ATRATIVIDADE DE PROJETOS DE SOFTWARE LIVRE: IMPORTÂNCIA TEÓRICA E ESTRATÉGIAS PARA ADMINISTRAÇÃO

ATTRACTIVENESS OF FREE AND OPEN SOURCE SOFTWARE PROJECTS: THEORETICAL IMPORTANCE AND STRATEGIES FOR MANAGEMENT

ATRACTIVIDAD DE PROYECTOS DE SOFTWARE LIBRE: IMPORTANCIA TEÓRICA Y ESTRATEGIAS PARA ADMINISTRACIÓN

RESUMO

Milhares de Projetos de Software Livre (PSL) foram e continuam sendo criados na Internet. Esse cenário aumenta as oportunidades de colaboração tanto quanto acirra a concorrência por usuários e contribuidores, que elevariam esses projetos a níveis superiores aos que seriam alcançados por seus fundadores sozinhos. E dado que o aprimoramento por meio de colaboração é o principal objetivo dos fundadores de PSL, a importância de entender e administrar a capacidade de atrair usuários e contribuidores fica estabelecida. Para auxiliar pesquisadores e fundadores nesse desafio, o conceito de atratividade é introduzido neste artigo, que desenvolve um ferramental teórico-gerencial sobre as causas, indicadores e consequências da atratividade, viabilizando sua administração estratégica.

PALAVRAS-CHAVE Software livre, administração de projetos, projetos colaborativos, atratividade, comunidades virtuais.

Carlos Denner dos Santos Jr. denner@ime.usp.br
Pós-doutorando e Professor-colaborador do Departamento de Ciência da Computação, Universidade de São Paulo – São Paulo – SP, Brasil

Recebido em 15.07.2010. Aprovado em 14.10.2010
Avaliado pelo sistema *double blind review*
Editor Científico: Maria Alexandra Cunha

ABSTRACT *Thousands of Free and Open Source Software Projects (FSP) were, and continually are, created on the Internet. This scenario increases the number of opportunities to collaborate to the same extent that it promotes competition for users and contributors, who can guide projects to superior levels, unachievable by founders alone. Thus, given that the main goal of FSP founders is to improve their projects by means of collaboration, the importance to understand and manage the capacity of attracting users and contributors to the project is established. To support researchers and founders in this challenge, the concept of attractiveness is introduced in this paper, which develops a theoretical-managerial toolkit about the causes, indicators and consequences of attractiveness, enabling its strategic management.*

KEYWORDS *Free software, project management, open source, attractiveness, virtual communities.*

RESUMEN *Millares de Proyectos de Software Libre (PSL) fueron y continúan siendo creados en Internet. Ese escenario tanto aumenta las oportunidades de colaboración como instiga la competencia por usuarios y contribuidores, que elevarían esos proyectos a niveles superiores a los que serían alcanzados por sus fundadores solos. Y dado que el perfeccionamiento por medio de colaboración es el principal objetivo de los fundadores de PSL, la importancia de entender y administrar la capacidad de atraer usuarios y contribuidores queda establecida. Para auxiliar investigadores y fundadores en ese desafío, en este artículo se introduce el concepto de atractividad y se desarrollan herramientas teórico-gerenciales sobre las causas, indicadores y consecuencias de la atractividad, viabilizando su administración estratégica.*

PALABRAS CLAVE *Software libre, administración de proyectos, proyectos colaborativos, atractividad, comunidades virtuales.*

INTRODUÇÃO: CENÁRIO ATUAL

A forma de desenvolvimento de software livre tem sofrido mudanças substanciais com o crescente envolvimento de organizações com e sem finalidade lucrativa em suas atividades. Atuando nessas organizações, estrategistas públicos e privados têm aberto o código-fonte de suas aplicações à população em geral, tornando-as "livres", com o intuito de aprimorá-las e divulgá-las ao menor custo possível (FITZGERALD, 2006; SANTOS JR, 2008). Dentre as organizações que utilizaram essa estratégia, estão a IBM, com a plataforma de desenvolvimento Eclipse; a Mozilla, com o navegador Firefox; a Google, com o sistema operacional para celulares Android; e a Light Infocon, com o banco de dados LightBase.

Organizações públicas também são usuárias dessa estratégia. Entre elas, estão a CAIXA, que disponibilizou o sistema de emissão de bilhetes Curupira; a Prefeitura de Itajaí, com o sistema i-Educar de gestão escolar; e a NASA, com o JavaGenes, que implementa algoritmos genéticos. Adicionalmente, existem projetos com softwares comunitários por origem, onde indivíduos iniciam o desenvolvimento e depois compartilham o software para reunir esforços com outras pessoas na tarefa de aprimorá-lo, sem intervenções organizacionais iniciais. Iniciativas comunitárias são encontradas aos milhares na Internet. Exemplos desses esforços são o Azureus e o EMule, que desenvolvem aplicações para compartilhamento de arquivos entre indivíduos; o VLC, que desenvolve um tocador de vídeos concorrente do MediaPlayer da Microsoft; e o WINE, que desenvolve uma ferramenta para implementar APIs do Windows em ambientes Linux, contribuindo com a portabilidade de aplicações desenvolvidas exclusivamente para o sistema operacional da Microsoft.

Ainda, fora do contexto de desenvolvimento de software, algumas organizações estão engajadas em projetos que dependem de contribuições de usuários para obterem sucesso. Entre essas organizações, pode-se mencionar a Google, com suas diversas aplicações beneficiadas por contribuições de usuários (*e.g.,* fotos no Google Earth); a Nokia, com sua estratégia de inovação aberta, promovendo debates entre usuários sobre temas de seu próprio interesse; e a SAP, com suas comunidades de inovação, onde os problemas dos usuários podem ser resolvidos por outros usuários e a solução disponibilizada a todos (FARHOOMAND, 2007).

Como descrito, organizações produtoras de tecnologia da informação estão "abrindo as fronteiras" dos seus processos produtivos, transformando software proprietário em livre através da disponibilização do código-fonte e a criação de um projeto de software livre (PSL). Dessa forma, o estrategista habilita sua aplicação a receber contribuições externas, além de criar um novo canal de divulgação do software/organização. Mas, infelizmente, a simples criação de um PSL não garante que contribuições aconteçam, ou que o novo canal de divulgação seja eficaz, gerando um problema administrativo.

Adicionalmente, a diversidade e a quantidade desses projetos já são imensas e devem continuar a crescer, o que dificulta a tarefa dos estrategistas de entender o fenômeno e, assim, "concorrer" eficazmente pela limitada mão de obra voluntária disponível (O'MAHONY, 2007). No entanto, a favor dos estrategistas, sabe-se que, dentro dessa diversidade de PSL, há pelo menos uma similaridade teórica. Existe um conceito crucial que pode ser utilizado no estudo desses projetos, apoiando decisões gerenciais: é a atratividade do projeto; ou a sua capacidade de atrair a atenção de contribuidores e usuários potenciais (AGERFALK e FITZGERALD, 2008; SANTOS JR, 2009).

SOFTWARE LIVRE, PROJETOS E COMUNIDADES

Software livre é caracterizado como uma aplicação para computador com o código-fonte disponível para inspeção, alteração e utilização por qualquer pessoa, física ou jurídica (VON HIPPEL e VON KROGH, 2003). Frequentemente, para facilitar que contribuições sejam recebidas, juntamente com o código-fonte e a aplicação, encontram-se as ferramentas necessárias para o seu desenvolvimento e utilização (*e.g.,* documentação, definições de boa conduta dentro do projeto e software de suporte à comunicação). Com a disponibilização dessa estrutura em uma página na Internet, os projetos de software livre são criados. Os projetos ficam, então, habilitados a receber contribuições externas, mas sem garantia de que isso irá ocorrer – o que depende das características do projeto e da disponibilidade de contribuidores.

Com as criações de projetos de software livre e o aparecimento de voluntários-desenvolvedores, o tradicional problema administrativo de comprar software pronto ou desenvolvê-lo internamente (*make-or-buy*) ganhou outra opção. Ao criar um projeto de software livre, organizações fazem uso de um tipo especial de terceirização, dado que elas desconhecem os desenvolvedores e não os remuneram como em terceirizações regulares (AGERFALK e FITZGERALD, 2008). A abertura de código-fonte é híbrida, pois o desenvolvimento do software ocorre em dois momentos. Primeiro, ele faz parte da decisão de ser desenvolvido internamente (*make*) e, depois de aberto,

passa a ser parte de uma decisão de "compra" (*buy*) da mão de obra, que é obtida sem contratos. O sucesso dessa empreitada é altamente dependente da formação de uma comunidade ao redor do software e da dinâmica de trabalho dessa comunidade.

O PAPEL DA COMUNIDADE E SUAS CARACTERÍSTICAS

A diversidade dos projetos de software livre, dos membros das comunidades e das formas de governança adotadas por elas ainda extrapolam o conhecimento literário (O'MAHONY, 2007). Ao certo, sabe-se que a noção de um movimento *underground* composto por *hackers* para combater o monopólio no mercado de software e a promover a liberdade de acesso ao conhecimento representa o fenômeno parcialmente. Iniciativas corporativas, comunitárias, mistas e individuais, com ou sem desenvolvimento de software envolvido, continuam a aparecer, deixando o cenário complexo (SHAH, 2006).

Com isso, a utilização das expressões *software livre* e *open source* tornou-se tão vasta que sua capacidade descritiva perdeu força. Ao classificarmos uma comunidade como de software livre, são criadas mais dúvidas que esclarecimentos sobre seu funcionamento. Por exemplo: Quem são os membros? Quantos e quais desses membros são de fato contribuidores? (KUK, 2006). Como esses membros estão distribuídos geograficamente? (GREWAL e outros, 2006). Como o agrupamento se iniciou? (MacCORMACK e outros, 2006). Quais motivações os levam a aprimorar a aplicação? (ROBERTS e outros, 2006). Quem desenvolveu o software antes de ele se tornar público? Como a comunidade protege seu trabalho de ser apropriado em software proprietário? (O'MAHONY, 2003; SHAH, 2006). A versão "livre" do software é parte de uma estratégia maior que inclui versões proprietárias do mesmo software? (BONACCORSI e outros, 2006; SANTOS JR, 2008). Muitas dessas perguntas foram feitas, mas ainda não foram respondidas, requerendo tanto estudos de caso quanto estudos de larga escala.

