UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS APLICADAS A EDUCAÇÃO
DEPARTAMENTO DE CIÊNCIAS EXATAS
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

GABRIELA FERNANDES PIMENTA

Análise de Interação entre Colaboradores em Projeto de Código Aberto

RIO TINTO – PB
2014

GABRIELA FERNANDES PIMENTA

Análise de Interação entre Colaboradores em Projeto de Código Aberto

Monografia apresentada ao Curso de Bacharelado em Sistemas da Informação da Universidade Federal da Paraíba, como pré-requisito para obtenção do grau de Bacharel em Sistemas da Informação, sob orientação do Prof. Me Yuri Malheiros.

RIO TINTO – PB

2014

P644a Pimenta, Gabriela Fernandes.

Análise de interação entre colaboradores em projeto de código aberto / Gabriela Fernandes Pimenta. – Rio Tinto: [s.n.], 2014.

52 f.: il. –

Orientador: Prof. Me. Yuri Malheiros.
Monografia (Graduação) – UFPB/CCAE.

1. Software - desenvolvimento. 2. Software – desenvolvimento em colaboração. 3. Software - código aberto.

UFPB/BS-CCAE *CDU: 004.4(043.2)*

GABRIELA FERNANDES PIMENTA

Análise de Interação entre Colaboradores em Projeto de Código Aberto

Monografia apresentada ao Curso de Bacharelado em Sistemas da Informação da Universidade Federal da Paraíba - UFPB, como pré-requisito para obtenção do grau de Bacharel em Sistemas da Informação, apreciada pela Banca Examinadora composta pelos seguintes membros:

Aprovada em: _____/_____/ 2014

BANCA EXAMINADORA

Orientador: Prof. Me Yuri Malheiros
Universidade Federal da Paraíba – Campus IV

Prof. Me. Marcus Williams Aquino de Carvalho
Universidade Federal da Paraíba – Campus IV

Prof. Me. Renata Viegas de Figueiredo
Universidade Federal da Paraíba – Campus IV

Senhor, guarda-me como a menina dos teus olhos, esconde-me debaixo da sombra das tuas asas, Dos ímpios que me oprimem, dos meus inimigos mortais que me andam cercando.

Salmo 17:8-9

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus por estar onde estou hoje e por estar realizando este sonho.

A meus pais, Marcos Pimenta e Maria de Fátima, e a minha irmã Amanda Pimenta e meus irmãos Marcos Pimenta e João Neto, que me deram força, acreditaram em mim e batalharam para que eu concluísse meu curso, e mesmo distantes, foram essenciais para minha chegada até aqui.

Ao meu namorado Ednaldo Onofre por todo o carinho e compreensão, principalmente nos últimos meses em que estive mais ausente devido ao estágio e aos trabalhos da universidade.

Agradeço a todos os meus colegas de turma, pois, foi uma honra conhecer a todos, e hoje posso dizer que fazem parte da minha história e serão lembrados sempre.

Agradeço todos os professores que tive a honra de conhecer e de ser aluna, que compartilharam seu conhecimento e que contribuíram com minha formação durante estes anos na universidade.

Ao meu orientador deste trabalho, Yuri Malheiros por ter acreditado em mim e ter aceitado ser meu orientador, pelo seu exemplo de dinamismo e trabalho que é a maior lição que um professor pode dar a seu aluno, gostaria de agradecê-lo por ter acreditado na minha caminhada.

Aos examinadores da banca, Renata e Marcus, pela ajuda dada após a pré banca que fez com que esta pesquisa se tornasse melhor

Considerando esta monografia como resultado de uma caminhada que não começou na UFPB, agradecer pode não ser tarefa fácil, nem justa. Para não correr o risco de injustiça, agradeço de antemão a todos que de alguma forma passaram pela minha vida e contribuíram para a construção de quem sou hoje.

RESUMO

Projetos de *software* com código aberto estão ganhando colaboradores gradualmente e sendo cada vez mais utilizados. Entretanto, o sucesso com projetos desse tipo que possuem grande porte são bastante dependentes da eficiência das interações entre os participantes, quanto mais centralizada a comunicação, maior a probabilidade de o projeto desestabilizar. Baseado nisso, é necessário que sejam analisadas as interações entre os colaboradores de tal projeto para que seja possível o gerenciamento do mesmo de forma eficaz. Desta forma, este trabalho visa analisar a rede social do projeto Rails, um dos projetos de código aberto mais populares do repositório GitHub. A análise utilizou métricas para medir a modularidade, de forma a identificar comunidades no projeto e de grau de centralidade para identificar os colaboradores mais participativos. Os resultados obtidos nesse trabalho possibilitam um melhor gerenciamento do projeto e maiores disseminação de conhecimento e possibilidade de sucesso.

Palavras-Chave: Projeto de *Software*. Código Aberto. Projeto *Rails*. Rede Social.

ABSTRACT

Software projects with open source are being increasingly used and gradually gaining employees. However, to succeed with such projects that have large scale are quite dependent on the efficiency of the interactions between the participants, the more centralized communication, the greater the likelihood of the project to destabilize. Based on that, the interactions among employees of such project must be analyzed, so it is able to do the project management effectively. Thus, this study aims to analyze the social network of the Rails project, one of the most popular open source projects on Github repository. The analysis used metrics to measure the modularity, in order to identify communities in the design, and the degree of centrality to identify collaborators more participatory. The results obtained in this work allow a better project management, greater dissemination of knowledge and possibility of success.

Keywords: Software Project. Open Source. Rails project. Social Network.

LISTA DE ILUSTRAÇÕES

FIGURA 2: Áreas de operação do Git..... 26
FIGURA 3: Exemplo de interface do Gephi 29
FIGURA 4: A estrutura de equipe de desenvolvimento FLOSS..... 32
FIGURA 5: A estrutura centralizada na esquerda e descentralizada na direita..... 33
FIGURA 6: Modelo de projeto Altamente centralizado 34
FIGURA 7: Modelo de projeto descentralizado..... 34
FIGURA 8: Fluxo de Desenvolvimento do Trabalho 36
FIGURA 8: Gráfico da distribuição dos Graus dos atores na rede 40
FIGURA 9: Modelo do arquivo com as informações das interações no GitHub..... 41
FIGURA 10: Imagem do arquivo sem o tratamento para facilitar interpretação 42
Figura 11: Rede das interações entre os colaboradores..... 43
Figura 12: GRAFO DAS COMUNIDADES DA REDE 44
Figura 13: Distribuição do número de comunidades dos atores..... 45
Figura 14: Interação entre as cinco maiores comunicades 46
Figura 15: rafaelfranca..... 47
Figura 16: senny 47
Figura 17: pixeltrix 47
Figura 18: steveklabnik 47
Figura 19: matthewd..... 47

