uma equipe de desenvolvimento interessada em construir software de qualidade. Assim, tem-se a oportunidade de aprender habilidades, atitudes e os desafios inerentes ao desenvolvimento de software no mundo real, o que pode aumentar a confiança dos estudantes quando forem iniciar suas atividades profissionais, bem como prepará-los melhor para o mercado de trabalho. Ainda, contribuir para projetos de software livre é uma

---

9 http://www.expressolivre.org/

10 https://softwarepublico.gov.br/

maneira de criar um portfólio, que pode ser utilizado para comprovar habilidades e conhecimentos prévios ao participar de um recrutamento.

Ferramentas para apoiar a entrada de novatos em projetos

O site openhatch.org se dedicava a combinar possíveis colaboradores com comunidades de software livre, provendo mecanismos de busca de tarefas, indicando mentores e oferecendo ferramentas e missões para aprendizado de tecnologias específicas. A organização que manteve o site oferecia ainda treinamentos (*Open Source Comes to Campus*) em universidades. Em maio de 2017 a organização anunciou seu fechamento. Entretanto, o site continua no ar buscando por pessoas que o mantenham.

Como resultado do trabalho de doutorado de um dos autores deste capítulo, Igor Steinmacher propôs o FLOSScoach, um portal web que oferece informações de maneira organizada de forma a melhor orientar os novatos que desejam contribuir para um projeto de software livre. O portal é um gerenciador de conhecimento que tem como base teórica o modelo de barreiras proposto pelo próprio autor (STEINMACHER et al., 2016).

O portal up-for-grabs.net é uma das iniciativas que listam projetos de software livre que etiquetam tarefas indicadas para novatos em seus repositórios hospedados no GitHub. Essa iniciativa visa atrair novos contribuidores para os projetos, partindo do pressuposto de que a utilização de rótulos para novatos é uma prática com potencial para atingir tal objetivo. O portal apresenta, para cada um dos projetos cadastrados, as seguintes informações: nome, descrição do projeto, etiquetas que o associam a características como linguagem ou área de atuação, o rótulo oficial utilizado para designar uma tarefa indicada para novatos e a quantidade de tarefas para novatos pendentes em seu repositório.

O site http://opensource.guide apresenta uma coleção de recursos para potenciais contribuidores, comunidades e empresas que desejam aprender como executar e contribuir para um projeto de software livre. O objetivo desse site é agregar as melhores práticas da comunidade.

Resumo

Sistemas de software livre são regidos por quatro liberdades que permitem a execução, adaptação, redistribuição e modificação desses sistemas livremente. Esse modelo de desenvolvimento surgiu na década de 80 com Richard Stallman, enraizado em laboratórios de pesquisa do MIT. Atualmente, no entanto, sistemas software livre estão presentes no dia-a-dia das pessoas, mesmo que de forma transparente. É possível encontrar sistemas desenvolvidos nesse modelo não somente em computadores de mesa, mas também em celulares, aparelhos de televisão e em dispositivos embarcados. Ademais, sistemas de software livre constituem uma presença comum no canivete suíço de tecnologias que engenheiros de software dominam para realizar suas atividades. Tais sistemas são desenvolvidos colaborativamente, e tem um foco em comunidades (intrinsecamente distribuídas). Fazer parte de tais comunidades por trazer benefícios para os participantes e para a sociedade em geral. Por isso, muitos desenvolvedores se engajam como meio de aprendizado e reconhecimento, podendo se tornar um meio de subsistência. Neste capítulo foram apresentados conceitos relacionados a software livre, sua história e modo de desenvolvimento. Buscou-se apresentar uma perspectiva sobre como o modelo se desenvolveu durante a história até chegar nos dias atuais. Além disso, discutiram-se potenciais benefícios que o modelo traz para a sociedade. Por fim, apresentou-se de maneira didática quais são os passos que interessado deve trilhar a fim de realizar contribuições de código para um projeto.