Como dito, além da utilização das expressões para referir-se ao desenvolvimento de software, utiliza-se delas para descrever comunidades de outras áreas, desde que os produtos gerados coletivamente sejam disponibilizados publicamente. Pode-se encontrar referência a qualquer comunidade onde indivíduos adicionam seus trabalhos uns sobre os outros, formando ambientes colaborativos, como usuários do modelo *open source* (O'MAHONY, 2007). Assim, como a aplicação do modelo "livre" é ampla e suas fronteiras não estão bem definidas, pesquisas na área devem ser encorajadas.

Essas iniciativas "livres" têm alterado a dinâmica da competição em vários mercados por apresentar uma alternativa grátis com qualidade compatível (ECONOMIDES e KATSAMAKAS, 2006). Por exemplo, é possível ver como o Wikipedia.org influencia decisões de compra de enciclopédias, como *blogs* substituem revistas e como software de compartilhamento de arquivos facilita a pirataria, atingindo os mercados da música, dos livros e de jogos. Fortalecendo o movimento, organizações públicas em países como Brasil, Dinamarca, Japão e outros da União Europeia também se tornaram usuárias e financiadoras de software livre (O'MAHONY, 2007); além da Sun (Oracle) e Aethra (SANTOS JR. e GONÇALVES, 2006).

TAMANHOS E COMPOSIÇÃO DAS COMUNIDADES, DESEMPENHO E INOVAÇÃO

O reconhecimento de software livre como de qualidade comparável à dos seus concorrentes proprietários sempre foi justificado pelo trabalho dos voluntários que se empenharam no seu desenvolvimento. Essa justificativa tem origem nos documentos fundadores do movimento, que enfatizaram o papel das pessoas, especialmente da quantidade delas, na detecção de erros e requisição de novas funcionalidades, mantendo a aplicação competitiva. Por mais que uma comunidade possua um grupo central responsável pela maior parte das tarefas de desenvolvimento e definição das diretrizes de nível mais alto, a importância do tamanho da comunidade de "leitores" do código-fonte e de usuários que reportam erros e requisitam novas funcionalidades sempre foi enfatizada.

Uma frase popular que retrata essa ênfase é encontrada em Raymond (1999), em um dos documentos fundadores do movimento software livre. Raymond (1999, p. 27) escreveu "*given enough eyeballs, all bugs are shallow*", expondo a suposição de que a probabilidade de encontrar erros em um software aumenta à medida que a quantidade de pessoas que o lê cresce. Adicionalmente, a diversidade de um grupo também aumenta com o crescimento do seu tamanho. Segundo O'Mahony (2007) e West e Anderson (1996), um grupo mais diverso inova mais, desenvolvendo novas funcionalidades.

Essas ideias justificam as ações organizacionais de habilitar o acesso a processos produtivos, anteriormente restritos à comunidade de forma geral, aumentando a qualidade de produtos e serviços com base em contribuições externas. Comumente, essas iniciativas aparecem

na literatura como sendo do modelo "*open innovation*" (von Hippel, 2005). Como dito, o sucesso das iniciativas é dependente da capacidade de a "oportunidade de contribuir" atrair colaboradores, voluntários ou não. A distinção entre voluntários e contribuidores remunerados é importante, pois, provavelmente, estrategistas preferem contribuições voluntárias (KOCH, 2004; STEWART e GOSAIN, 2006).

Consequentemente, mas não intencionalmente, a importância de atrair voluntários em quantidade expressiva para os estrategistas fomenta a concorrência entre os projetos. Com a quantidade crescente de projetos de software livre e o reconhecimento da quantidade limitada de mão de obra disponível, a situação acentua-se ainda mais (WEST e ANDERSON, 1996). Nessa corrida, informações que visam a explicação e predição da atratividade de projetos de software livre se tornam interessantes, pois elas podem servir de apoio a decisões gerenciais. Decisões estas que fomentam o mercado de software livre, beneficiando a população de forma geral, que terá acesso facilitado a mais, e melhores, softwares e seus processos produtivos.

ATRATIVIDADE DE PROJETOS DE SOFTWARE LIVRE

Atratividade é a capacidade de captar voluntários a um projeto de software livre, de promover visitas ao portal do projeto e *downloads* do software para teste e uso; além de estar correlacionada (1) à quantidade de conteúdo gerado (*e.g.,* bugs encontrados); (2) à eficiência das comunidades; e (3) ao tempo gasto para completar tarefas (SANTOS JR, 2009). Assim, com informações sobre o que influencia a atratividade dos projetos de software livre, é possível explicar e prever o desempenho das comunidades, tornando o entendimento desse construto importante e oportuno.

Em Santos Jr. (2009), o conceito de atratividade em projetos de software livre foi introduzido empiricamente, pela primeira vez, através de um estudo sobre aproximadamente 5.000 projetos do SourceForge.net. Atratividade foi desenvolvida como um construto latente que se expressa de pelo menos três formas, pois: (a) um projeto deve trazer visitantes ao seu portal; (b) os administradores do projeto esperam que os visitantes baixem o software e se tornem usuários; e (c) um interessado pode se tornar membro do projeto e contribuir para o aprimoramento do software. É para explicar e maximizar o número de visitas, *downloads* e membros que o estudo das causas de atratividade se justifica.

Santos Jr. (2009) propôs um modelo teórico completo, onde projetos com diferentes características têm índices de atratividade diversos, sugerindo que tais características influenciam na atratividade (Figura 1). Entre essas características estão a(s) licença(s) sob a qual o software foi disponibilizado à comunidade (*e.g.,* GPL – GNU *General Public License*), o público-alvo do projeto (*e.g.,* desenvolvedores ou usuários finais), os estágios de desenvolvimento das versões do software disponibilizadas no portal (*e.g.,* beta ou estável) e o domínio da aplicação (*e.g.,* multimídia ou banco de dados).

Como dito anteriormente, nem todos os objetivos de uma organização que tornou seu software livre são atingidos quando seu projeto recebe visitas, seu software é baixado, ou voluntários são recrutados. Na literatura, sucesso no contexto de projetos de software livre já foi observado de diversas formas, como: número de *downloads* (STEWART e GOSAIN, 2006); habilidade de avançar nas fases de desenvolvimento – de "beta" para "estável", por exemplo (RAJA e TRETTER, 2006); ou a quantidade de membros de um projeto (CROWSTON e HOWISON, 2006; KOCH, 2004). Mas essas medidas não representam sucesso completamente; elas são meios de se chegar a ele (ou mantê-lo).

As organizações esperam que suas aplicações sejam aprimoradas pela comunidade, além de que conteúdo traduzível em redução de custos, aumento de receitas ou melhoria de software/serviço seja gerado dentro dessas comunidades (SANTOS JR, 2008). Em outras palavras, se a atratividade de um projeto deve ser considerada crucial para as organizações ou indivíduos que tornam suas aplicações "livres", ela tem de influenciar variáveis como (1) nível de atividade dessas comunidades; (2) qualidade desses softwares; ou (3) qualidade do serviço prestado pela comunidade aos usuários da aplicação. Assim, atratividade também não deve ser considerada "sucesso" por si só.

Diversos pesquisadores já sugeriram conexões entre variáveis como o tamanho dos grupos (uma faceta de atratividade) com índices de produtividade (CROWSTON e outros, 2005; KOCH, 2004). Entre esses índices, pode-se citar a quantidade de linhas de código gerada (MOCKUS e outros, 2000), a velocidade de resolução de *bugs* e a quantidade de *bugs* encontrados e resolvidos (STEWART e GOSAIN, 2006) e, ainda, a probabilidade de resolução das requisições abertas (SANTOS JR, 2009). Dessa forma, a importância do estudo e da administração da atratividade dos projetos fica estabelecida, justificando atenção especial para suas causas com o intuito de administrá-las estrategicamente e, assim, favorecer o aprimoramento do software.

Com isto em mente, o estudo dos impactos diretos de decisões pontuais, como a da escolha da licença sob a qual liberar um código-fonte na atratividade de um projeto de software livre (e indiretos dessas mesmas decisões em índices de produtividade, mediados pela atratividade deles), apresenta-se como uma poderosa ferramenta gerencial que estreita as relações entre academia e mercado.