LISTA DE TABELAS

TABELA 1: Vantagens dos softwares livre 18
TABELA 2: Software livre X Open Source 22
Tabela 3: Comunidades mais relevantes 45
Tabela 4: Principais colaboradores 48

LISTA DE ABREVIATURAS E SIGLAS

BSD - Berkeley Software Distribution

CA - Código Aberto

DDS - Desenvolvimento Distribuído de Software

ES - Engenharia de Software

FSF - Free Software Foundation

MIT - Massachusetts Institute of Technology

MPL - Mozilla Public License

SL - Software Livre

TI – Tecnologia da Informação

SUMÁRIO

1 INTRODUÇÃO 14
1.1 OBJETIVOS 15
1.1.1 Objetivo Geral 15
1.1.2 Objetivo Específico 15
1.2 ESTRUTURA DO TRABALHO 15
2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 SOFTWARE LIVRE 17
2.1.1 Vantagens e Desvantagens 18
2.2 LICENÇAS FREQUENTEMENTE UTILIZADAS 19
2.2.1 GPL – Licença Pública Geral GNU 19
2.2.2 Licença BSD 20
2.2.3 A Licença MIT 20
2.2.4 Licença Apache 20
2.2.5 Licença Mozilla 20
2.2.6 OPENSOURCE 21
2.3 SOFTWARE LIVRE X OPEN SOURCE 22
2.4 DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE (DDS) 23
2.5 GIT 24
2.5.1 ÁREAS DE OPERAÇÃO DO GIT 25
2.5.2 GITHUB 26
2.6 REDES SOCIAIS 27
2.7 GEPHI 29
3 A ESTRUTURA SOCIAL DO DESENVOLVIMENTO DE SOFTWARE LIVRE E DE CÓDIGO ABERTO 31
4 METODOLOGIA 36
4.1 SOBRE O ESTUDO DO EXPERIMENTO 36

4.1.1 Revisão Bibliográfica de caráter exploratório 36

4.1.2 Estudo do Experimento do Projeto Rails 36

4.1.3 Coleta de Dados do Projeto Rails 37

4.1.4 Análise das Métricas do Projeto Rails 37

5 ANÁLISE DE INTERAÇÃO ENTRE COLABORADORES EM PROJETO DE CÓDIGO ABERTO 39

5.1 MÉTRICAS DE ANÁLISE DE REDES SOCIAIS 39

5.2 GERAÇÃO DE GRAFOS NA REDE GEPHI 41

6 CONCLUSÃO E SUGESTÕES DE TRABALHOS FUTUROS 49

REFERÊNCIAS 50

1 INTRODUÇÃO

Observa-se que os softwares de código aberto, atualmente, estão sendo cada vez mais utilizados por empresas de diferentes tipos e tamanhos. Diante disso, faz-se necessário compreender alguns aspectos relacionados à sua dinâmica de desenvolvimento e utilização. Pode-se destacar como ponto fundamental, a interação entre os colaboradores dentro de tais projetos.

Especialmente em sistemas de grande porte, um desenvolvimento otimizado depende da eficiência dessas interações, as quais, quando centralizadas em poucos colaboradores, centralizam também a informação e o conhecimento. Isto acaba por quebrar o pensamento que um software de código aberto com muitos usuários tem necessariamente muitos colaboradores importantes. Assim, é essencial analisar e entender o nível de interação entre os colaboradores de projetos de código aberto, bem como a distribuição das relações de colaboração entre os participantes, de tal forma a proporcionar o gerenciamento de projetos de código livre eficaz, garantindo-se a qualidade do projeto, diminuindo o número de bugs, facilitando a manutenção e reuso do código, e consequentemente a gerência de configuração e mudança.

Diante disso, o presente trabalho tem por finalidade realizar uma análise aprofundada das interações entre colaboradores em um projeto específico de código aberto. Para realizar essa análise, foi feito um estudo do experimento com o projeto Rails[1], disponibilizado no GitHub[2].

O projeto Rails é um framework que auxilia o desenvolvimento de sistemas web. Seu código fica disponível no GitHub, que é uma ferramenta na web que hospeda projetos utilizando o sistema de controle de versão Git[3].

Para atingir a análise acima, foi necessário seguir algumas etapas metodológicas. Foram extraídos, inicialmente, dados obtidos através da própria API do GitHub, foram coletados os autores de todas *as Issues*[4] e os *pull requests* com o estado "open" e os autores dos comentários feitos em cada *Issue.* Em cima da análise desses

[1] O projeto Rails é um framework que auxilia o desenvolvimento de sistemas web no GitHub.
[2] O GitHub é um serviço gratuito que aloja projetos que usam o Git como controle de versão.
[3] O Git é um sistema de controle de versão.
[4] Issues*:* Cada colaborador que adiciona uma *"Issue"* no *GitHub* ganha um nó, os colaboradores que comentam uma "Issue" são interligados, inclusive o que cria a "Issue".
.

dados foram calculadas métricas, Modularidade e em seguida foi utilizada Centralidade Grau.

Uma vez extraídos os dados definidos anteriormente, com o auxílio da API do próprio GitHub através de uma script, foi feita uma análise e comparação com o modelo Onion. Para isso foram gerados grafos, onde os nós desses grafos representam os colaboradores do projeto, e as arestas denotam essas interações.

Segundo Crowston; Howison (2013), o estudo do modelo Onion, demonstra um determinado padrão de centralização ou descentralização de comunicação entre as equipes. O artigo faz uma análise examinando 120 equipes de projeto do *SourceForge,* o que representa uma ampla gama de tipos de projeto de Software Livre, por sua centralização das comunicações, revelada nas interações no sistema de rastreamento de *bugs*. Os *bugs SorceForge* no modelo Onion são equivalentes as *Issues* do GitHub. No resultado descobriram-se informações sobre projetos *FLOSS* e características que podem ser melhoradas para auxiliar no gerenciamento dos mesmos.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Este trabalho tem por finalidade analisar a estrutura social da interação entre os colaboradores de projetos de código aberto através de dados disponíveis no GitHub.

1.1.2 Objetivo Específico

- Capturar dados das Issues e Pull requests de um projeto de código aberto;
- Construir grafos que representem as interações entre os colaboradores;
- Analisar estrutura de interações;
- Calcular métricas de redes sociais nos grafos de interações.

1.2 ESTRUTURA DO TRABALHO

Os assuntos tratados nesse trabalho estão organizados em seis capítulos.

No Segundo capítulo é apresentado o Referencial teórico do trabalho desenvolvido. Incluindo uma breve introdução sobre software livre, assuntos relacionados ao OpenSource, a utilização de software distribuído e as habilidades

necessárias para um bom desenvolvimento dos projetos, como o sistema de controle de versão (GIT) pode ser projetado para lidar com projetos muito grandes com rapidez e eficiência, a influência da redes sociais na sociedade, e a introdução ao Gephi, a ferramenta para manipulação de grafos mais utilizada no mercado.

