Um Estudo sobre Sustentabilidade de Projetos de Software Livre

Flávia Linhalis Arantes[1]

[1]Núcleo de Informática Aplicada à Educação (NIED)
Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brazil

farantes@unicamp.br

Abstract. *Open source projects sustainability should be treated broadly, thinking not only in financial resources, but also in all the dynamics involved in the life cycle of a sustainable project. In this work, I mean by sustainable the management model that can support the project through various resources – not only funds of a single company or institution. Particular aspects related to Brasilian open source software sustainability are also discussed in this paper. In Brazil, most open source projects have arisen as results of research projects at universities or demanded by government institutions. The continuity of such projects depends greatly on research continuity or government funding. Our aim with this research is to contribute with open source projects sustainability, in general, but mainly in countries like Brazil, where projects have to survive with volunteers and reduced financial resources. In this paper a literature overview about open source software sustainability is presented, considering community growth, source code and tools management.*

Resumo. *A sustentabilidade de projetos de software livre deve ser tratada de maneira ampla, sem pensar apenas no recurso financeiro em si, mas em toda a dinâmica envolvida no ciclo de vida de um projeto sustentável. Entende-se aqui por sustentável, o modelo de gestão em que é possível sustentar o projeto por meio de diversos recursos, sem que ele seja mantido financeiramente por uma única empresa ou instituição. Alguns aspectos particulares relacionados à sustentabilidade de projetos de software livre no contexto brasileiro também são discutidos neste artigo. A maioria dos projetos de software livre genuinamente brasileiros surgiram como resultados de pesquisas em universidades ou demandados por instituições do governo. A continuidade desses projetos depende grandemente da continuidade das pesquisas ou de financiamento do governo. O objetivo do estudo apresentado neste artigo é contribuir com a sustentabilidade de projetos de software livre, em geral, mas principalmente com casos de países como o Brasil, onde os projetos precisam sobreviver com voluntários ou poucos recursos financeiros. Este artigo apresenta uma revisão da literatura na área de sustentabilidade de projetos de software livre considerando não apenas os recursos financeiros, mas também o crescimento da comunidade, o gerenciamento do código-fonte e das ferramentas de comunicação.*

1. Introdução

A classe de software denominada software livre ou *open source* é uma classe de produtos de software distribuídos com direitos e restrições especiais, e frequentemente desenvolvi-

dos em projetos voluntários. Uma definição objetiva e amplamente aceita é a adotada pela OSI (*Open Source Initiative*): Software Livre é qualquer software cuja licença garanta ao seu usuário liberdades relacionadas ao uso, alteração e redistribuição. Seu aspecto fundamental é o fato do código-fonte estar livremente disponível para ser lido, estudado ou modificado por qualquer pessoa interessada [OSI 2011].

O termo software livre usualmente aparece em conjunto com outras características importantes: distribuição via Internet, desenvolvimento colaborativo e descentralizado, existência de uma comunidade de usuários, ferramentas de comunicação e desenvolvimento distribuídas, forte individualismo em contraste com hierarquias rígidas de empresas.

O que conhecemos como software livre hoje começou em meados dos anos oitenta, em resposta às necessidades de diversos desenvolvedores e usuários, que entendiam que o software é algo ativo que deve ser compartilhado livremente [Stallman 1985]. Esse movimento culminou no Projeto GNU[1] e na criação da *Free Software Foundation*.

Durante os anos noventa, o software livre experimentou um período de crescimento e conquistou seu espaço. Nos dias de hoje, pode-se dizer que ele adquiriu uma nova dimensão, diferente do puro idealismo e movimento filosófico da FSF [Lawton 2009, Riehle 2010]. Isto é, o software livre deixou de ser um movimento à parte, passou a ser economicamente relevante e hoje é integrante da indústria de software mundial.

Com a popularidade do software livre, grandes companhias como IBM, Netscape, Hewlett-Packard, dentre outras, passaram a incluir software livre em suas estratégias de mercado [Riehle 2010, Capek et al. 2005, Dinkelacker et al. 2002, Hecker 1999]. Governos estão usando software livre para reduzir custos e estimular o desenvolvimento de software local [Allen 2010, Pastrana and López 2009, Simon 2005]. Universidades e escolas ao redor do mundo estão adotando software livre como plataformas para educação [Wang et al. 2010, Mohamed et al. 2009, Barry 2009, Shockey and Cabrera 2005].