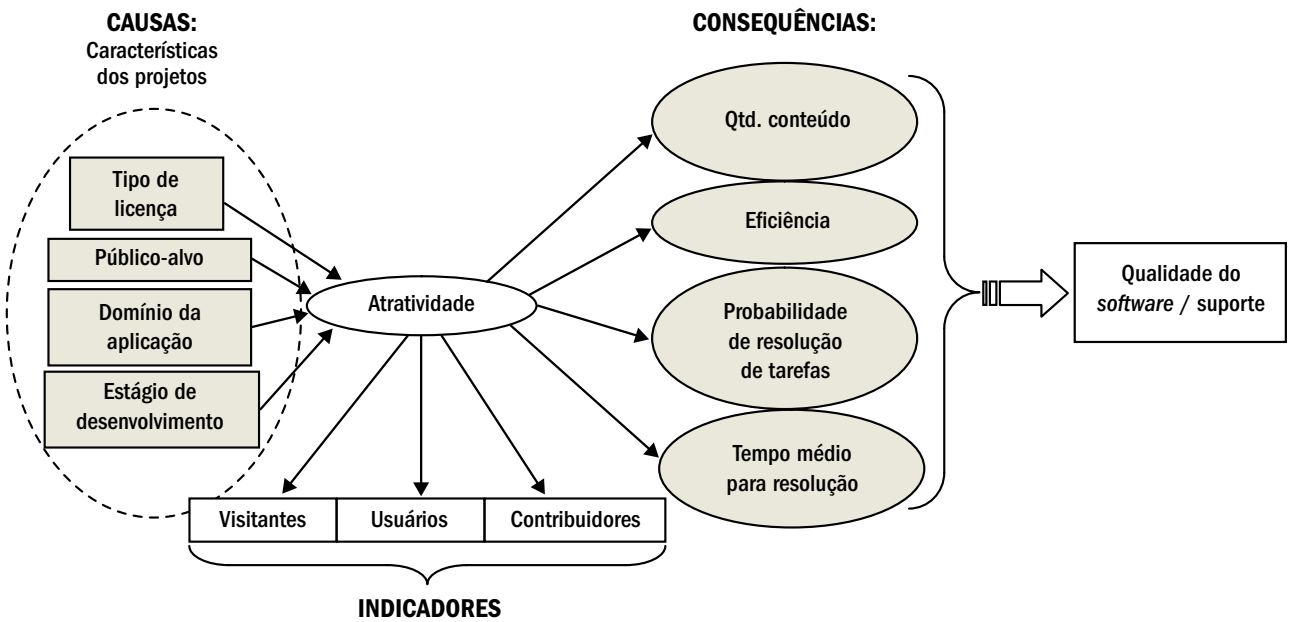
A seguir, apresenta-se um estudo empírico realizado para averiguar as hipóteses até então discutidas em relação às causas da atratividade dos projetos. O objetivo da próxima seção é apresentar um teste independente e alternativo daquele que pode ser visto em Santos Jr. (2009), ilustrando como as causas de atratividade podem ser observadas. Posteriormente, algumas das causas de atratividade identificadas e validadas empiricamente são discutidas com maiores detalhes, através da elaboração de pareceres pontuais para a aplicação prática direta na administração de projetos de software livre.

TESTE EMPÍRICO: MÉTODO E RESULTADOS

A enorme quantidade de projetos de software livre exige a utilização de uma amostra igualmente grande para representar um teste realista de qualquer teoria geral sobre suas dinâmicas. Algumas fontes de dados sobre PSL estão disponíveis na Internet. Por exemplo, a Universidade de Notre Dame (http://www.nd.edu/~oss/Data/data.html) disponibiliza informações sobre todos os projetos do Sourceforge.net para pesquisadores credenciados. Adicionalmente, há o projeto FLOSSmole.org, que inclui dados sobre PSL de outros repositórios além do Sourceforge.net, e que pode ser acessado sem a necessidade de credenciamento (HOWISON e outros, 2006). Apesar do acesso a essas bases de dados não representar um desafio, seu entendimento e a preparação dos seus dados para análises estatísticas representam. Para amenizar essas dificuldades, o projeto DBOSS (http://ccsl.ime.usp.br/dboss) foi criado.

O DBOSS (*database for open source software statistics*) é um projeto financiado pela FAPESP e em desenvolvimento no Centro de Competência em Software Livre da USP. Os objetivos do DBOSS incluem a unificação das bases de dados de Notre Dame e do FLOSSmole e a preparação automática dos dados coletados para a realização de testes estatísticos. Atualmente, o DBOSS possui informações sobre todos os projetos do Sourceforge.net (via FLOSSmole), o maior repositório de software livre do mundo. Utilizou-se o DBOSS como fonte de dados para a realização dos testes apresentados a seguir, especialmente

Figura 1 – **Causas, indicadores e consequências de atratividade**



Fonte: Adaptado de SANTOS JR. (2009, p. 37).

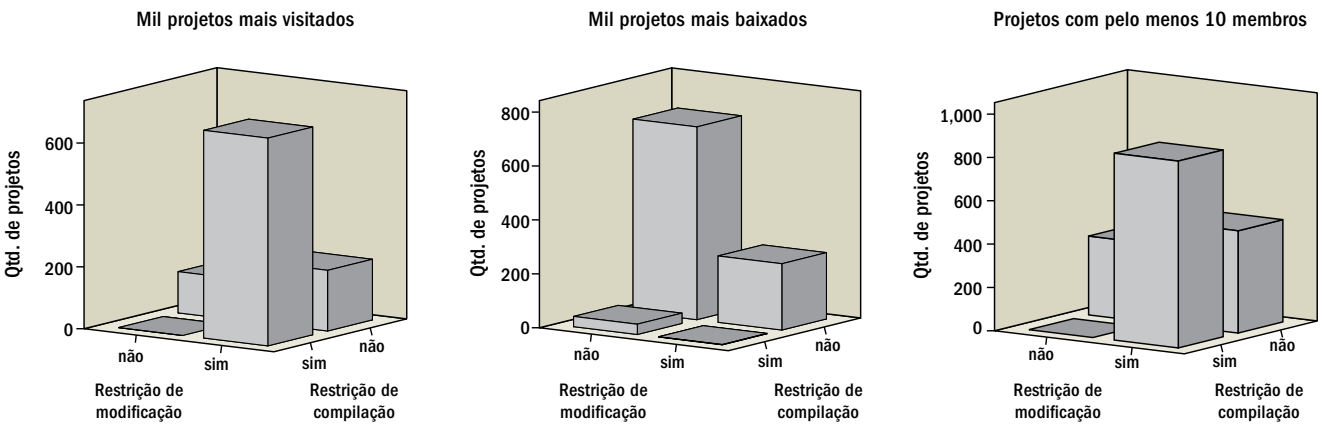
para diferenciar da amostra de Santos Jr. (2009), que utilizou Notre Dame como fonte. Obteve-se dados relativos a maio de 2009.

No total, o DBOSS possui informações sobre mais de 164 mil projetos. Mas essa amostra total não pode ser utilizada, requerendo filtro, pois alguns projetos são inativos ou não possuem software, por exemplo. Para contornar essas limitações, primeiramente, dividiu-se a amostra total em três subamostras, representando uma parcela dos projetos mais atrativos. A primeira amostra inclui os mil projetos mais visitados, a segunda tem os mil projetos mais baixados e a terceira os projetos com pelo menos 10 membros. Com esse agrupamento, é possível investigar como se dá a distribuição dentro dessas amostras "atrativas" para averiguar se há prevalência de determinados tipos de projetos nelas. Pois, se determinado tipo de projeto (*e.g.,* em versão estável, ou já testada) é mais atrativo que um projeto em beta (ainda em teste), deve haver mais projetos em versão estável nessas subamostras do que em beta. Especificamente, focaram-se quatro características dos projetos (tipo de licença, público-alvo, domínio da aplicação e estágio de desenvolvimento), conforme pode ser visto na Figura 1.

Para estudar os efeitos do tipo de licença como o software foi disponibilizado, classificou-se cada um dos projetos de acordo com duas restrições impostas pela(s) licença(s) adotada(s), que são discutidas em detalhes na seção "Os efeitos das restrições impostas pelas licenças". Assim, foram criados quatro grupos de projetos em uma matriz 2 por 2 (restrição de modificação, sim ou não, e de compilação, sim ou não) em cada uma das três amostras "atrativas". Na primeira amostra, a dos mil mais visitados, observa-se que aproximadamente 600 projetos possuem as duas principais restrições impostas por licenças em software livre, enquanto menos de 200 projetos não possuem nenhuma restrição. Com pouco mais de 200 projetos, aparece o grupo que tem apenas a restrição de modificação. Finalmente, não existem projetos nessa amostra apenas com a restrição de compilação. Esse padrão de frequência de projetos por restrição de licença se repete tanto na amostra dos mil projetos mais baixados quanto na dos projetos com pelo menos 10 membros, dando indicativos consistentes de que a escolha da licença de fato influencia a atratividade do projeto (ver Gráfico 1).

Para avaliar os efeitos do domínio da aplicação (processo específico que o software auxilia), coletaram-se informações sobre os "tipos dos projetos" que estão disponíveis em suas páginas Web e representam os "mercados" em que eles operam e disputam usuários e voluntários. Para classificar os projetos, criaram-se duas variáveis binárias (multimídia e banco de dados) que receberam valor "1" quando o projeto pertencia àquele(s) grupo(s), e "0" quando não. Dessa forma, foi possível comparar a frequência do aparecimento de projetos desses dois tipos em cada uma das três amostras "atrativas". Entre os mil projetos mais visitados, cerca de 300 deles têm relação com multimídia e menos de 50 são relacionados a banco de dados (ferramentas de auxílio ao desenvolvimento de banco de dados). A maior parte dos projetos dessa amostra não tem relação com esses domínios de aplicação, ficando, então, enquadrados como "outros". Isso não é uma surpresa, dada a grande quantidade de domínios de aplicação a que os projetos podem pertencer (mais de 30). O padrão de aparecimento dos projetos é similar nas amostras dos projetos mais baixados e com pelo menos 10 membros (ver Gráfico 2). Assim, fica evidente que os projetos multimídia

Gráfico 1 – **Distribuição da quantidade de projetos por restrição de licença em três amostras atrativas**



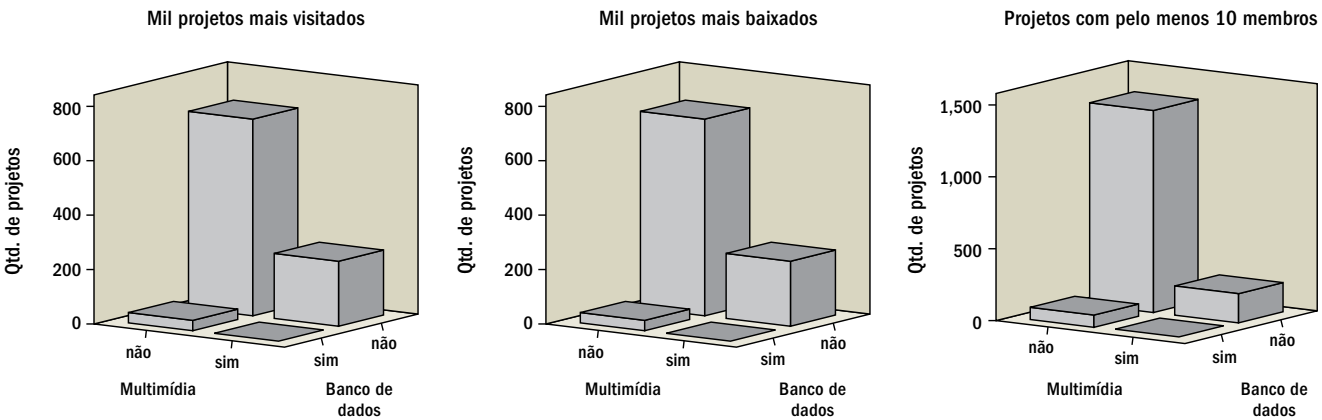
aparecem com mais frequência que os projetos banco de dados, sugerindo que também há influência do domínio da aplicação nos níveis de atratividade.