No Terceiro capítulo é apresentado o trabalho relacionado que inspirou essa pesquisa.

No Quarto capítulo será apresentada a metodologia utilizada na pesquisa e o fluxo seguido, detalhando cada fase do desenvolvimento, as métricas utilizadas e a coleta de dados.

No Quinto capítulo será exposta a análise de interação entre colaboradores em projeto de código aberto, apresentando todo o processo do seu desenvolvimento. Serão relatados os resultados encontrados.

Logo após, no sexto e último capítulo serão fornecidas as considerações finais do trabalho e sugestões para trabalhos futuros

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SOFTWARE LIVRE

Nas décadas de 50 e 60, a ideia de vender software era muito remota, os software eram gratuitos, pois existiam poucos computadores no mundo e o valor real estava na máquina em si e não nos programas. Com o crescimento da informática, criação de empresas de TI e a demanda que surgiu com a disseminação dos computadores, fizeram com que a venda de softwares começassem a ser separada das vendas das máquinas. Dessa forma, apareceram as indústrias com foco em mecanismos de proteção de propriedade intelectual como direitos autorais e patentes, para se proteger da acirrada concorrência e garantir suas vantagens competitivas. (LUZ, 2007)

Segundo Campos (2006) o movimento que defende o uso dos softwares livre começou na década de 80 com o Projeto GNU fundado por Richard Stallman.

Podemos definir que Software Livre, ou Free Software, como um software que pode ser usado, copiado, estudado, modificado e redistribuído sem restrição. Software livre é aquele que mesmo sendo disseminado livremente possui também uma licença para sua utilização. (COSTA E SANTOS, 2014).

Pereira (2004) lembra que o Software Livre além de ser disponibilizado juntamente com seu código fonte sem precisar de autorização, qualquer pessoa que tenha acesso ao software pode corrigir ou modificar seu código fonte.

Um software só pode ser considerado livre se possuir quatro tipos de liberdade, são elas (CAMPOS, 2006):

Liberdade Nº 0 - A liberdade para executar o programa, para qualquer propósito;

Liberdade Nº 1 - A liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades. O acesso ao código-fonte é um pré-requisito para esta liberdade;

Liberdade Nº 2 - A liberdade de redistribuir, inclusive vender, cópias de modo que você possa ajudar ao seu próximo;

> Liberdade Nº 3 - A liberdade de modificar o programa, e liberar estas modificações, de modo que toda a comunidade se beneficie. O acesso ao código-fonte é um pré-requisito para esta liberdade;

Qualquer pessoa que tenha acesso a um Software Livre pode usá-lo de modo pessoal ou profissional. Mesmo o software sendo modificado não é necessário que seja informado a ninguém em particular ou de nenhum modo em especial que a modificação foi realizada. Ou seja, independente de qualquer tipo de pessoa, física ou jurídica, ou de Sistema Operacional, o Software livre pode ser usado para qualquer tipo de trabalho ou atividade.

É livre, pois representa liberdade e não gratuidade. Software Livre não é necessariamente Não-Comercial (SILVA, 2014). Qualquer tipo de serviço, como por exemplo, distribuições customizadas podem ser serviços pagos. Com esse novo modelo de mercado de Software, as empresas estão trabalhando com prestações de serviços ligados ao Software Livre. Podemos citar como serviços, treinamento, suporte e customização desses produtos.

Entretanto existem algumas regras que devem ser seguidas. Se o software foi modificado e essa sua extensão for comercializada, as modificações devem ser encaminhadas a seus desenvolvedores iniciais, conforme prega a Free Software Foundation[5].

2.1.1 Vantagens e Desvantagens

A utilização de software livre implica em uma série de vantagens e desvantagens. Na tabela 01 podemos analisar algumas vantagens de se utilizar o Software livre de acordo com as dimensões e aspectos que se destacam.

TABELA 1: VANTAGENS DOS SOFTWARES LIVRE
FONTE: Garcia et al (2010)

DIMENSÃO	ASPECTOS
Custo	O software livre aproveita equipamentos tidos como obsoletos; Pode ser gratuito para ser usado; Proporciona vantagens econômicas

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

Customização	É possível adaptar o programa conforme necessidades; É um produto com flexibilidade
Facilidade/praticidades	Verificam-se organizações virtuais. Existe liberdade de executar o programa e estudar como ele funciona; Possui sistema e aplicativos geralmente configuráveis.
Liberdade de ação	Pode ser aprimorado e personalizado. É permitido acesso ao código-fonte; Possibilita aperfeiçoar o programa; Oferece liberdade importante para a sociedade.
Segurança	Há menor vulnerabilidade a invasões e vírus
Social	Encontra-se o produto em permanente construção coletiva; Permite cooperação e compartilhamento.; Gera beneficio para toda a comunidade; Oferece contribuição à sociedade; Favorece a inclusão digital.

Como desvantagens podemos citar:

- Interface de usuário não é uniforme nos aplicativos;
- Instalação e configuração pode ser difícil;
- Mão de obra escassa e/ou custosa para desenvolvimento e/ou suporte.

2.2 LICENÇAS FREQUENTEMENTE UTILIZADAS

O software livre é baseado em licenças. Elas são como um contrato de adesão para a utilização daquele software. A partir da aceitação do usuário, o mesmo deverá cumprir com as regras estabelecidas pelo autor do software, caso não cumprimento destas regras o usuário poderá ser processado.

2.2.1 GPL – Licença Pública Geral GNU

Essa licença se aplica a maioria dos software livre existentes da FSF. A grande maioria das licenças de software tem como restringir a liberdade de uso e do compartilhamento. Já a GNU tem como objetivo garantir essa liberdade de compartilhar

e alterar softwares de livre, tornando-os assim livres de fato para qualquer usuário. (HEXSEL, 2002)

2.2.2 Licença BSD

De acordo com Sabino e Kon (2009) a BSD License foi a primeira licença de software livre escrita, e a mais usada até hoje. Foi criada na Universidade da Califórnia em Berkeley para o sistema operacional da faculdade originado do UNIX chamado então de Berkeley Software Distribution. Esta licença é usado em muitos softwares licenciados de modo permissivo.

2.2.3 A Licença MIT

A MIT, também conhecida como licença X11 ou X, é uma licença permissiva assim como a BSD. Ou seja, não existe proibição sobre o que pode ser feito com o software e seu código-fonte, desde que o portador do software siga a única regra dela: "O aviso de copyright acima e esta permissão deverá ser incluído em todas as cópias ou partes substanciais do Software." (SABINO E KON, 2009)

2.2.4 Licença Apache

Essa licença é utilizada por um dos projetos mais conhecidos de software livre, o servidor web apache. Assim como a licença MIT e BSD, também é uma licença permissiva, porém dá ênfase à proteção da criação da marca apache (SABINO E KON, 2009).