O motivo para governos e outras instituições estarem adotando software livre é primordialmente financeiro. É possível economizar milhões com o uso de plataformas livres [Allen 2010, Richter et al. 2009]. Uma pergunta natural que surge ao verificarmos o crescente uso desse tipo de software pelo qual não se paga nenhuma taxa ou licença é: como os projetos de software livre se sustentam?

Pesquisas relacionadas à sustentabilidade de comunidades de software livre [Hecker 1999, Lawton 2009, Riehle 2010] apontam que os lucros são provenientes da oferta de serviços como suporte, consultoria e treinamento, bem como da manutenção do produto ou ainda do desenvolvimento de customizações ou de novos softwares feitos sob encomenda. Entende-se aqui por sustentável, o modelo de gestão em que é possível sustentar o projeto, por meio de diversos recursos, sem que seja mantido financeiramente por uma única empresa ou instituição.

Neste artigo, a sustentabilidade de projetos de software livre é tratada de maneira ampla, sem pensar apenas no recurso financeiro em si, mas em toda a dinâmica envolvida no ciclo de vida de um projeto de software livre sustentável. Com base em uma revisão

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

da literatura foi possível concluir que a sustentabilidade de um software livre está intimamente relacionada a três fatores – crescimento da comunidade, recursos financeiros e gerenciamento de software.

Em um trabalho prévio, Arantes e Freire (2011) apresentam um ciclo de vida sustentável para projetos de software livre, o qual considera os fatores citados acima – crescimento da comunidade, recursos financeiros e gerenciamento de software – como base para a sustentabilidade de projetos de software livre em geral. A contribuição do presente artigo é apresentar uma versão estendida e aprimorada do trabalho de Arantes e Freire (2011). Além disso, este artigo inclui alguns aspectos e fragilidades relacionados à sustentabilidade de projetos genuinamente brasileiros.

Este artigo está organizado da seguinte maneira: a seção 2 retoma o ciclo de vida sustentável e os aspectos relacionados à sustentabilidade de projetos de software livre. A seção 3 discute alguns aspectos de sustentabilidade no contexto brasileiro. A seção 4 conclui o artigo e, finalmente, a seção 5 aponta direcionamentos futuros.

2. Aspectos relacionados à sustentabilidade de projetos de software livre

A maioria dos softwares livres começa com um desenvolvedor ou um grupo pequeno, que, buscando resolver um problema particular, intelectual ou de negócio, desenvolve um software deixando-o posteriormente disponível para outras pessoas utilizarem. Juntando-se ao idealizador ou desenvolvedor do software, os usuários e outros colaboradores acabam criando uma espécie de comunidade colaborativa em torno do software. Ao dar aos usuários o acesso ao software e seu código-fonte, acaba-se incentivando voluntários a evoluírem o mesmo [Hippel and Krogh 2003]. Nesse ponto, um bom gerenciamento se faz necessário para que o projeto e a comunidade de software livre possam continuar a crescer.

Arantes e Freire (2011) apresentam um ciclo de vida sustentável para projetos de software livre, o qual considera fatores relacionados ao crescimento da comunidade, aos recursos financeiros e ao gerenciamento do software.

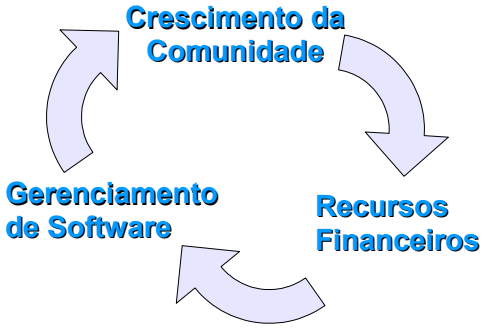

Figure 1. Ciclo de vida sustentável para projetos de software livre [Arantes and Freire 2011].

De acordo com a Figura 1, o crescimento e a continuidade da comunidade podem resultar mais naturalmente em recursos financeiros que, se bem gerenciados, podem ser

revertidos em benefícios para a comunidade e estimular seu crescimento, alimentando o ciclo de vida sustentável do software livre.

No restante desta seção são apresentados aspectos de sustentabilidade relacionados aos três fatores identificados no ciclo de vida da Figura 1.