Leituras recomendadas

- **The Cathedral and the Bazaar** (RAYMOND, 1997). É um ensaio sobre os processos

de desenvolvimento de software tendo como base experiências do desenvolvimento do Linux e do Fetchmail. O artigo gira em torno da (agora chamada) Lei de Linus, que diz que "Dado um número suficiente de olhos, todos os *bugs* são triviais".

- **Producing Open Source Software**. (FOGEL, 2018). É um livro sobre o lado humano do desenvolvimento de código aberto. Descreve como os projetos de sucesso operam, as expectativas dos usuários e desenvolvedores e a cultura do software livre. O livro é lançado sob uma licença livre. Segundo o autor, não é necessária nenhuma experiência anterior com software livre, como desenvolvedor ou como usuário. O livro está disponível online, e pode ser encontrado em diversos idiomas. A versão em português está em andamento, e pode contar com sua ajuda.
- **Open Source Licensing** (ROSEN, 2005). Este livro é tido como um guia completo para as leis que regem software livre, para desenvolvedores, gerentes e advogados. É crucial entender como as licenças de software livre funcionam - e suas bases legais. O consultor jurídico da Open Source Initiative, Lawrence Rosen, apresenta um guia explicando as leis de propriedade intelectual que suportam o licenciamento de código aberto, examina cuidadosamente as principais licenças de hoje e ajuda você a fazer as melhores escolhas para o seu projeto ou organização.
- **Let me in: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects** (STEINMACHER et al., 2018). É um artigo apresentando diretrizes para novos desenvolvedores e mantenedores de comunidades para melhorar a entrada de novatos em projetos de software livre.

Lista de Atividades

1) Da perspectiva do cidadão, discuta como a aplicação de software livre pode beneficiar ou prejudicar a sociedade.
2) Que tal colocar a mão na massa? A melhor maneira de entender como uma comunidade online funciona é na prática.
    a. Para começar, escolha um tópico de seu interesse na Wikipedia[11] (algum tópico já existente) e contribua de forma significativa, adicionando informações relevantes. (Sugere-se criar uma conta para fazer a contribuição, pois isso reduz a possibilidade de reversão e possibilita acompanhar possíveis discussões)
    b. Siga sua alteração, tendo especial atenção a revisões e rejeições. Aproprie-se de sua submissão e tente mantê-la
    c. Descreva sua experiência traçando um paralelo entre o processo de contribuição para um projeto de software livre e o processo que você seguiu no Wikipedia. Quais atividades são similares/diferentes? Quais foram os problemas enfrentados?
3) Tente contribuir para algum projeto de software livre que você considere interessante. Observe o processo fornecido no capítulo, estude (e use) as ferramentas de apoio a novatos apresentadas, e tente se envolver.

---

11 http://www.wikipedia.org/

Referências

BALALI, Sogol; STEINMACHER, Igor; ANNAMALAI, Umayal; SARMA, Anita; GEROSA, Marco A. **Newcomers' Barriers... Is That All? An Analysis of Mentors' and Newcomers' Barriers in OSS Projects**. Computer Supported Cooperative Work, 2018.

BLACKDUCK 2016. **The Tenth Annual Future of Open Source Survey**. Disponível em: https://www.blackducksoftware.com/2016-future-of-open-source. Acessado em 20/06/2018.

KON, Fabio. **Sistemas de arquivos distribuídos.** Dissertação de mestrado, Instituto de Matemática e Estatística - Universidade de São Paulo, 1994.

KON, Fabio; LAGO, Nelson; MEIRELLES, Paulo; SABINO, Vanessa. **Software Livre e Propriedade Intelectual: Aspectos Jurídicos, Licenças e Modelos de Negócio**. Jornada de Atualização em Informática, Congresso da Sociedade Brasileira de Computação, 2012. Disponível em http://ccsl.ime.usp.br/files/slpi. pdf. Acessado em 10/05/2018.