2.2.5 Licença Mozilla

A Mozilla Public License, ou MPL, serviu como modelo para muitas licenças de software livre comerciais existentes no mercado.

As suas regras são bem elaboradas e claras: o código que é coberto pela licença deve ser redistribuído pelos termos da licença Mozilla, mas também pode ser utilizado em trabalhos maiores, que podem estar sob outra licença (SABINO E KON, 2009).

Nesta seção vimos que um Software Livre pode ser usado, copiado, estudado, modificado e redistribuído sem restrição a utilização. Conhecemos também as maiores vantagens de utiliza-lo e as licenças mais utilizadas no mercado.

No próxima seção veremos o que é OpenSource e se existe diferença ou relação com a utilização de Software Livre.

2.2.6 OPENSOURCE

Em 1989, o Richard Stallman, fundador do movimento software livre, do projeto GNU e da FSF, criou um tipo de licença de software chamada "copyleft", hoje mais conhecida como "open source" (SANT'ANNA,2014).

Um software que faz uso de uma destas licenças requer que este obedeça alguns princípios, são eles: (PEREIRA et. al, 2014)

1. Distribuição livre. A licença não deve restringir ninguém de dar ou vender o software bem como distribui-lo numa distribuição de software de várias fontes.

2. O código fonte tem que estar incluído ou ter acesso gratuito. Facilmente obtido de forma clara.

3. Trabalhos derivados. As modificações e redistribuições do software têm de ser

permitidas bem como a possível redistribuição do mesmo sobre a mesma licença

4. Integridade do autor do código fonte. A licença pode restringir o código de ser

distribuído modificado apenas se as mesmas licenças permitirem que as modificações sejam redistribuídas como patches.

5. Sem descriminação contra pessoas ou grupos. Ninguém pode ser bloqueado ao uso do software.

6. Sem descriminações contra grupos de trabalho. A sua utilização não pode ser vedada a nenhum tipo de fim como por exemplo o seu uso num negócio ou numa pesquisa genética.

7. Distribuição da licença. A licença que vai com o software tem que se referir ao programa todo sem a necessidade de licenças adicionais por outros grupos.

8. Licença não deve ser específica de um produto. A licença do programa não pode ser dependente do facto do mesmo fazer parte de uma distribuição particular de software.

9. Licença não pode ser restritiva a outro software. Esta não pode obrigar que outro software com o qual é incluída, sejam open source por exemplo.

10. Licença deve ser tecnologicamente neutra. Não devem ser obrigatórias formas específicas de aceitar a licença.

2.3 SOFTWARE LIVRE X OPEN SOURCE

Para Filho et al (2005) os princípios do software livre (SL) quanto ao de código aberto (CA), fundamentam-se nas premissas básicas que são a de liberdade de expressão, acesso à informação e do caráter eminentemente coletivo do conhecimento, o software é mais uma forma de representar e organizar o conhecimento, ou seja, é um bem comum que deve ser construído e disponibilizado democraticamente, e não privatizado. Sua difusão e uso devem ser livres.

Para Campos (2006) o movimento Free Software e o movimento Open Source são como dois campos políticos dentro da comunidade de software livre, que mesmo discordando dos princípios básicos, concordam (mais ou menos) nas recomendações práticas. Dessa forma eles trabalharem juntos em diversos projetos específicos. Podemos disser que os dois movimentos são parceiros e não competitivos.

Apesar de ambos terem o foco em desenvolvimento e distribuição de software, podem o encontrar diferenças entre eles podemos, como por exemplo, podemos analisar a Tabela 02.

TABELA 2: SOFTWARE LIVRE X OPEN SOURCE
FONTE:SOUSA et. al., 2011

SOFTWARE LIVRE	OPEN SOURCE
Não diz respeito à gratuidade, mas sim à liberdade (executar, estudar, redistribuir, aperfeiçoar)	Permissão de trabalhos derivados (código pode ser usado em software proprietário)
Garantia e perpetuação das liberdades	Ligadas a questões práticas de produção e negócio
Existe o efeito “contaminação”	Não há o efeito “contaminação”

Nesta subseção conhecemos que OpenSource nada mais é, que um tipo de licença de software e que mantem uma relação com software livre, pois, ambos pertencem ao domínio de softwares não proprietários.

No próximo capitulo abordaremos a necessidade de utilizar desenvolvimento distribuído de software em projetos.

2.4 DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE (DDS)

Os investimentos em empresas de desenvolvimento de software vem crescendo bastante, criando novas formas de colaboração e competição entre essas empresas. (HERBSLEB et. al., 2001). Com o aumento dessas demandas, algumas crises começaram a surgir nessas empresas: falhas em projetos e falta de recursos com as habilidades necessárias para um produto de qualidade. Assim, o Desenvolvimento Distribuído de Software (DDS) surgiu como sendo uma oportunidade de minimizar essas crises encontradas com o crescimento de demandas de softwares. Alguns fatores contribuíram para acelerar este cenário. Entre eles, podemos citar (CARMEL, 1999):

- A necessidade de recursos globais para serem utilizados a qualquer hora;
- As vantagens de estar perto do mercado local, incluindo o conhecimento dos clientes e as condições locais para explorar as oportunidades de mercado;
- A grande pressão para o desenvolvimento time-to-market, utilizando as vantagens proporcionadas pelo fuso horário diferente, no desenvolvimento conhecido como follow-the-sun (24 horas contínuas, contando com as equipes fisicamente distantes).

Podemos disser que o DDS tem como objetivo a colaboração e cooperação entre departamentos de organizações e pela criação de grupos de desenvolvedores que trabalham em conjunto, podendo até estarem localizados em cidades ou países diferentes. Segundo Carmel (1999), qualquer profissional que trabalhe com engenharia de software, sendo com o desenvolvimento de software tradicional quanto o distribuído, encontra várias dificuldades no desenvolvimento. As principais características que os diferenciam são:

- dispersão geográfica (distância física);
- dispersão temporal (diferenças de fuso-horário);

- diferenças culturais (idioma, tradições, costumes, normas e comportamento).

Essas diferenças levam em consideração a existência de três atores principais no processo de desenvolvimento de software, tendo papéis distintos uns dos outros, cada um com a sua importância. Esses atores são (PRIKLADNICK, 2002):

- Equipe de desenvolvimento: representa todas as pessoas envolvidas no desenvolvimento de um determinado projeto, podendo também ser formada por um conjunto de sub-equipes. Esta equipe pode envolver pessoas responsáveis pela área de negócios, gerência de projetos, desenvolvimento, testes, controle de qualidade, responsáveis pelo suporte de ferramentas dentro do projeto, entre outros.
- Cliente: é a pessoa física ou jurídica que solicitou e contratou o desenvolvimento de um determinado projeto.
- Usuário: representa as pessoas responsáveis por fornecer todas as informações necessárias (requisitos) para o correto desenvolvimento do projeto e são responsáveis por utilizar o produto gerado.