2.1. Crescimento da Comunidade

Entender a dinâmica da organização e do gerenciamento de uma comunidade de software livre não é simples. Pelo contrário, em muitos casos é um cenário que, inicialmente, passa a ideia de "caos organizacional". Não há uma fronteira nítida entre usuários e desenvolvedores, pois uma mesma pessoa pode ser um ou outro em diferentes momentos. Todos os usuários podem dar opinião no desenvolvimento de uma solução, não apenas os responsáveis diretos pelo código. Esse método descentralizado de desenvolvimento foi cunhado por Eric Raymond de estilo "bazar", em contraste ao estilo "catedral" utilizado para descrever organizações de software comerciais tradicionais [Raymond 1999]. Enquanto em organizações tradicionais, o sistema de software é altamente centrado em algumas pessoas, que possuem postos de trabalho fixos em companhias de desenvolvimento de software; no estilo bazar, as pessoas estão distribuídas em diversas localidades, existe uma atmosfera de democracia, com uma variedade de vozes e agendas influenciando no desenvolvimento do software.

Alguns fatores são importantes para a existência e continuidade de uma comunidade de software livre: interatividade, variedade de participantes, número grande de membros, espaço na Internet que possibilite a interatividade, assuntos de interesse comum e colaboração entre os participantes. Uma premissa básica é que as pessoas devem trazer informações e compartilhá-las abertamente com o grupo [Vincentin 2007]. Em resumo, o sucesso de uma comunidade e até mesmo a confiabilidade do software estão relacionadas à comunicação e à colaboração entre seus participantes [Joode and Bruijne 2006, Stamelos et al. 2002]. Outro atributo importante de uma comunidade é o tamanho da população. Uma comunidade torna-se mais valiosa quanto mais integrantes juntarem-se a ela.

Como pode ser observado, o contato entre os participantes de uma comunidade é um fator de grande importância para sua continuidade. Sabe-se que a interação e a cooperação entre os membros contribui para uma visão de sólido desenvolvimento computacional para a comunidade, aumentando assim a confiabilidade do software.

A pesquisa realizada por Vincentin (2007) mostra que a continuidade de uma comunidade de software livre depende também da adoção do software por ela produzido. Isso irá aumentar a visibilidade da comunidade e, consequentemente, gerar oportunidades de trabalho em serviços agregados (suportes, melhorias, *upgrades*) que podem contribuir com a sustentabilidade do projeto.

Pesquisas ainda identificaram vários fatores para que empresas ou usuários tomem a decisão por adotar determinado software livre e, consequentemente, contribuir com a continuidade e crescimento de sua comunidade. Esses fatores incluem disponibilidade de suporte, tamanho da comunidade e atributos técnicos do software como confiabilidade, segurança, qualidade e performance [Morgan and Finnegan 2007, Dedrick and West 2004].

Apesar da aparente falta de mecanismos tradicionais de coordenação, tais como planos, processos definidos e documentação de projeto, a qualidade de muitos sistemas de software livre pode ser comparável aos seus similares comerciais [Stamelos et al. 2002, Lakhani and Wolf 2003]. De acordo com Joode e Bruijne (2006), a abertura da comunidade e do processo de desenvolvimento está relacionada com a confiabilidade e qualidade do software. Essa afirmação parte do princípio de que quanto mais pessoas estiverem prestando atenção no software e interagindo entre si, mais fácil será localizar e resolver seus defeitos. O seguinte axioma Linux ilustra perfeitamente essa questão: "*with enough eyeballs, all bugs are shallow*." [Zymaris 2003][2].

O trabalho de Santos e colegas apresenta um modelo para examinar o que atrai colaboradores para um projeto de software livre, contribuindo para a criação de uma comunidade (ou ambiente de trabalho) que pode melhorar e promover o projeto de forma sustentável. Os autores argumentam que a atratividade de um projeto de software livre está relacionada a um processo orientado ao usuário, envolvendo contribuições complementares de vários grupos [Santos et al. 2013].