De forma análoga, obtiveram-se informações relativas ao público-alvo dos projetos, sobre o tipo de usuário que os softwares dos projetos auxiliam. Ilustrativamente, classificamos os projetos com foco em usuário final e desenvolvedor de software. Um editor de texto, por exemplo, busca usuários finais, enquanto uma linguagem de programação auxilia desenvolvedores. Através dos resultados, verificou-se que entre os mil projetos mais visitados, cerca de 400 deles visam exclusivamente a usuários finais, uma audiência não técnica. Adicionalmente, pouco mais de 200 projetos têm foco exclusivo em desenvolvedores e, aproximadamente, 300 projetos focam em usuários finais e desenvolvedores. Conforme pode ser visto no Gráfico 3, o padrão de frequência dos projetos é similar na amostra dos mil projetos mais baixados. Já na amostra dos projetos com pelo menos 10 membros, o padrão é diferente, mas o público-alvo parece continuar a exercer uma influência e diferenciar a atratividade dos projetos. Nessa terceira amostra, os projetos com foco exclusivo em desenvolvedores são os mais comuns (500+), seguidos pelos que focam em outros públicos (500-), pelos que focam em desenvolvedores e usuários finais (400-) e por aqueles que buscam usuários finais (300-). Isto sugere que projetos técnicos (desenvolvedores) têm maior facilidade em encontrar contribuidores do que projetos não técnicos (usuários finais), que, por sua vez, serão mais visitados e baixados que os técnicos.

Finalmente, para explorar os efeitos do estágio de desenvolvimento (grau de maturidade) do software na atratividade, classificaram os projetos em software em beta (em teste) e estável (testado). Se o estágio tem efeito sobre a atratividade, projetos em diferentes estágios devem possuir diferentes quantidades de visitantes, usuários e contribuidores. Em conformidade com essa expectativa, verificou-se que, entre os mil projetos mais visitados, mais de 500 estão no estágio estável, exclusivamente. Em segundo lugar, estão os projetos com software em beta (200+), seguidos pelos que não possuem software nesses dois estágios e pelos que possuem software nas duas versões, uma em beta e outra em estável (100-). A distribuição dos mil projetos mais baixados é similar à dos mais visitados. Mas a distribuição dos projetos com pelo menos 10 membros é diferente, apesar de que projetos em estágios distintos continuam a apresentar diferentes índices de atração (Gráfico 4). Depois dos projetos com software em "outros", estável é o estágio preferido pelos contribuidores (quase 600 projetos), seguidos pelos projetos em beta (mais de 300) e pelos que têm as duas versões (cerca de 100).

A análise anterior baseou-se na disparidade da quantidade de determinados tipos de projetos em amostras altamente atrativas para justificar a hipótese de que alguns tipos de projeto são mais atrativos do que outros, aparecendo com maior frequência. Existem diversas formas de testar uma mesma hipótese. A seguir, apresenta-se um teste alternativo, mais abrangente e robusto. Dessa vez, compararam-se as médias, os máximos, e os desvios-padrão do número de visitas, *downloads*, e membros dos projetos de diferentes categorias, levando em consideração a amostra completa. Por amostra completa, entendem-se todos os projetos ativos (sem data de desligamento) e com software para ser baixado (número de *downloads* maior que zero). A amostra total

Gráfico 2 – **Distribuição da quantidade de projetos por domínio da aplicação (tipo de projeto) em três amostras atrativas**



era de aproximadamente 164 mil projetos; pós-filtro, a amostra passou a ser de 65.446.

Os projetos foram divididos em quatro grandes categorias (licença, público-alvo, domínio da aplicação e estágio de desenvolvimento). Dentro dessas categorias, cada projeto foi classificado como tendo, ou não, até três características (*e.g.,* licença: sem restrição, modificação e compilação). Os projetos podem escolher qualquer combinação dessas três características, o que resulta em mais grupos de projetos do que características. Para cada grupo, calcularam-se a média, o desvio-padrão e o máximo, em cada uma das variáveis dependentes (visitas, *downloads* e membros). Posteriormente, utilizou-se a técnica de análise multivariada de variância (MANOVA) para verificar se havia diferença estatisticamente significativa entre os grupos de determinada categoria, controlando pela idade do projeto e levando em consideração as três variáveis dependentes simultaneamente (ver sobrescrito "a" na Tabela 1). Com esse resultado significativo para todas as categorias ($P < 0.01$), realizou-se uma análise univariada de variância (ANOVA), para verificar em qual(is) variável(is) dependente(s) as diferenças estavam (ver sobrescrito "b" na Tabela 1). Como os resultados dos testes ANOVA também foram significativos para cada uma das variáveis dependentes, prosseguiu-se com uma série de comparações de pares (*i.e.*, entre a média de cada um dos grupos e a de todos os outros). Nessas comparações de pares, utilizou-se o método de correção Bonferroni para manter o erro tipo I (alfa) em 0,05. Quando o resultado do teste de comparação da média de um grupo com a de outro foi encontrado significativo, o número que identifica o grupo que aquele difere estatisticamente foi colocado

Gráfico 3 – **Distribuição da quantidade de projetos por tipo de usuário em três amostras atrativas**

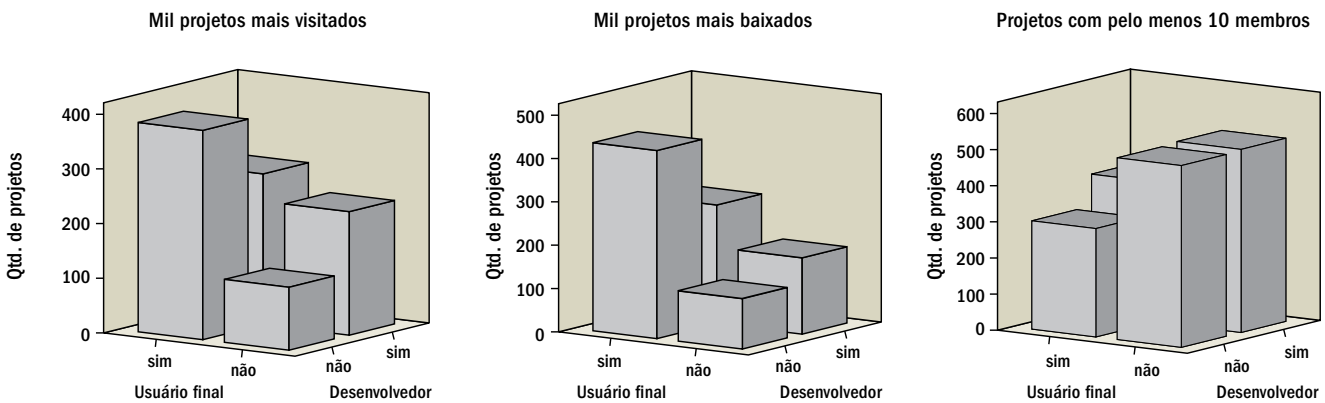
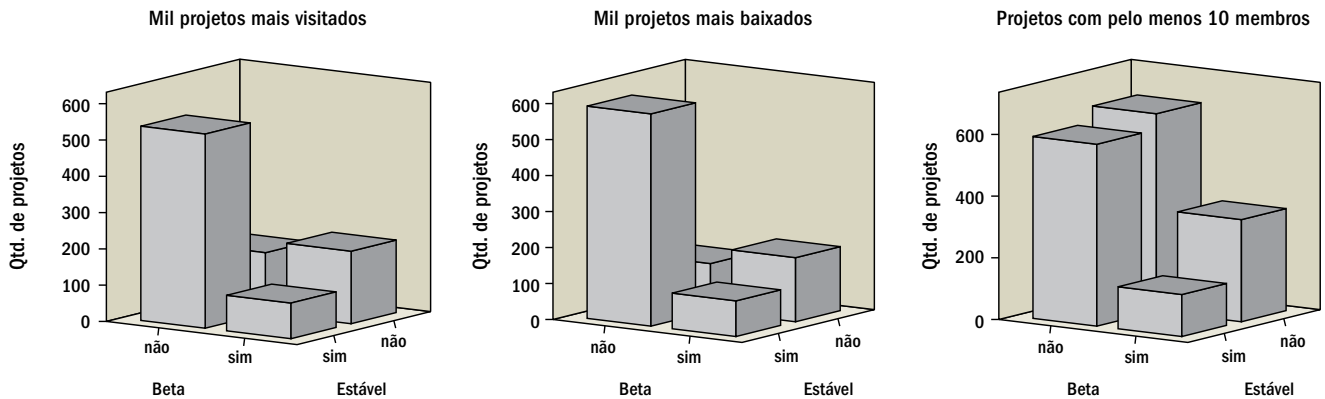


Gráfico 4 – **Distribuição da quantidade de projetos por estágio de desenvolvimento do software em três amostras atrativas**



de forma sobrescrita em sua média. Todos os resultados dessa análise estão na Tabela 1. A seguir, ilustrativamente, discutem-se os efeitos das restrições das licenças na atratividade dos projetos. Uma leitura correlata pode ser feita em relação às outras grandes categorias.