Segundo Damian (2006), embora o mercado já tenha ciência dos problemas e possíveis soluções usando o DDS, ainda necessita-se de um maior entendimento a ser alcançado a cerca do assunto. Como também o amadurecimento de novos métodos, técnicas e práticas.

O modelo de desenvolvimento distribuído é muito comum em projetos OpenSource, já que o código fonte ficará aberto podendo ser lido e alterado pela comunidade.

Nesta subseção percebemos que usar a técnica de desenvolvimento distribuído de software é sinônimo de obter um projeto colaborativo e cooperativo entre departamentos de organizações e pela criação de grupos de desenvolvedores que trabalham em conjunto ou em localidades diferentes.

Na próxima subseção conheceremos o que é a ferramenta GIT e quais são suas funcionalidades.

2.5 GIT

Desenvolvido por Linus Torvalds (criador do Linux), inicialmente devido a necessidade de ter um software capaz de controlar a versão do kernel do linux, o Git é um sistema de controle de versão, com foco em facilitar a vida de quem executa projetos em equipe. Ele permite que duas ou mais pessoas trabalhem juntas, como também pode ser utilizado para quem trabalha sozinho, devido a sua possibilidade de controlar as versões do projeto.

Os diretórios de trabalho do Git funcionam como um repositório de informação, registando todas as alterações e versões do software que está sendo utilizado. A maioria dos controles de versão guardam as mudanças do código como alterações de um determinado arquivo. Ou seja, a cada mudança no arquivo, o sistema guarda apenas as mudanças e não guarda o arquivo inteiro.

O Git é multiplataforma e funciona nos sistemas operacionais Windows, MacOS e Linux sem alterar o desempenho mantendo-se eficiente em todos eles.

O Git trata os dados como snapshots. Toda vez que realizamos uma operação de commit ou salvamos o estado do projeto no Git, o Git salva um snapshot de como todos os arquivos presentes no momento e salva a referência desse estado. Os arquivos que não foram modificados, não serão salvos na nova versão, o Git cria um link para a versão anterior idêntica que já foi guardada em outro momento (EIS, 2012).

O Git é um projeto de código aberto coberto pela Licença Pública Geral GNU v2. Foi projetado e escrito originalmente por Linus Torvalds e atualmente é mantido por Junio C. Hamano.

Podemos citar como vantagens para se utilizar o Git (GAMA,2014):

- Branches facilitados e independentes
- Merges facilitados - Mais rápido
- Distribuído
- Pode ocupar menos espaço que um checkout SVN
- Integração com o GitHub

2.5.1 ÁREAS DE OPERAÇÃO DO GIT

Os locais de operação são as áreas onde os arquivos irão transitar enquanto estão sendo editados e modificados. São elas: *Working Directory, Stage Area, Git directory* (Figura 02) (EIS, 2012).

No *Git Directory* são salvos os dados e objetos do projeto. Ele é o diretório mais importante do Git e é ele que será copiado quando alguém clonar a versão do projeto, entenda-se clonar como sendo a cópia do projeto para a maquina.

O *Work Direct*ory é o local de trabalho, onde os arquivos podem ser usados e alterados. Em outras palavras, o work directory é a pasta de arquivos dos projeto.

Quando é feita uma alteração no arquivo, ele vai para o Staging Area, que pode ser chamada de área intermediária. A *Staging Area* contém o *Git Directory* juntamente com os arquivos modificados, onde ele guarda as informações sobre o que vai no próximo commit.

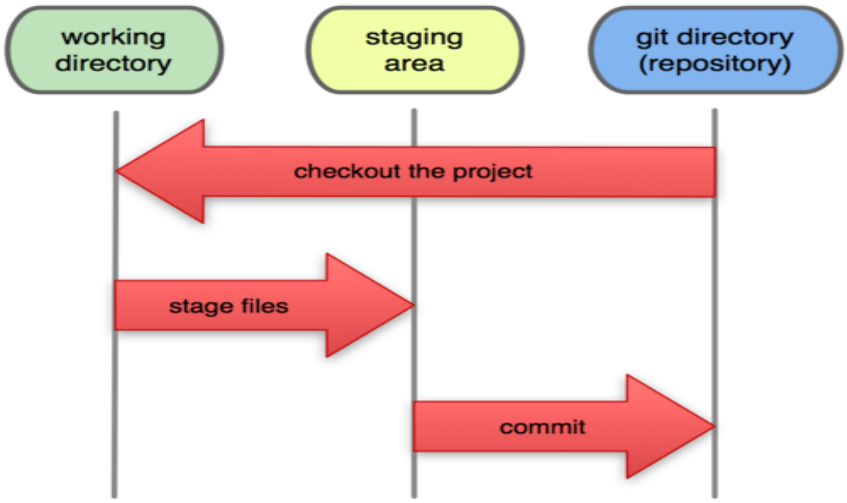


FIGURA 1: ÁREAS DE OPERAÇÃO DO GIT
FONTE: EIS (2012)

2.5.2 GITHUB

O GitHub é um serviço gratuito de *Web Hosting* que aloja projetos que usam o Git como controle de versão. Escrito em Ruby on Rails pelos desenvolvedores da Logical Awesome (Chris Wanstrath, PJ Hyett e Tom Preston), o GitHub é uma plataforma colaborativa é bastante fácil de partilhar, dessa forma pode ser melhorado por outras pessoas da equipe. Nele também conseguimos gerir o código fonte dos projetos que são desenvolvidos. (PINTO, 2013).

Podemos criar vários repositórios no GitHub, sendo eles públicos ou privados. Dessa forma os outros membros da equipe consegue baixar, modificar e receber atualizações dos projetos.

Em cada repositório, o GitHub conta também com a *Issue* trazer, um gerenciador de problemas, onde os colaboradores se interagem. De forma a ser uma maneira de manter o controle de tarefas, melhorias e bugs para os projetos.

Nesta subseção conhecemos um sistema de controle de versão, chamado Git, que lida com bastante eficiência com gerenciamento de código fonte distribuído. Vimos como podemos trabalhar o GitHub com integração com o Git.

Na próxima subseção iremos reconhecer o que é uma rede social e como foi realizado análise das propriedades especificas existentes.

2.6 REDES SOCIAIS

Segundo Tamaél (2005), todos estão inclusos na sociedade por meio de alguma relação que desenvolveram durante suas vidas, seja no ambiente familiar, como na época escolar, como também na comunidade que vive e no ambiente de trabalho. A própria natureza humana nos liga a outras pessoas e estrutura a sociedade em rede

Analisar comportamentos e opiniões das pessoas depende da estrutura social o qual ele se insere, ou seja, um conjunto de relações que esse cidadão estabelece a partir de sua interação com os outros, e não os atributos individuais, como por exemplo, classe, sexo, idade e gênero. Essa análise de redes forma um novo paradigma no estudo sobre a estrutural social. Podemos falar que essa estrutura social compreende a rede de relações e de limitações que mede as escolhas, orientações, comportamentos e opiniões de cada pessoa.