2.2. Recursos Financeiros

De acordo com Lawton (2009), o movimento social que existia em torno do software livre na década de 80 tem dado lugar também a estratégias comerciais. Algumas pesquisas relacionadas à sustentabilidade de comunidades de software livre [Hecker 1999, Lawton 2009, Riehle 2010] apontam que, na maioria dos casos, os lucros são provenientes da oferta de serviços como suporte, consultoria e treinamento, bem como manutenção do produto ou ainda no desenvolvimento de customizações ou de novos softwares feitos sob encomenda. Há ainda empresas que distribuem livros ou outros artigos associados com algum suporte técnico aos softwares livres que compõem um produto. Um bom exemplo é a companhia O'Reilly & Associados, que publica livros que documentam e explicam vários produtos de software livre (Linux, Perl, GNU Emacs, etc.). Os softwares livres podem ainda ser financiados por vários segmentos da sociedade, como governo, meios acadêmicos e corporações [Capek et al. 2005, Capra et al. 2009].

Para receber mais apoio direto do governo, os grupos relacionados ao software livre muitas vezes transformam-se em ONGs (Organizações Não Governamentais), OSCIPs (Organizações da Sociedade Civil de Interesse Público), Fundações e até microempresas [Vincentin 2007, Riehle 2010]. Muitos aspectos econômicos relacionados a projetos de software livre têm sido investigados na literatura [Shirali-Shahreza and Shirali-Shahreza 2008]. Por exemplo, Riehle (2007) tenta responder uma das maiores questões no campo econômico relacionada a projetos de software livre: como o pagamento dos desenvolvedores é determinado? Outro exemplo é o trabalho de Lerner e Tirole [Lerner and Tirole 2002] que discute questões sobre a justificativa econômica dos projetos de software livre.

O aspecto econômico é de grande importância para a sustentabilidade de projetos de software livre. Uma vez que uma comunidade de software livre consegue angariar recursos financeiros, é preciso que haja uma dinâmica eficiente para o gerenciamento desses recursos e da própria comunidade, para que a mesma continue a crescer e a angariar mais recursos. Nas próximas seções são apresentados importantes aspectos de gerenciamento,

[2] "Com olhos suficientes, todos os bugs são superficiais".

principalmente relacionados ao código-fonte e às ferramentas de suporte à comunicação e à sincronização do trabalho, o que pode contribuir com a redução de custos e melhorar a sustentabilidade dos projetos.

2.3. Gerenciamento de software

Um projeto de software livre representa um conjunto formado por software, comunidade e ferramentas, os quais formam a base para comunicação e trabalho colaborativo, possibilitando que a comunidade possa expressar suas ideias e opiniões. Para coordenar o trabalho que realizam, os membros da comunidade utilizam a Internet através de ferramentas simples e amplamente disponíveis. Há duas categorias de ferramentas utilizadas em projetos de software livre. A primeira delas diz respeito ao gerenciamento do código-fonte; e a segunda está relacionada à comunicação entre os membros da comunidade e às ferramentas utilizadas pela equipe de desenvolvimento para comunicação e sincronização do trabalho. As subseções a seguir abortam esses dois aspectos separadamente.

2.3.1. Código-fonte

Por definição, o código-fonte que um projeto de software livre produz é distribuído através de uma licença livre. Esta base de código é frequentemente mantida em um repositório público de onde qualquer pessoa pode copiá-la. Alterações são normalmente feitas por um grupo pequeno de pessoas – em muitos casos, por uma pessoa só.

Normalmente o desenvolvimento do código-fonte é feito de forma isolada; cada desenvolvedor cria alterações localmente. Uma vez decidido que a alteração é correta e desejável, o processo de integração é iniciado. Durante esse processo, é comum que diversas pessoas comentem sobre a alterações, e que esta seja revisada e mesmo reescrita caso necessário [Reis 2003].

Embora a maioria do desenvolvimento seja feito *off-line* e individualmente, o desenvolvimento de software livre é um processo de trabalho em equipe que precisa de comunicação [Asklund and Bendix 2002]. É importante que os envolvidos no desenvolvimento do código-fonte possam discutir o seu trabalho e compartilhar seu conhecimento com outras pessoas. A comunicação e a integração da equipe de desenvolvimento pode melhorar a sustentabilidade do projeto de software livre, pois o aumento do compartilhamento de informação e cooperação podem contribuir para a solução de problemas comuns ao código-fonte. Além disso, a cooperação entre a equipe pode contribuir para que a mesma proponha soluções para melhorar questões como estrutura [Terceiro et al. 2010], modularização [Baldwin and Clark 2006, Sethanandha et al. 2010], dentre outras que são importantes para diminuir a complexidade estrutural do código-fonte e facilitar alterações.