Para capturar os efeitos da licença, os projetos foram separados em seis grupos distintos: (1) sem restrição; (2) múltiplas licenças, mas sem restrição; (3) restrição de modificação; (4) múltiplas licenças com restrição de modificação; (5) restrição de modificação e compilação (GPL); e (6) múltiplas licenças com restrição de modificação e compilação (a explicação do significado de cada uma dessas restrições se encontra na próxima seção). Os valores de média, máximo, desvio-padrão (risco) e quantidade de projetos (popularidade da configuração) foram, então, calculados por grupo. Ordenando-se os grupos de acordo com esses valores, obtêm-se informações sobre seus graus de atratividade, comparativamente.

No caso das licenças, a escolha mais popular é a GPL, que é seguida pelas licenças sem restrição (*e.g.,* BSD – *Berkeley Software Distribution License*) e pelas que possuem apenas restrição de modificação (LGPL – GNU *Lesser* GPL). Quanto à atratividade, em média, os projetos mais visitados possuem licenças múltiplas com restrição de modificação, que são estatisticamente diferentes dos projetos GPL (grupo 5) e sem restrição (grupo 1), com $P < 0{,}01$; os mais baixados têm licenças múltiplas com restrições de modificação e compilação (estatisticamente diferente de todos os outros grupos, exceto do "2) Licenças múltiplas (sem restrição)"); e os com mais membros têm licenças múltiplas sem restrições (acima da média geral e estatisticamente diferente dos grupos 1 e 5). A abordagem contingencial de licenciamento parece ser positiva para a atratividade do projeto.

Ainda em relação às médias, os projetos menos visitados possuem LGPL (com média estatisticamente diferente dos grupos 1 e 5), os menos baixados não possuem restrição (estatisticamente diferente de todos, exceto grupo 5), e os com menos membros, único grupo abaixo da média geral, têm GPL (estatisticamente diferente de todos os outros grupos). Adicionalmente, o projeto que recebeu o maior número de visitas de todos os tempos (máximo) e foi baixado o maior número de vezes é licenciado sob a GPL; e o projeto que possui mais membros é LGPL. Quanto ao desvio-padrão, o grupo com maior dispersão de visitas e *downloads*, acima da média geral, é GPL (e o com menor, LGPL); enquanto a maior variação de número de membros se encontra no grupo com licenças múltiplas e sem restrição (e a menor no grupo GPL).

Apesar de variados, esses resultados sugerem que, em média, os projetos mais atrativos possuem software com opções múltiplas de licenciamento. Além disso, a escolha mais popular, pela GPL, parece resultar em projetos medianamente atrativos em relação a usuários e visitantes e pouco atrativos na obtenção de contribuidores. Essa baixa atratividade de contribuidores é contrabalanceada por um risco (desvio-padrão) menor na atração deles. Projetos GPL atraem poucos desenvolvedores, mas, de forma consistente, tendem a ficar mais próximos da quantidade média de membros. Infelizmente, os riscos de falhar em atrair visitantes e usuários são maiores quando tal licença é escolhida, acima da média geral. Tendo em vista esses efeitos desejados e colaterais da GPL, fundadores de PSL devem, antes de optarem pela licença mais popular, estabelecer os objetivos que desejam atingir com a abertura do código (*e.g.,* mais usuários ou contribuidores). A seguir, passa-se a uma discussão detalhada dos efeitos das restrições das licenças na atratividade do projeto, buscando uma aplicação estratégica e gerencial dos resultados obtidos. Posteriormente, uma discussão similar é apresentada para cada uma das outras características dos projetos.

OS EFEITOS DAS RESTRIÇÕES IMPOSTAS PELAS LICENÇAS

O que define formalmente se um software é livre é sua licença. Tornar um software "livre" é disponibilizá-lo sob uma licença que garanta sua utilização e modificação, mesmo que com devidas restrições. É no momento da escolha da licença que se decide em que termos o software será distribuído (SABINO e KON, 2009). Dentre as licenças mais utilizadas por softwares livres estão a GPL, a LGPL e a BSD, mas existem dezenas de licenças disponíveis. Assim, para facilitar o seu entendimento e a sua escolha, as licenças são classificadas de acordo com as restrições que elas impõem (SANTOS JR, 2009; SABINO e KON, 2009). As licenças são classificadas em (1) recíprocas totais (restrições de modificação e compilação); (2) recíprocas parciais (restrição de modificação); e (3) permissivas (sem restrições).

As recíprocas totais são as licenças que obrigam os trabalhos derivados e os que compilem com ligações ao código original, a serem distribuídos nos mesmos termos da licença original (restrição de modificação e compilação). A intenção dessas licenças é garantir que a comunidade continue a ter acesso às melhorias dos produtos e ao conhecimento relacionado ao código comunitário inicial. A adoção de licenças desse tipo busca garantir

Tabela 1 – **Estatísticas descritivas e de comparação de médias por categoria de projeto (Parte 1)**

CATEGORIAS DE PROJETOS	QTD. DE PROJETOS	MÉDIA		
LICENÇAS (RESTRIÇÕES) [a,b]		VISITAS	DOWNLOADS	MEMBROS
1) Sem restrição (acadêmica)	10992 (2)	226611,71 (5)3,4*,5*,6*	7968,69 (6)2*,3*,4*,6*	2,36 (5)2*,3*,4*,5*,6*
2) Licenças múltiplas (sem restrição)	267 (6)	741122,41 (3)5	14625,51 (4)1*,3,5*	3,53 (1)1*,5*
3) Restrição de modificação (lgpl)	8165 (3)	205843,62 (6)1*,5*	13339,06 (5)1*,2,4*,5*,6*	2,6 (4)1*,4*,5*,6*
4) Licenças múltiplas (restrição de modificação)	3029 (4)	1533667,6 (1)1*,5*	45090,39 (2)1*,3*,5*,6	3,1 (3)1*,3*,5*
5) Restrições de modificação e compilação (GPLs)	42578 (1)	704749,94 (4)1*,2,3*,4*,6*	30774,32 (3)2*,3*,4*,6*	2,03 (6)1*,2*,3*,4*,6*
6) Licenças múltiplas (restrições de modificação e compilação)	415 (5)	1306760,6 (2)1*,5*	55982,17 (1)1*,3*,4,5*	3,36 (2)1*,3*,5*
Tipo de usuário (Público-alvo) [a,b]				
1) Outros	17868 (1)	107020,33 (8)2*,3*,4*,5*,6*,7*,8*	4010 (8)2*,3*,4*,5*,6*,7*,8*	2,21 (5)2*,5*,6*,7*
2) Administradores de sistema	3115 (5)	328391,3 (5)1*,3*,5*,6*,7*,8*	6990,34 (6)1*,4,5*,6*,7*,8*	1,84 (8)1*,3*,4*,7*,8*
3) Desenvolvedores	15675 (3)	153995,48 (7)1*,2*,4,5,7*	6124,83 (7)1*,5*,6*,7*,8*	2,25 (4)2*,5*,6*,7*
4) Administradores de sistema e desenvolvedores	2127 (7)	245736,57 (6)1*,3,5*,6*,7*,8	24257 (5)1*,2,5*,7*,8*	2,55 (3)2*,5*,6*,7*
5) Usuários finais	15721 (2)	901533 (3)1*,2*,3,4*,7*	44793,55 (3)1*,2*,3*,4*,7*,8*	1,88 (7)1*,3*,4*,7*,8*
6) Usuários finais e administradores de sistemas	2135 (6)	2525625,27 (2)1*,2*,4*	52770,03 (2)1*,2*,3*,7*,8	2,02 (6)1*,3*,4*,7*,8*
7) Usuários finais e desenvolvedores	6700 (4)	646935,25 (4)1*,2*,3*,4*,5*	42219,88 (4)1*,2*,3*,4*,5*,6*	2,94 (1)1*,2*,3*,4*,5*,6*,8*
8) Usuários finais e desenvolvedores e administradores de sistemas	2105 (8)	4652116,73 (1)1*,2*,4	156649,44 (1)1*,2*,3*,4*,5*,6	2,75 (2)2*,5*,6*,7*
Tipo do projeto (Domínio da aplicação) [a,b]				
1) Outros	43992 (1)	672438,52 (3)2*,3*,4*,5*,6*,7	26606,48 (2)2*,3*,4*,5*,6*,7*	2,18 (6-7)2*,4*,6*
2) Ferramentas de desenvolvimento	11105 (2)	219127,25 (5)1*,3*,5*,6*	10055 (5)1*,5*,6*	2,33 (5)1*,3*,5*
3) Banco de dados	2428 (4)	362972,69 (4)1*,2*,4*,5*,6*	17237,42 (3)1*,4*,5*,6*	2,18 (6-7)2*,4*,6*
4) Banco de dados e ferramentas de desenvolvimento	706 (5)	147247,62 (6)1*,3*	7169,06 (6)1*,2*,3*,5*	2,78 (1)1*,3*,5*
5) Multimídia	6457 (3)	940619,77 (1)1*,2*,3*	51543,77 (1)1*,2*,3*,4*	2,17 (8)2*,4*,6*
6) Multimídia e ferramentas de desenvolvimento	613 (6)	783653,17 (2)1*,2*,3*	13226,41 (4)1*,2*,3*	2,76 (2)1*,3*,5*
7) Multimídia e banco de dados	125 (7)	65808,87 (8)1	4226,93 (7)1*	2,7 (3)
8) Multimídia e banco de dados e ferramentas de desenvolvimento	20 (8)	68568,8 (7)	1471,95 (8)	2,55 (4)
Estágio de desenvolvimento [a,b]				
1) Outros	18777 (1)	77741,59 (8)2*,3*,4*,5*,6*,7*,8*	4062,39 (7)2*,3*,4*,5*,6*,7*,8*	2,23 (6)2*,3*,4*,5*,6*,7*,8*
2) Estável	15772 (3)	1746759,76 (2)1*,3*,4*,5*	77089,72 (2)1*,3*,4*,5*,7*	2,52 (5)1*,3*,4*,5*,6*,8*
3) Beta	17326 (2)	298461,28 (5)1*,2*,4*,5*,6*	10597,18 (6)1*,2*,4*,5*,6*,8*	1,96 (7)1*,2*,4*,5*,6*,7*,8*
4) Beta e estável	1698 (5)	2498543,55 (1)1*,2*,3*,5*,7*	80659,65 (1)1*,2*,3*,5*,7*	3,85 (3)1*,2*,3*,5*,7*
5) Alpha	11019 (4)	80072,49 (7)1*,2*,3*,4*,6*,7*,8	2988,48 (8)1*,2*,3*,4*,6*,7*,8*	1,84 (8)1*,2*,3*,4*,6*,7*,8*
6) Alpha e estável	131 (8)	369196,34 (3)1*,3*,5*	52322,21 (4)1*,3*,5*,7*	3,86 (2)1*,2*,3*,5*,7*
7) Alpha e beta	561 (6)	270324,53 (6)1*,4*,5*	11298,74 (5)1*,2*,4*,5*,6*	2,81 (4)1*,3*,4*,5*,6*
8) Alpha e beta e estável	162 (7)	361020,38 (4)1*,5	72237,45 (3)1*,3*,5*	4,23 (1)1*,2*,3*,5*
TOTAL GERAL	65446	604531,02	25525,33	2,22