Com base em sua atividade, as redes sociais existentes em habito empresarial trabalham em torno de disseminar informações e conhecimentos. Sabemos que espaços organizacionais podem ser tanto presenciais quanto virtuais, porém as pessoas com os mesmos objetivos trocam experiências, criando bases e gerando informações relevantes para o setor em que atuam.

A rede social no trabalho pode ocorrer de várias formas, desde uma conversa informal, um encontro depois do expediente do trabalho, uma reunião, congressos, como também uma situação formal.

Macedo (1999), subdividi essas redes em redes de confiança, redes de trabalho ou consulta e redes de comunicação.

- Redes de confiança: é aquela que compartilha "informações secretas" a um número determinado de pessoas autorizadas.
- Redes de trabalho ou consulta: é aquela que usa estruturas informais e proporciona contato entre as pessoas que possuem informação que facilite o trabalho.
- Redes de comunicação: é aquela que ajuda a troca de informações de trabalho. É chamada de "amizades de escritórios".

Muitos sociólogos relatavam que as unidades básicas dessas redes sociais eram as díades, ou seja, a estrutura de relação entre pessoas seria a menor. Dessa forma, as relações entre as pessoas que formavam um grupo se daria de forma quase aleatória (WELLMAN, 1999:31).

Uma outra unidade básica seria as tríades, de formato triangular. Este modelo relata que entre duas pessoas, existe uma pessoa em comum. Ou seja, as duas pessoas têm mais possibilidade de se conhecerem e de fazerem parte de uma mesma rede social.

Sabemos que as redes possuem propriedades especificas e que essas propriedades estão presentes na análise de redes sócias. Segundo Scott (2000), podemos destacar algumas formas de analisar as propriedades dessas redes, são elas:

- Grau de conexão: é a quantidade de conexões que um nó possui. Podemos falar que dois nós que são conectados entre si são chamados de adjacentes e os nós aos quais um determinado nó é adjacente são denominados vizinhança. O grau de conexão entre os nós se dá pela quantidade de conexões que possui na rede. Quanto maior o grau de conexão mais central é o nó na rede.
- Densidade: é a medida que informa o grau de conexão de uma rede. Como explica SCOTT (2000, p.69), o conceito descreve o nível geral de conexão entre os pontos de um grafo. Portanto, uma extensão do número de conexões de um grafo em relação ao número máximo de conexões que o mesmo grafo suporta. Ou seja, a densidade se dá pela quantidade de conexões que grafo possui.
- Rede hierárquica e modular: é a relação de lei de potência entre o coeficiente de agregação de um nó e o seu grau. Isso implica que os nós distantes são partes de

áreas de alta agregação e que a comunicação entre estas áreas (também chamadas módulos) é feita por um pequeno número de nós.

Neste capitulo vislumbramos o quanto a análise de redes estabelece um novo paradigma na pesquisa sobre a estrutural social.

No próximo capítulo iremos conhecer a ferramenta Gephi e como é seu funcionamento em redes sociais.

2.7 GEPHI

O desenvolvimento de novas formas de visualização de informações tem sido uma das áreas mais ativas nas humanidades digitais. Ferramentas de grafos para visualizar redes vêm se destacando em os projetos voltados para a manipulação de dados históricos, espaciais e textuais. (CLARA,2013).

O Gephi é uma das ferramentas para manipulação de grafos mais utilizada no mercado. É um software livre colaborativo mantido por um consórcio sediado na França, com inúmeras aplicações em áreas como as ciências biológicas ou a economia.

Na figura 03, percebemos que o Gephi é uma plataforma interativa de visualização e exploração de todos os tipos de redes e sistemas complexos, grafos dinâmicos e hierárquicos (MURAKAMI, 2011).

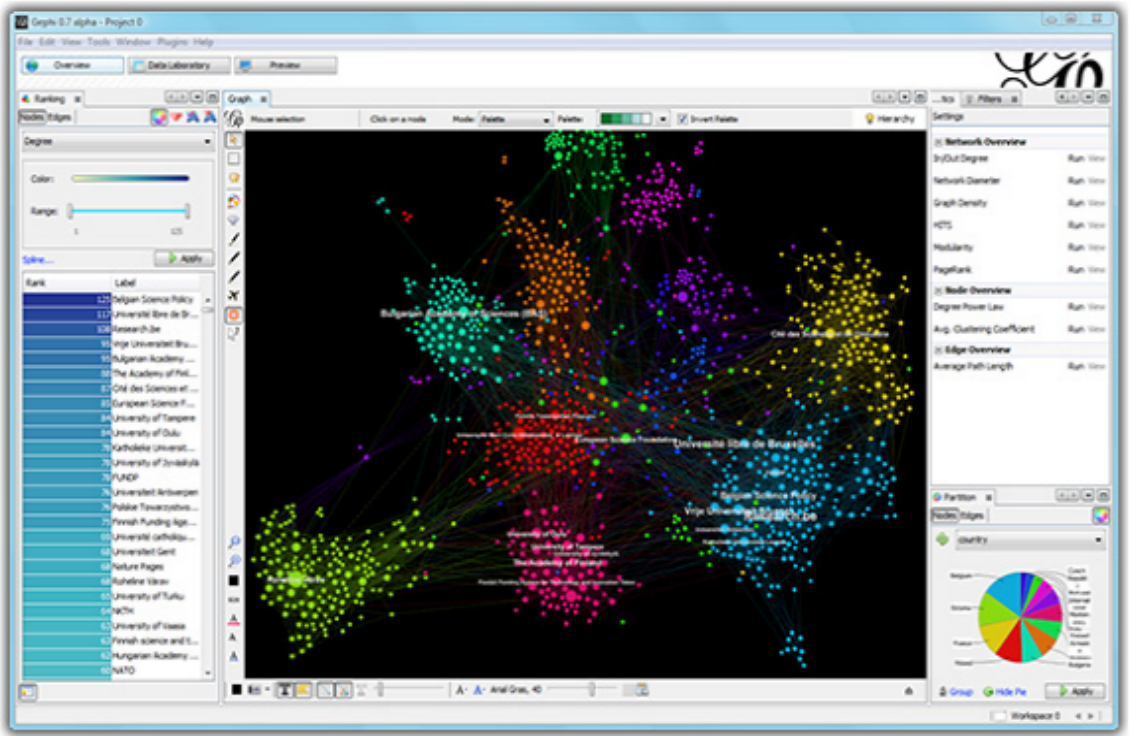


FIGURA 2: EXEMPLO DE INTERFACE DO GEPHI
FONTE: MURAKAMI (2011)

Após o estudo e análise das funcionalidades dessas ferramentas, optamos por realizar uma análise de interação entre colaboradores em projetos de código.