Baldwin e Clark (2006) argumentam que o código com estrutura mais modular é um incentivo para desenvolvedores fazerem parte e continuarem a participar no desenvolvimento. Um sistema é dito ter modularidade se suas partes podem ser projetadas de forma independente, mas ainda assim trabalhar em conjunto para apoiar o todo. A modularidade cria oportunidades para o intercâmbio de valioso trabalho entre os desenvolvedores. Terceiro e colegas (2010) afirmam que a complexidade estrutural aumenta o custo de manutenção de um projeto de software, pois o código se torna mais difícil de entender, e em consequência mais difícil de modificar. Em projetos de software livre, esse

aumento do esforço pode representar uma dificuldade adicional para a entrada de novos participantes. A falta de atrativos para colaboradores representa uma ameaça para a sustentabilidade do projeto, especialmente aqueles que não são financiadas por uma única organização e contam com a contribuição de voluntários.

A estrutura e a modularização do código não são os únicos fatores que podem contribuir com a sustentabilidade dos projetos de software livre com relação ao código-fonte. Sethanandha e colegas (2010) discutem o gerenciamento de *patches* como um elemento de sustentabilidade importante, pois facilita o empacotamento das contribuições por parte de desenvolvedores distribuídos.

Outro aspecto relevante na consolidação de novas versões é a participação de membros da comunidade com diferentes *backgrounds*, não apenas desenvolvedores. Alguns projetos de software livre podem possuir uma gama heterogênea de usuários, com necessidades e pontos de vista diferentes. Nesse sentido, validar mudanças no código e na interface com usuários finais é importante.

2.3.2. Ferramentas para comunicação

Projetos em que os desenvolvedores estão distribuídos geograficamente são considerados mais difíceis de gerenciar. Outro problema de muitos projetos de software livre é a descontinuidade do time de desenvolvedores. Para problemas como esses, a necessidade de se comunicar, interagir e socializar usando ferramentas de comunicação com suporte computacional é ainda maior.

Os membros da comunidade se comunicam normalmente através de páginas web, listas de discussão e fóruns. Alguns projetos utilizam adicionalmente repositórios públicos de defeitos e formas de comunicação em tempo real, como chats. Em geral, usuários participantes tendem a usar estes veículos para comunicar experiências, problemas e solicitações.

Desenvolvedores também participam ativamente das discussões, fazendo boa parte do suporte ao usuário, e argumentando com outros desenvolvedores questões de projeto e implementação. Em projetos de dimensão um pouco maior existem listas e canais separados para discussão relacionada ao desenvolvimento em si.

Com relação às ferramentas para gerenciamemtno do código-fonte, normalmente é usada uma ferramenta de controle de versões, como SVN, Mercurial e GitHub, para armazenar a base de código do projeto. Também são utilizadas ferramentas para desenvolvimento e visualização do código fonte, como Eclipse, ferramentas para acompanhamento de defeitos (ou alterações), como Bugzilla, e ferramentas para controle de tickets, como RedMine.

Algumas iniciativas oferecem hospedagem e ferramentas de gerenciamento gratuitas para projetos de software livre. Uma dessas iniciativas é o SourceForge que contribui com a sustentabilidade dos projetos oferecendo hospedagem e recursos para gestão, incluindo ferramentas de comunicação, controle de versões, fóruns, sistema de doação e Marketplace [SourceForge 2011].

3. Aspectos de sustentabilidade no contexto brasileiro

Em 2003, o governo brasileiro iniciou um movimento em favor do software livre. O objetivo era encorajar órgãos federais e estaduais a migrarem para plataformas livres [Richter et al. 2009]. Em dois anos, vários ministérios do governo e instituições governamentais como ITI (Instituto de Tecnologia da Informação), Embrapa, Dataprev, Serpro, etc, estavam rodando Linux e outros softwares livres, o que tornou o Brasil o maior usuário Linux da América Latina no setor público [Benson 2005, Coelho 2006].