Legenda:
a: Indica que o resultado do teste multivariado de variância (MANOVA) é significativo (Wilks' Lambda; P < 0.01). A data de criação do projeto foi utilizada como variável-controle, pois projetos mais antigos têm mais chance de serem visitados, por exemplo.
Teste baseado na transformação logarítmica das variáveis dependentes, realizada para resolver problemas de normalidade, que foi avaliada através de Skewness e Kurtosis fora do intervalo -1 e 1.
b: Indica que o resultado do teste univariado de variância (ANOVA), que separa os efeitos em visitas, downloads e membros, é significativo para cada uma delas (F-teste; P < 0.01). A data de criação do projeto foi utilizada como variável-controle.
Teste baseado na transformação logarítmica das variáveis dependentes, realizada para resolver problemas de normalidade, que foi avaliada através de Skewness e Kurtosis fora do intervalo -1 e 1.
Nota 1: Números entre parênteses indicam a posição daquele grupo em comparação com os outros (ranking). Por exemplo, o grupo "Licenças Múltiplas (restrição de modificação)" tem a maior (1) média de downloads dentre os diferentes grupos de tipos de licenças.
Nota 2: Os números sobrescritos aos números entre parênteses indicam os grupos que diferem estatisticamente daquele grupo, com correção Bonferroni (P < 0.05). Quando P < 0.01, o número aparece com um asterisco. Por exemplo, dentro da categoria Licenças, o grupo "5) Restrições de Modificação e Compilação (GPLs)" possui média de número de membros estatisticamente diferente dos grupos 1), 2), 3), 4), e 6), com P < 0.01.

Tabela 1 – **Estatísticas descritivas e de comparação de médias por categoria de projeto (Parte 2)**

CATEGORIAS DE PROJETOS	MÁXIMO			DESVIO-PADRÃO		
LICENÇAS (RESTRIÇÕES) [a,b]	VISITAS	DOWNLOADS	MEMBROS	VISITAS	DOWNLOADS	MEMBROS
1) Sem restrição (acadêmica)	390363773 (4)	5127847 (5)	272 (2)	4909316,75 (5)	96664,31 (5)	4,79 (5)
2) Licenças múltiplas (sem restrição)	134982678 (6)	809152 (6)	75 (5-6)	8343951,33 (4)	64955,64 (6)	7,63 (1)
3) Restrição de modificação (lgpl)	238009101 (5)	43412985 (2)	428 (1)	3369464,07 (6)	487550,69 (4)	6,29 (2)
4) Licenças múltiplas (restrição de modificação)	1219424330 (2)	34233918 (3)	133 (4)	34532569,55 (2)	732524,15 (2)	6,02 (3)
5) Restrições de modificação e compilação (GPLs)	8082625022 (1)	276650340 (1)	247 (3)	45180644,94 (1)	1758323,06 (1)	3,59 (6)
6) Licenças múltiplas (restrições de modificação e compilação)	392253767 (3)	8547290 (4)	75 (5-6)	19412420,82 (3)	545008,89 (3)	6,01 (4)
Tipo de usuário (Público-alvo) [a,b]						
1) Outros	731648064 (4)	11916078 (6)	272 (2)	5878856,7 (6)	124391 (6)	4,25 (4)
2) Administradores de sistema	390363773 (6)	5127847 (7)	69 (7)	8272861,8 (5)	100051,35 (7)	2,45 (8)
3) Desenvolvedores	181935035 (7)	4928728 (8)	149 (4)	2032565,7 (8)	68060,84 (8)	3,68 (5)
4) Administradores de sistema e desenvolvedores	94251685 (8)	18306510 (5)	85 (5)	2841929,07 (7)	465924,29 (5)	4,42 (3)
5) Usuários finais	2274690494 (3)	276650340 (1)	80 (6)	27478622,72 (3)	2267070,18 (2)	2,76 (7)
6) Usuários finais e administradores de sistemas	3324388012 (2)	18430064 (4)	59 (8)	74299740,17 (2)	714763,26 (3)	2,96 (6)
7) Usuários finais e desenvolvedores	634883570 (5)	24209454 (3)	428 (1)	10310440,54 (4)	567494,96 (4)	7,85 (1)
8) Usuários finais e desenvolvedores e administradores de sistemas	8082625022 (1)	219720250 (2)	247 (3)	176755416,18 (1)	4898694,21 (1)	7,48 (2)
Tipo do projeto (Domínio da aplicação) [a,b]						
1) Outros	8082625022 (1)	276650340 (1)	428 (1)	44773946,35 (1)	1724308,72 (1)	4,47 (5)
2) Ferramentas de desenvolvimento	391292361 (4)	9318930 (4)	162 (2)	4220504,9 (5)	155426,97 (4)	4,56 (3)
3) Banco de dados	519498584 (3)	9432437 (3)	103 (4)	10655033,56 (3)	297183,59 (3)	3,69 (8)
4) Banco de dados e ferramentas de desenvolvimento	14799396 (6)	628095 (6)	70 (5)	755186,64 (6)	37093,31 (6)	4,75 (2)
5) Multimídia	1217922972 (2)	39905079 (2)	126 (3)	19065653,5 (2)	801194,04 (2)	3,8 (7)
6) Multimídia e ferramentas de desenvolvimento	181935035 (5)	1287822 (5)	63 (6)	8920959,42 (4)	77010,46 (5)	4,52 (4)
7) Multimídia e banco de dados	1215904 (7)	173180 (7)	50 (7)	179699,93 (7)	16766,36 (7)	5,74 (1)
8) Multimídia e banco de dados e ferramentas de desenvolvimento	480764 (8)	6399 (8)	20 (8)	124058,43 (8)	1819,53 (8)	4,26 (6)
Estágio de desenvolvimento [a,b]						
1) Outros	134982678 (5)	9318930 (4)	272 (2)	1780405,94 (4)	100757,23 (6)	4,21 (6)
2) Estável	8082625022 (1)	276650340 (1)	247 (3)	73583753,15 (1)	2900808,41 (1)	5,37 (3)
3) Beta	1232118852 (2)	21610849 (2)	84 (4)	11591521,9 (3)	261304,46 (5)	2,71 (7)
4) Beta e estável	1219424330 (3)	19110342 (3)	428 (1)	43474123,19 (2)	833416,23 (2)	12,56 (1)
5) Alpha	173006051 (4)	2934063 (7)	59 (6-7)	1762807,31 (5)	41237,95 (8)	2,19 (8)
6) Alpha e estável	15192094 (8)	3021535 (6)	34 (8)	1556356,76 (8)	296745,27 (4)	4,79 (4)
7) Alpha e beta	26035968 (6)	1490927 (8)	59 (6-7)	1715162,57 (7)	89434,82 (7)	4,38 (5)
8) Alpha e beta e estável	18862215 (7)	8547290 (5)	75 (5)	1741434,54 (6)	678478,8 (3)	8,53 (2)
TOTAL GERAL	8082625022	276650340	428	37301675,99	1438556,86	4,4

Legenda:

a: Indica que o resultado do teste multivariado de variância (MANOVA) é significativo (Wilks' Lambda; $P < 0.01$). A data de criação do projeto foi utilizada como variável-controle, pois projetos mais antigos têm mais chance de serem visitados, por exemplo.

Teste baseado na transformação logarítmica das variáveis dependentes, realizada para resolver problemas de normalidade, que foi avaliada através de Skewness e Kurtosis fora do intervalo -1 e 1.

b: Indica que o resultado do teste univariado de variância (ANOVA), que separa os efeitos em visitas, downloads e membros, é significativo para cada uma delas (F-teste; $P < 0.01$). A data de criação do projeto foi utilizada como variável-controle.

Teste baseado na transformação logarítmica das variáveis dependentes, realizada para resolver problemas de normalidade, que foi avaliada através de Skewness e Kurtosis fora do intervalo -1 e 1.

Nota 1: Números entre parênteses indicam a posição daquele grupo em comparação com os outros (ranking). Por exemplo, o grupo "Licenças Múltiplas (restrição de modificação)" tem a maior (1) média de downloads dentre os diferentes grupos de tipos de licenças.