3 A ESTRUTURA SOCIAL DO DESENVOLVIMENTO DE SOFTWARE LIVRE E DE CÓDIGO ABERTO

Este capítulo apresenta o trabalho relacionado com a pesquisa proposta, contendo análise sobre o padrão de centralização ou descentralização de comunicação entre as equipes.

A publicação The Social Structure of free and open source software development.Crowston e Howison (2013), (A estrutura social do desenvolvimento de software livre e de código aberto) aborda os conceitos essenciais para o desenvolvimento deste trabalho.

Conforme mencionado por Crowston e Howison (2013), existem diferenças entre desenvolvimento de código livre e código aberto com relação ao desenvolvimento de código fechado, ou tradicional, ou comercial ou qualquer outra forma de produção de software que existem além de FLOSS.

São enfatizadas duas metáforas para descrever a organização de projetos FLOSS: a Catedral e o Bazar.

A Catedral descreve projetos de software desenvolvidos por grupos fechados, com pequenas aberturas para participação externa; e o Bazar, descreve projetos desenvolvidos de forma mais transparentes, abertos à participação de qualquer desenvolvedor que tenha interesse.

Algumas particularidades que são percebidas nos projetos de organização Bazar são:

- Utilizam diferentes licenças de software;
- Usam conjuntos de ferramentas distintos;
- Na maioria das vezes são formados por voluntários, em vez de empregados;

Visto algumas características, Crowston e Howison (2013) comentam que muitos dos supostos pontos fortes de desenvolvimento FLOSS estão intimamente ligados à comunicação exclusiva ou estrutura social dos projetos.

A Estrutura social de projetos de software permite que a investigação de questões de coordenação, controle, socialização, aprendizagem e continuidade, que são temas de grande interesse em estudos de grupos colaborativos e desenvolvimento livre.

A Centralização de comunicação refere-se aos métodos de comunicação entre os membros do projeto, como por exemplo, e–mail, sistemas de relatórios de bugs e mensagens instantâneas.

Projetos com alta centralização de comunicação têm um pequeno número de indivíduos que falam para um grupo maior de participantes, os quais não falam entre si, mas respondem ao pequeno grupo central.

Por outro lado, uma rede de comunicação descentralizada existiria em um projeto, quando a maioria dos membros de projetos falarem com um grande número de outros membros do projeto, não apenas um grupo limitado (CROWSTON; HOWISON, 2013).

Estudos feitos em projetos FLOSS demonstram um modelo hipotético de desenvolvimento que possui uma estrutura hierárquica que é caracterizada como um modelo cebola o qual pode ser visto na figura 04.

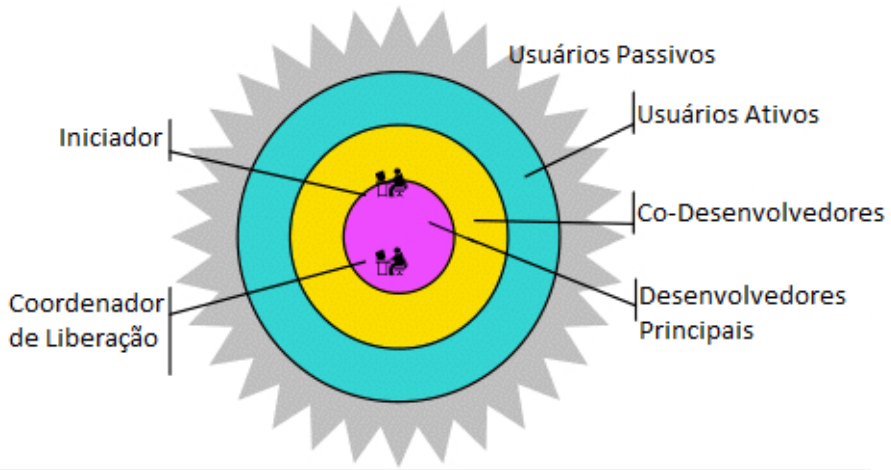


FIGURA 3: A ESTRUTURA DE EQUIPE DE DESENVOLVIMENTO FLOSS
FONTE: Crowston; Howison, 2005

Analisando a Figura 04, percebe-se que no centro do modelo cebola estão os desenvolvedores principais, que contribuem com a maioria do código e supervisionam o projeto e a sua evolução.

No próximo anel para fora estão os Co–desenvolvedores que são responsáveis pelos patches (por exemplo, correções de bugs), que são revisados e conferidos pelos desenvolvedores principais.

Mais externamente estão os usuários ativos que não contribuíram com código, mas fornecem casos de uso e relatórios de bugs bem como participam dos testes de novos lançamentos.

Por fim e com um limite virtualmente desconhecido, estão os usuários passivos do software que não participam de listas do projeto ou fóruns relacionados.

Baseado nisso, este estudo relacionado visou identificar os padrões de comunicação que seguem os relatórios da fase de gerenciamento de Bugs de um projeto de software, (CROWSTON; HOWISON, 2005).

Esta fase foi escolhida pelos autores porque engloba o momento no qual a maioria dos problemas do projeto é identificada e também é a fase do projeto que tem o maior número de participantes.

No estudo de caso, foi feito uma análise utilizando redes sociais de projetos do SourceForge, e examinando as 120 equipes dos mesmos, o que representa uma ampla gama de tipos de projeto de Software Livre.

Nesse estudo, os dados gráficos coletados pelos autores tiverem duas estruturas: Centralizada e descentralizada com relação à comunicação. Ambas quais são apresentadas na figura 05.

FIGURA 4: A ESTRUTURA CENTRALIZADA NA ESQUERDA E DESCENTRALIZADA NA DIREITA
FONTE: Crowston; Howison, 2005

Baseados nessas estruturas, os projetos demonstrados nas Figuras 06 e 07 abaixo mostram, respectivamente, os resultados obtidos em um projeto com comunicação centralizada e descentralizada.

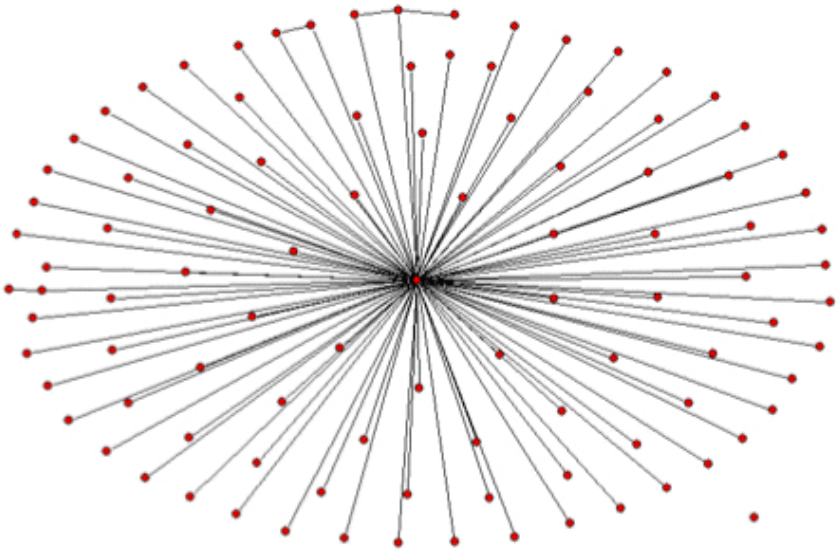


FIGURA 5: MODELO DE PROJETO ALTAMENTE CENTRALIZADO
FONTE: Crowston; Howison, 2005

Pode-se observar na Figura 6 o modelo do projeto com comunicação centralizada.