Um argumento importante para a adoção de software livre por parte do governo brasileiro foi a redução de gastos com o pagamento de licenças. Outro forte argumento é que, com essa medida, o Brasil passaria a produzir o seu próprio software, o que proporciona uma independência tecnológica e a possibilidade de criar programas perfeitamente ajustados às necessidades locais. Esse argumento não apenas melhora a independência com elação aos fornecedores de software, mas também contribui para a geração de empregos no país [Richter et al. 2009].

No Brasil, é muito comum que os recursos financeiros dos projetos de software livre sejam provenientes de agências governamentais, institutos de pesquisa e universidades, muitas vezes através de projetos financiados por órgãos de fomento [Torres 2010]. Um exemplo disso é o projeto Ginga, que é um *middleware* de código aberto para TV digital. O Ginga atualmente é comercializado por uma empresa nacional, mas foi desenvolvido pela PUC-Rio e UFPB, com apoio financeiro de órgãos de fomento como CAPES, CNPq, FINEP e MCT [SBC 2010]. Outro exemplo é o TelEduc, primeiro software brasileiro na área de Educação a Distância. O TelEduc foi desenvolvido pelo Núcleo de Informática Aplicada à Educação (NIED) e pelo Instituto de Computação (IC), da Unicamp e atualmente é mantido com recursos da universidade. Assim como o Ginga, o TelEduc foi desenvolvido em uma universidade, apoiado por órgãos de fomento. O software CACIC (Configurador Automático e Coletor de Informações Computacionais) é o primeiro Software Público Brasileiro (SPB), resultado do Consórcio de Cooperação entre a SLTI (Secretaria de Logística Tecnologia da Informação) e a Dataprev (Empresa de Tecnologia e Informações da Previdência Social). O CACIC é capaz de fornecer um diagnóstico preciso do parque computacional de uma empresa ou instituição e atualmente é mantido pelo governo, assim como vários outros softwares listados no portal do SPB[3].

Os projetos de software livre financiados por apenas uma instituição correm o risco de acabar, caso a instituição provedora não tenha recursos suficientes ou mude sua política de financiamento de projetos. Essa verdade se aplica principalmente em projetos de software livre que surgiram como resultados de projetos de pesquisa em universidades. Essa fragilidade dos projetos de software livre está presente em nosso país, onde a maioria dos softwares são financiados pelo governo ou pelas universidades.

Outra característica dos projetos de software livre nacionais é a não existência de comunidades de apoio bem solidificadas, como acontece em projetos de outros países. Não é cultura em nosso país contribuir financeiramente com projetos de software livre, o que seria importante para sustentação dos mesmos, tornando-os menos dependentes de sua instituição de origem.

[3]http://www.softwarepublico.gov.br/

4. Conclusões

Neste artigo, a sustentabilidade de projetos de software livre foi associada a três fatores – crescimento da comunidade, recursos financeiros e gerenciamento do software. O crescimento e continuidade da comunidade é um fator que leva em consideração a interatividade e a cooperação entre os participantes, bem como a visibilidade, a adequação ao mercado e a confiabilidade do software. Uma consequência da existência desses atributos é o surgimento de possibilidades financeiras advindas da adoção do software livre. Tais recursos financeiros precisam ser adequadamente gerenciados considerando recursos humanos, infra-estrutura de hardware e software. O bom gerenciamento dos recursos financeiros pode trazer benefícios para a comunidade e incentivar ainda mais o seu crescimento, alimentando o ciclo de vida sustentável do software livre.

O estudo apresentado neste artigo não levou em consideração particularidades de projetos de software livre, o que é muito comum quando se pensa em comunidades, onde cada uma tem sua autonomia e possui sua própria maneira de gerir recursos. Portanto, este estudo não pode ser considerado conclusivo ou definitivo.

A principal contribuição deste artigo foi mostrar uma visão geral, relacionando diferentes fatores que contribuem para a sustentabilidade de projetos de software livre. Este estudo pode servir como base conceitual para a criação e manutenção de projetos de software livre sustentáveis, em geral, bem como para projetos que contam com poucos recursos financeiros, como é o caso da maioria dos projetos de software livre brasileiros, onde muitos são dependentes de uma única instituição. Além disso, o estudo realizado pode contribuir para testes empíricos de verificação para comunidades já existentes.