FOGEL, Karl. **Producing Open Source Software**. O'Reilly. Disponível em http://producingoss.com, 2018.

JEWELL, Teresa. **Open to Everyone: How Open Source Communities Can Benefit from Diversity Without Disunity.** Open Source Business Resource, August 2010. DIsponível em https://timreview.ca/article/369

RAYMOND, Eric S. **The Cathedral and the Bazaar.** Disponível em http://catb.org/esr/writings/cathedral-bazaar, 1997-2009.

ROSEN, Lawrence. **Open Source Licensing: Software Freedom and Intellectual Property Law**. New Jersey: Prentice Hall, 2005.

PINTO, Gustavo; STEINMACHER, Igor ; DIAS, Luiz Felipe F.; GEROSA, Marco A. **On the challenges of open-sourcing proprietary software projects.** EMPIRICAL SOFTWARE ENGINEERING, v. online, p. 1-27, 2018.

STEINMACHER, Igor; CONTE, Tayana U.; REDMILES, David F.; GEROSA, Marco Aurélio. **Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects.** In: The 18th ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW 2015). ACM, 2015. p. 1-13.

STEINMACHER, Igor; CONTE, Tayana U.; TREUDE, Christoph; GEROSA, Marco A. **Overcoming open source project entry barriers with a portal for newcomers**. In: the 38th International Conference, 2016, Austin. Proceedings of the 38th International Conference on Software Engineering - ICSE '16. p. 273-284.

STEINMACHER, Igor; TREUDE, Christoph; GEROSA, Marco A. **Let me in: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects**. IEEE SOFTWARE, 2018.

Sobre os Autores

Igor Steinmacher. Doutor em Ciência da Computação pelo Instituto de Matemática e Estatística da Universidade de São Paulo (IME/USP), em 2015, é Professor Assistente da Northern Arizona University (NAU), EUA, afastado da Universidade Tecnológica Federal do Paraná (UTFPR) - Campus Campo Mourão. Foi membro da Comissão Especial de Sistemas Colaborativos (2015-16) e membro da Comissão de Educação (2015-17) da Sociedade Brasileira de Computação. Tem interesse em software livre, aspectos humanos e sociais da engenharia de software e sistemas colaborativos, tendo atuado com mineração de repositórios e pesquisas qualitativas relacionadas a esses tópicos. http://lattes.cnpq.br/5529725593221391

Igor Wiese. Doutor em Ciência da Computação pelo Instituto de Matemática e Estatística da Universidade de São Paulo (IME/USP), em 2016, é Professor Adjunto da Universidade Tecnológica Federal do Paraná (UTFPR) - Campus Campo Mourão. Tem interesse em software livre, aspectos humanos e sociais da engenharia de software, tendo atuado com mineração de repositórios e sistemas de recomendação aplicados à engenharia de software. http://lattes.cnpq.br/0447444423694007

Gustavo Pinto. Doutor em Ciência da Computação pelo Centro de Informática da Universidade Federal de Pernambuco (CIn/UFPE). É professor Adjunto da Universidade Federal do Pará (UFPA). Sua pesquisa se concentra nas interações entre pessoas e código,

abrangendo as áreas de engenharia de software e linguagens de programação. http://lattes.cnpq.br/1631238943341152

Marco A. Gerosa. Doutor em Informática pela PUC-Rio, é professor Associado da Northern Arizona University (NAU), EUA, e membro da pós-graduação do Depto. de Ciência da Computação do IME/USP. Sua pesquisa concentra-se na intersecção da Engenharia de Software e Sistemas Colaborativos, investigando aspectos técnicos e sociais de comunidades de software livre. Foi editor de periódicos científicos, chair de comitê de programa de eventos internacionais e nacionais e membro de diversos comitês de programa de eventos. http://lattes.cnpq.br/4507073071352893