Nota 2: Os números sobrescritos aos números entre parênteses indicam os grupos que diferem estatisticamente daquele grupo, com correção Bonferroni ($P < 0.05$). Quando $P < 0.01$, o número aparece com um asterisco. Por exemplo, dentro da categoria Licenças, o grupo "5) Restrições de Modificação e Compilação (GPLs)" possui média de número de membros estatisticamente diferente dos grupos 1), 2), 3), 4), e 6), com $P < 0.01$.

a sustentabilidade do movimento de software livre por "força de lei". A GPL representa a origem dessas ideias de reciprocidade em licenças e é a escolha mais popular dos criadores de projetos de software livre. Conforme os resultados demonstraram, a decisão de adotar uma licença recíproca total afeta a atratividade do projeto.

A opção por uma licença com reciprocidade total é positiva para a atratividade do projeto de uma forma geral (ver Gráfico 1). Uma parcela considerável dos desenvolvedores de software livre considera essa restrição sustentadora do modelo "livre" de desenvolvimento e, portanto, prioriza o envolvimento em projetos e a utilização de softwares desse tipo (SANTOS JR., 2009).

Especificamente, a GPL impõe duas restrições ao software do projeto: (a) que o código-fonte de qualquer modificação feita no trabalho original seja incluído com a nova versão do software que será redistribuída; e (b) que o software livre não troque informações com software proprietário durante a compilação, pois o software proprietário teria que se tornar livre por reciprocidade. Como dito anteriormente, a escolha pela GPL é em geral positiva, aumentando as chances de o projeto se tornar um dos mais populares. Não é sem razão que GPL seja a escolha da grande maioria dos projetos. O ponto negativo dessa escolha é a dificuldade de utilização dos projetos GPL em soluções proprietárias, comuns em ambientes comerciais, reduzindo a chance de atrair colaboradores corporativos em geral.

De forma menos restritiva, as licenças chamadas recíprocas parciais não obrigam que os trabalhos que aproveitem o software apenas como um componente, ou troquem informações com ele durante a compilação, sejam disponibilizados sob licença do mesmo tipo. Mas modificações diretas no trabalho original continuam a ter de ser disponibilizadas sob licença do mesmo tipo (SABINO E KON, 2009). A escolha dessas licenças, especialmente a LGPL, tem se tornado mais comum, devido aos criadores dos projetos almejarem que a sua utilização seja mais ampla, sem ferir todos os princípios que buscam sustentar a difusão de software livre. A adoção de licenças desse tipo também influencia a atratividade geral do projeto positivamente (SANTOS JR., 2009).

As licenças chamadas *permissivas*, ou *acadêmicas*, são as menos restritivas. Essas licenças devem ser adotadas quando a intenção for permitir modificações, utilização e distribuição do software de forma não controlada. Os trabalhos derivados de projetos desse tipo podem ser inclusive proprietários (SABINO E KON, 2009). Essa licença é útil quando a exploração comercial dos derivados da tecnologia é provável.

A opção por uma licença que permite a utilização e a modificação do software sem restrições tende a não ser bem vista por desenvolvedores e usuários de software livre (última colocada em média de *downloads* e penúltima em número de membros). Adicionalmente, análises estatísticas mostraram que projetos com esse tipo de licença (*e.g.*, BSD), que possibilitam a criação de software proprietário baseado em software livre, têm sua atratividade geral afetada negativamente (SANTOS JR, 2009).

Por último, existe a possibilidade de licenciamento múltiplo, quando o criador do projeto decide distribuir o software com diferentes licenças, dependendo da natureza do usuário ou colaborador. Um software pode, por exemplo, ser distribuído sob a GPL para pessoas físicas e sob licença proprietária para pessoas jurídicas. Dessa forma, as restrições ao código-fonte dependem da licença que se aplica a determinado usuário. Esta é uma forma de licenciamento que está se tornando popular em projetos de software livre (WATSON e outros, 2008). Em tal forma, intenções comerciais podem ser realizadas ao mesmo tempo em que uma boa imagem é mantida com a comunidade de desenvolvedores de software livre. Um problema com essa escolha é a dificuldade de garantir que em determinada situação uma licença seja válida e em outra não, podendo gerar um problema administrativo custoso. Estatisticamente, o impacto do licenciamento múltiplo é positivo na atratividade (SANTOS JR, 2009). De acordo com a Tabela 1, ele também aparece nas duas primeiras colocações em média de visitas, *downloads* e membros.

OS EFEITOS DOS DOMÍNIOS DA APLICAÇÃO E DOS PÚBLICOS-ALVO

As influências do público-alvo e do domínio da aplicação em atratividade ocorrem de forma similar, através do mesmo mecanismo pelo qual oferta e demanda interagem para a definição de preço na economia clássica. Quanto maior o público-alvo, maior a demanda dos usuários e voluntários; e, dependendo do "mercado" (domínio da aplicação) em que o software atuar, a oferta de mão de obra e o nível de competição são afetados, influenciando, por consequência, a atratividade de um projeto.

De acordo com o SourceForge.net, quanto ao público-alvo, projetos de software livre podem ser desenvolvidos para usuários finais, desenvolvedores, administradores de sistema, usuários finais avançados e outros. Usuário final é um dos maiores grupos em quantidade de pessoas, mas, devido ao perfil não técnico desse grupo, ele não oferece a maior oferta de mão de obra de voluntários.

Alternativamente, um projeto que busca usuários finais tem maior chance de ser procurado e utilizado, pois usuários não técnicos são mais comuns do que desenvolvedores (ver "Projetos com pelo Menos 10 Membros" no Gráfico 2). De forma geral, o efeito de um software auxiliar tarefas interessantes para usuários finais é positivo para sua atratividade (SANTOS JR, 2009).

A comunidade de software livre possui a peculiaridade de que a maior parte dos desenvolvedores também são usuários finais e administradores dos programas de computador que desenvolvem (SANTOS JR. e GONÇALVES, 2008). Assim, mesmo com a quantidade de projetos que auxiliam usuários finais, desenvolvedores e administradores de sistemas sendo pequena (última colocada), esse público-alvo é técnico e imenso, oferecendo uma enorme oferta de contribuidores e usuários (maior média de visitas e *downloads*; segunda maior de membros). Logo, um projeto desse tipo tende a ser altamente atrativo e organizações com softwares desse grupo devem ser encorajadas a disponibilizá-los.

O público-alvo administradores de sistemas encontra-se entre os menores. Dessa forma, deve-se esperar uma procura menor tanto pelo uso do software quanto pelo envolvimento nas atividades de desenvolvimento do projeto. Similarmente, a escolha do público-alvo "outros" deve ser evitada. Possivelmente, "outros" representa uma entidade que não descreve um processo de forma eficaz. O impacto de divulgar o público-alvo como "outros" afeta negativamente a atratividade (SANTOS JR., 2009). O ideal é procurar outra opção que descreva melhor o projeto.

O público-alvo usuários finais avançados também não está entre as maiores populações existentes e interessadas no movimento de software livre. Mas, provavelmente, nessa população, encontram-se usuários com forte inclinação técnica. Essa característica lhes dá a vantagem de identificar *bugs* como um usuário final e, ainda, garante a capacidade de eles reportarem com precisão onde o problema se encontra, facilitando a sua solução. Assim, o foco em usuários finais avançados é positivo para a atratividade (SANTOS JR, 2009).

Em relação ao domínio da aplicação, sabe-se que software de certo tipo foca determinadas tarefas realizadas por usuários específicos. Assim, o domínio da aplicação define as fronteiras de competição entre projetos, já que a maioria dos usuários tende a escolher só um navegador *web*, por exemplo; colocando todos os projetos do domínio *browser* em concorrência. Da mesma forma, espera-se que um contribuidor se dedique a somente um projeto de determinado domínio. Além disso, o domínio do projeto limita a quantidade de potenciais interessados em usar e desenvolver um software.

Os impactos de "competir" em cada um dos maiores domínios disponíveis no SourceForge.net foram avaliados e concluiu-se, resumidamente, que softwares relacionados a banco de dados, ciência, sociologia e educação tendem a ser menos atrativos (SANTOS JR, 2009). Enquanto softwares listados com tópicos relacionados à Internet, segurança, editores de texto, jogos e multimídia são mais atrativos. Em consonância com esses resultados, as análises apresentadas neste artigo mostram que software do tipo banco de dados, em geral, é estatisticamente menos atrativo do que software do tipo multimídia (exceção de número de membros, que não difere estatisticamente – ver também Gráfico 3).

OS EFEITOS DO ESTÁGIO DE DESENVOLVIMENTO DO SOFTWARE

A análise empírica realizada neste artigo, em consonância com estudos anteriores, mostrou que a atratividade de um projeto de software livre é diretamente afetada pelo estágio de desenvolvimento que o software disponibilizado se encontra. De forma resumida, deve-se evitar liberar software em estágios iniciais, onde usuários ainda não possam realizar pelo menos algumas das funções que necessitam, e contribuidores não encontrem material concreto para o qual eles possam contribuir (SANTOS JR., 2009).

Como "planejamento" é o primeiro estágio de desenvolvimento de um software, seu efeito negativo em atratividade é o maior. Pré-*alpha* ainda é um estágio considerado imaturo, sinalizando para a comunidade que o software não está pronto para o uso nem para receber contribuições. Não há garantias de que o projeto será levado à frente pelos seus idealizadores. Em sequência, *alpha* é o estágio de desenvolvimento transicional, por ser o último a ter efeito negativo em atratividade (SANTOS JR., 2009). Quando o software disponibilizado só possui uma versão *alpha*, sua média de *downloads* e membros é a menor de todas, com diferença estatisticamente significativa de todas as outras opções (Tabela 1).

No estágio *beta*, espera-se encontrar um software com funcionalidades básicas, prontas para teste, o que fornece material tanto para usuários quanto para colaboradores. Liberar software em *beta* é a decisão mais popular na hora de criar projetos de software livre (SANTOS JR, 2009). O estágio "estável" garante que as principais funcionalidades foram testadas e funcionam. Além disso, um projeto com software em estável sinaliza para potenciais usuários e

colaboradores que ali se encontra uma comunidade ativa, que pode fornecer suporte e desenvolver funcionalidades adicionais. Mas apesar de esses dois estágios estarem dentro da área de efeito positivo em atratividade, software em estável é estatisticamente mais atrativo do que software em beta, tanto na perspectiva do usuário quanto na do desenvolvedor (Gráfico 4 e Tabela 1).