Projetos de software livre que angariam poucos recursos financeiros precisam gerenciar com um certo cuidado aspectos relacionados ao código-fonte (como os apresentados na seção 2.3.1) e ferramentas de comunicação (seção 2.3.2). Um bom gerenciamento e modularização do código-fonte, por exemplo, podem facilitar a identificação e correção de bugs, bem como incentivar os desenvolvedores a entrarem no projeto como voluntários e a continuarem a participar no desenvolvimento, o que é fundamental em projetos com poucos recursos financeiros. Neste artigo, foi argumentado que um gerenciamento adequado da infra-estrutura de software e de recursos humanos pode reduzir os recursos financeiros necessários para manter o projeto e contribuir com sua sustentabilidade.

5. Direções Futuras

Quando um software livre começa a ser desenvolvido, muitas vezes não existe preocupação em divulgá-lo ao público alvo, principalmente no caso de softwares livres resultantes de projetos de pesquisa. Nesses casos, é comum o software ser desenvolvido sem uma comunidade de usuários e desenvolvedores, o que dificulta sua continuidade depois que o projeto de pesquisa termina.

Um desdobramento da pesquisa apresentada neste artigo trata da identificação de aspectos de sustentabilidade específicos para softwares livres originários de resultados de pesquisas científicas. Mais especificamente, pretende-se investigar assuntos relacionados aos seguintes tópicos e aos seus desdobramentos:

- Aspectos fundamentais de sustentabilidade – crescimento da comunidade, recursos financeiros e gerenciamento do software – aplicados aos casos particulares de

projetos de software livre que têm sua sede nas universidades.

- Características que diferenciam os projetos de software livre sediados em universidades.
- Criação de uma comunidade sustentável em torno de projetos de software livre já existentes.
- Casos de sucesso de projetos de software livre que foram criados a partir de projetos de pesquisa em universidades e podem ser considerados, atualmente, sustentáveis.

References

Allen, J. P. (2010). Open source deployment at the city and county of san francisco: From cost reduction to rapid innovation. *Hawaii International Conference on System Sciences*, pages 1–10.

Arantes, F. L. and Freire, F. M. P. (2011). Aspects of an open source software sustainable life cycle. In *7th International Conference on Open Source Systems (OSS 2011). Hissam et al (Eds)*, pages 325–329, Salvador, Brazil. IFIP AICT 365.

Asklund, U. and Bendix, L. (2002). A study of configuration management in open source software projects. In *IEE Proceedings – Software*, pages 40–46.

Baldwin, C. Y. and Clark, K. B. (2006). The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Manage. Sci.*, 52:1116–1127.

Barry, B. I. A. (2009). Using open source software in education in developing countries: The sudan example. In *International Conference on Computational Intelligence and Software Engineering (CiSE 2009)*, pages 1–4.

Benson, T. (2005). Brazil – free software's biggest and best friend. New York Times. Available online: http://www.nytimes.com/2005/03/29/technology/29computer.html.

Capek, P. G., Frank, S. P., Gerdt, S., and Shields, D. (2005). A history of ibm's open-source involvement and strategy. *IBM Syst. J.*, 44:249–257.

Capra, E., Francalanci, C., Merlo, F., and Rossi Lamastra, C. (2009). A survey on firms participation in open source community projects. In Boldyreff, C., Crowston, K., Lundell, B., and Wasserman, A., editors, *Open Source Ecosystems: Diverse Communities Interacting*, volume 299 of *IFIP Advances in Information and Communication Technology*, pages 225–236. Springer Boston.

Coelho, M. A. (2006). Segundo workshop regional latinoamericano – floss world. Available online: http://www.flossworld.org/conf2/presentations/ATT01171.pdf.

Dedrick, J. and West, J. (2004). An exploratory study into open source platform adoption. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 8 - Volume 8*, pages 1–10, Washington, DC, USA. IEEE Computer Society.

Dinkelacker, J., Garg, P. K., Miller, R., and Nelson, D. (2002). Progressive open source. In *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, pages 177–184, New York, NY, USA. ACM.

Hecker, F. (1999). Setting up shop: The business of open-source software. *IEEE Softw.*, 16:45–51.

Hippel, E. v. and Krogh, G. v. (2003). Open source software and the "private-collective" innovation model: Issues for organization science. *Organization Science*, 14:209–223.

Joode, R. W. and Bruijne, M. (2006). The organization of open source communities: Towards a framework to analyze the relationship between openness and reliability. *Hawaii International Conference on System Sciences*, 6:118b.

Lakhani, K. and Wolf, R. G. (2003). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. *Social Science Research Network*, pages 1–27.

Lawton, G. (2009). The changing face of open source. *Computer*, 42:14–17.

Lerner, J. and Tirole, J. (2002). Some simple economics of open source. *The Journal of Industrial Economics*, 50(2):197–234.

Mohamed, N., Seman, M. S. A., and Hussein, R. (2009). Open source software in information technology education. *Information Management and Engineering, International Conference on*, pages 99–102.

Morgan, L. and Finnegan, P. (2007). Benefits and drawbacks of open source software: An exploratory study of secondary software firms. In *Proceedings of the 3rd IFIP International Conference on Open Source Systems (OSS)*, pages 307–312. Springer.

OSI (2011). Open source initiative. Disponível online: www.opensource.org.

Pastrana, A. and López, E. S. (2009). Possibilities of open source software in developing local small business. In *Proceedings of the 2009 International Conference on Intelligent Networking and Collaborative Systems*, INCOS '09, pages 413–416, Washington, DC, USA. IEEE Computer Society.

Raymond, E. S. (1999). *The Cathedral and the Bazaar*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1st edition.

Reis, C. R. (2003). Caracterização de um processo de software para projetos de software livre. Master's thesis, Universidade de São Paulo.

Richter, D., Zo, H., and Maruschke, M. (2009). A comparative analysis of open source software usage in germany, brazil, and india. In *Proceedings of the 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, ICCIT '09, pages 1403–1410, Washington, DC, USA. IEEE Computer Society.

Riehle, D. (2010). The economic case for open source foundations. *Computer*, 43:86–90.

Santos, C., Kuk, G., Kon, F., and Pearson, J. (2013). The attraction of contributors in free and open source software projects. *The Journal of Strategic Information Systems*, 22(1):26 – 45. Service Management and Engineering in Information Systems Research.

SBC (2010). Integração que deu certo. Revista Computação Brasil. páginas 8 e 9.

Sethanandha, B. D., Massey, B., and Jones, W. (2010). Managing open source contributions for software project sustainability. *Technology Management for Global Economic Growth PICMET 2010 Proceedings of PICMET 10*, pages 1–9.

Shirali-Shahreza, S. and Shirali-Shahreza, M. (2008). Various aspects of open source software development. In *International Symposium on Information Technology. ITSim 2008*, volume 4, pages 1 –7.

Shockey, K. and Cabrera, P. J. (2005). Using open source to enhance learning. In *6th International Conference on Information Technology Based Higher Education and Training (ITHET 2005)*, pages F2A/7 – F2A12.

Simon, K. D. (2005). The value of open standards and open-source software in government environments. *IBM Syst. J.*, 44:227–238.

SourceForge (2011). Sourceforge – find and develop open source software. http://sourceforge.net/.

Stallman, R. (1985). The gnu manifesto. Disponível online: http://www.gnu.org/gnu/manifesto.html.

Stamelos, I., Angelis, L., Oikonomou, A., and Bleris, G. L. (2002). Code quality analysis in open source software development. *Inf. Syst. J.*, 12(1):43–60.

Terceiro, A., Rios, L. R., and Chavez, C. (2010). An empirical study on the structural complexity introduced by core and peripheral developers in free software projects. In *Proceedings of the 2010 Brazilian Symposium on Software Engineering*, SBES '10, pages 21–29, Washington, DC, USA. IEEE Computer Society.

Torres, G. G. (2010). Entrevista sobre esforço coordenado entre academia, indústria e governo. Revista Computação Brasil. páginas 4 e 5.

Vincentin, I. C. (2007). *Desenvolvimento de Software Livre no Brasil: estudo sobre a percepção dos envolvidos em relação às motivações ideológicas e de negócios.* PhD thesis, Faculdade de Economia, Administração e Contabilidade da Universidade de São Paulo.

Wang, H., Blue, J., and Plourde, M. (2010). Community source software in higher education. *IT Professional*, 12:31–37.

Zymaris, C. (2003). Linux security strong as ever. Linux article. Available at http://www.zdnet.com/news/linux-security-strong-as-ever/298545.