O estágio de desenvolvimento "maduro" garante que a maior parte das funcionalidades almejadas pelo software já foram desenvolvidas e testadas (SANTOS JR, 2009). Esse estágio sinaliza que, além de maduro, o software encontra-se em desenvolvimento. Então, potenciais usuários e colaboradores sabem que erros reportados serão verificados e funcionalidades requisitadas serão desenvolvidas. Esse estágio de desenvolvimento também é visto positivamente por organizações, o que pode aumentar as chances do projeto receber financiamento.

DESENVOLVENDO ESTRATÉGIAS PARA ADMINISTRAÇÃO DA ATRATIVIDADE

Este artigo tem dois propósitos principais. O primeiro deles é apresentar o conceito de atratividade de projetos de software livre para possibilitar a observação, o estudo e a administração desse atributo determinante do sucesso dos projetos. O segundo é discutir de que forma diversas características dos PSL, muitas vezes escolhidas sem o devido cuidado, podem influenciar sua adoção e seu aprimoramento por contribuidores externos, potencialmente minimizando sua utilidade e qualidade no futuro.

Com esses dois propósitos satisfeitos, espera-se ter desenvolvido informações úteis na priorização de recursos para a abertura de código-fonte na internet, pois determinados projetos têm mais chance de sucesso do que outros. Adicionalmente, demonstrou-se que o momento de disponibilizar um software – seu estágio de desenvolvimento – também influencia o interesse que ele despertará em desenvolvedores e usuários de software livre. E, finalmente, argumentou-se que a decisão do tipo de licenciamento adotado deve ser criteriosa e baseada nas expectativas futuras de utilização do software. Assim, munidos com essas informações, idealizadores e criadores de PSL podem atuar como estrategistas, de maneira a interagir de modo mais eficaz com a comunidade de software livre, produzindo bens públicos mais interessantes e com maior chance de serem aprimorados por voluntários, ficando assim disponíveis para a população em geral.

Obviamente, não existe uma estratégia ideal e geral para criar projetos de software livre de sucesso. O desenvolvimento de uma estratégia bem-sucedida do gênero depende dos objetivos futuros a serem perseguidos pelos criadores do software e da responsabilidade em manter o modelo livre de desenvolvimento sustentável. Mas os resultados reportados aqui possibilitam entender de que forma algumas decisões que definem as características de um projeto de software livre afetam variáveis cruciais para o sucesso da estratégia de liberar código-fonte na Internet. As conclusões aqui apresentadas, quando adaptadas à realidade de um projeto específico, podem ser úteis aos seus idealizadores, que antes estavam em desvantagem por não disporem dessas informações estratégicas.

Finalmente, é necessário dizer que a análise deste artigo possui limitações. Primeiramente, apesar de a amostra utilizada ser grande, ela é restrita aos projetos de um único repositório. Nem todos os PSL estão no SourceForge.net e, portanto, amostras alternativas devem ser utilizadas em estudos futuros para averiguar a consistência dos resultados. Em segundo lugar, estatisticamente, as análises não levaram em consideração as diferentes categorias de projetos na presença das demais. Por exemplo, os efeitos da licença foram verificados de forma independente dos efeitos do estágio de desenvolvimento. Apesar dos efeitos dessas características serem aparentemente independentes, estudos futuros devem averiguar se os resultados aqui reportados são mantidos em análises que consideram todas as características ao mesmo tempo. Por último, outros influenciadores de atratividade existem e estudos futuros devem buscar identificá-los. Em especial, o papel da usabilidade de um software em sua taxa de adoção já foi identificado e pesquisas futuras de atratividade devem buscar incorporá-lo (RAJANEN e IIVARI, 2010).

NOTA DE AGRADECIMENTO

Agradeço, primeiramente, a FAPESP (Processo: 2009/02046-2) pelo apoio financeiro; o Prof. Fabio Kon, o doutorando Paulo Meirelles e o bolsista Marcos Bonci Cavalca pelo constante apoio, disponibilidade e parceria durante a realização desta pesquisa no Centro de Competência em Software Livre (CCSL) do IME-USP.

REFERÊNCIAS

Agerfalk, P. J; Fitzgerald, B. Outsourcing to an unknown workforce: exploring opensourcing as a global sourcing strategy. *MIS Quarterly*, v. 32, n. 2, p. 385-409, 2008.

BONACCORSI, A; GIANNANGELI, S; ROSSI, C. Entry strategies under competing standards: hybrid business models in the open source software industry. *Management Science*, v. 52, n. 7, p 1085-1098, 2006.

CROWSTON, K; ANNABI, H; HOWISON, J; MASANGO, C. Towards a portfolio of FLOSS project success measures. 26th International Conference on Software Engineering, *Proceedings*. Edinburgh, UK, 2005.

CROWSTON, K; HOWISON, J. Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology & Policy*, v. 18, n. 4, p 65-85, 2006.

ECONOMIDES, N; KATSAMAKAS, E. Two-sided competition of proprietary vs. open source technology platforms and the implications for the software industry. *Management Science*, v. 52, n. 7, p 1057-1071, 2006.

FARHOOMAND, A. Opening up of the software industry: the case of SAP. *Communications of the Association for Information Systems*, v. 20, n. 1, p 800-811, 2007.

FITZGERALD, B. The transformation of open source software. *MIS Quarterly*, v. 30, n. 3, p 587-598, 2006.

GREWAL, R; LILIEN, G. L; MALLAPRAGADA, G. Location, location, location: how network embeddedness affects project success in open source systems. *Management Science*, v. 52, n. 7, p 1043-1056, 2006.

HOWISON, J; CONKLIN, M; CROWSTON, K. FLOSSmole: a collaborative repository for FLOSS research data and analyses. *International Journal of Information Technology and Web Engineering*, v. 1, n. 3, p. 17–26, 2006.

KOCH, S. Profiling an open source project ecology and its programmers. *Electronics Markets*. Special Section: Open Source Software, v. 14, n. 2, 2004.

KUK, G. Strategic interaction and knowledge sharing in the KDE developer mailing list. *Management Science*, v. 52, n. 7, p 1031-1042, 2006.

MACCORMACK, A; RUSNAK, J; BALDWIN, C. Y. Exploring the structure of complex software designs: an empirical study of open source and proprietary code. *Management Science*, v. 52, n. 7, p 1015-1030, 2006.

MOCKUS, A; FIELDING, R. T; HERBSLEB, J. A case study of open source software development: the Apache server. 22nd International Conference on Software Engineering. *Proceedings*, 2000.

O'MAHONY, S. Guarding the commons: how community managed software projects protect their work. *Research Policy*, v. 32, n. 7, p 1179, 2003.

O'MAHONY, S. The governance of open source initiatives: what does it mean to be community managed? *Journal of Management & Governance*, v. 11, n. 2, p 139-150, 2007.

RAJA, U; TRETTER, M. *Investigating open source project success*: a data mining approach to model formulation, validation and testing. Working Paper. Texas A&M University, College Station, Texas, 2006.

RAJANEN, M; IIVARI, N. Traditional usability costs and benefits: fitting them into open source software Development. 18th European Conference on Information Systems (ECIS). *Proceedings.* Pretoria, South Africa, 2010.

RAYMOND, E. S. *The cathedral and the bazaar: musings on Linux and open source by an accidental revolutionary*. Sebastopol, CA: O'Reilly Press, 1999.

ROBERTS, J. A; HANN, I.-H; SLAUGHTER, S. A. Understanding the motivations, participation, and performance of open source software developers: a longitudinal study of the Apache projects. *Management Science*, v. 52, n. 7, p 984-999, 2006.

SABINO, V; KON, F. Licenças de software livre história e características. *Relatório Técnico RT-MAC-IME-USP, 2009*. Disponível em: http://ccsl.ime.usp.br/files/relatorio-licencas.pdf. Acessado 01/11/2010.

SANTOS JR, C. Understanding partnerships between corporations and the open source community: a research gap. *IEEE Software*, v. 25, n. 6, 2008.

SANTOS JR, C. *Open source software projects' attractiveness, activeness, and efficiency as a path to software quality*: an empirical evaluation of their relationships and causes. Department of Management Information Systems at SIUC, Carbondale, IL, 2009. Disponível em: http://opensiuc.lib.siu.edu/dissertations/2. Acessado 01/11/2010.

SANTOS JR, C; GONÇALVES, M. Análise da substituição de um software proprietário por um software livre sob a ótica do custo total de propriedade: estudo de caso do setor de peças automobilísticas. *Revista Contemporânea de Contabilidade*, v. 1, n. 6, p 39-60, 2006.

SANTOS JR, C; GONÇALVES, M. A resource-based explanation for the Apache consistent dominance in the web-server industry. In: ENCONTRO ANUAL DA ASSOCIAÇÃO NACIONAL DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ADMINISTRAÇÃO, 32, *Anais Eletrônicos*. Rio de Janeiro: EnANPAD, 2008.

SHAH, S. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, v. 52, n. 7, p 1000-1014, 2006.

STEWART, K; GOSAIN, S. The impact of ideology on effectiveness in open source software development teams. *MIS Quarterly*, v. 30, n. 2, p 291-314, 2006.

VON HIPPEL, E. *Democratizing innovation*. Boston, MA: MIT Press, 2005.

VON HIPPEL, E; VON KROGH, G. Open source software and the "private-collective" innovation model: issues for organization science. *Organization Science*, v. 14, n. 2, 2003.

WATSON, R. T; BOUDREAU, M.-C.; YORK, P. T; GREINER, M. E; WYNN JR, D. The business of open source. *Communications of the ACM*, v. 51, p. 41-46, 2008.

WEST, M; ANDERSON, N. Innovation in top management teams. *Journal of Applied Psychology*, v. 81, n. 6, p. 680-693, 1996.