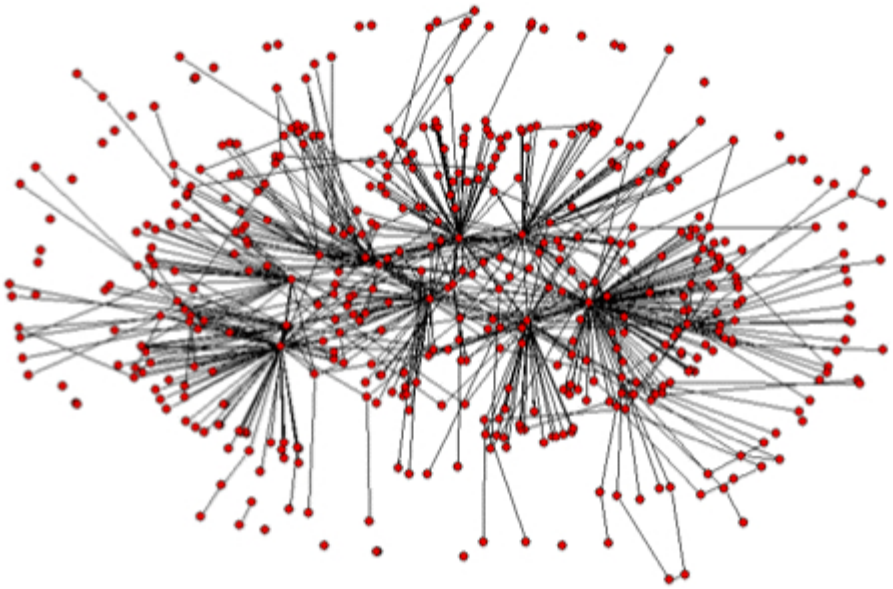


FIGURA 6: MODELO DE PROJETO DESCENTRALIZADO
FONTE: Crowston; Howison, 2005

Pode-se observar na Figura 7 o modelo do projeto com comunicação descentralizada.

As conclusões dos autores, baseados nas informações coletadas, mostraram que um determinado padrão de centralização das comunicações ou descentralização não é uma característica dos projetos FLOSS, na fase de gerenciamento de bugs.

Baseado nisso, este o trabalho apresentado inspirou os objetivos do trabalho atual, de modo que análise da comunicação proposta pelo autores foi reproduzida neste

trabalho em um projeto real que possui um destaque no GitHub nas Issues por ser de grande porte e possuir vários comentários de colaboradores diferentes.

Desta forma, foi realçada a relevância do estudo em questão para projetos com código Open Source, mostrando que baseado no estudo em questão outros projetos poderão melhorar a comunicação e, consequentemente, a qualidade do projeto.

No próximo capitulo vamos apresentar a metodologia.

4 METODOLOGIA

Neste capítulo serão apresentados os conceitos envolvidos no processo de pesquisa, assim como suas etapas. Em seguida serão apresentados os aspectos metodológicos da definição do experimento, juntamente com o fluxo do processo e por fim, como foi feita a coleta dos resultados obtidos no trabalho.

4.1 SOBRE O ESTUDO DO EXPERIMENTO

Visando melhorar a organização da pesquisa científica do trabalho em questão apresenta uma organização sistemática, pois exigiu uma disciplina constante no processo de pesquisa. O fluxo de tal processo pode ser visto na Figura 01, e a descrição de cada etapa do fluxo pode ser visualizada nas subseções abaixo.

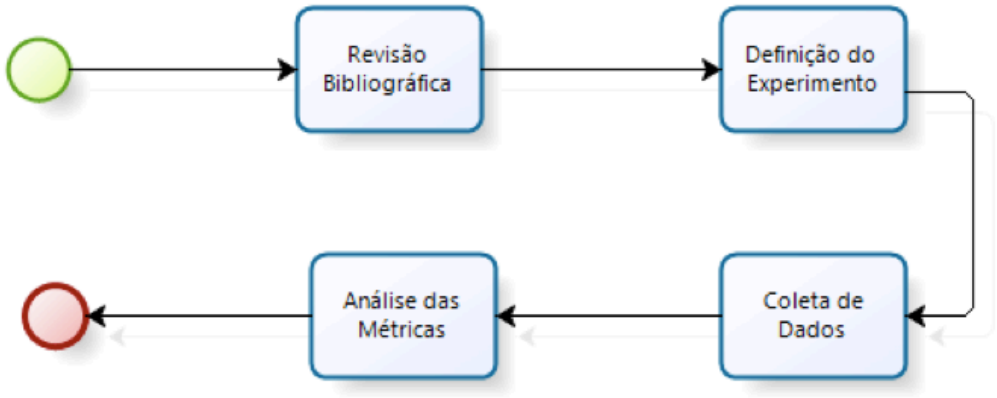


FIGURA 7: FLUXO DE DESENVOLVIMENTO DO TRABALHO

4.1.1 Revisão Bibliográfica de caráter exploratório

A primeira etapa da pesquisa foi focada na revisão bibliográfica. Nessa etapa foram estudados os trabalhos relacionados e foi definida a base teórica que foi necessária nos capítulos de desenvolvimento e na análise das informações obtidas.

4.1.2 Estudo do Experimento do Projeto Rails

O experimento realizado nesse trabalho teve como foco o projeto Rails, que se destacou acima de outros por ser de código aberto e ter seus dados disponíveis no GitHub, características obrigatórias para a análise feita neste estudo.

O Projeto Rails também foi escolhido por ser um dos frameworks mais populares de desenvolvimento web e por ser um dos projetos mais populares que estão hospedados no GitHub, possuindo 22.830 estrelas, 8.574 *forks*, 45.866 *commits.*

O código do Rails teve seu primeiro commit no dia 23/11/2004 e as estatísticas desse projeto faz com que se tenha informação suficiente para analisar bem a interação dos colaboradores no projeto.

4.1.3 Coleta de Dados do Projeto Rails

Os dados capturados foram as *Issues* e os *pull requests* com status *Open* do projeto Rails, no total 649. Foi utilizado um script na linguagem *Ruby* que usa a biblioteca *Octokit*, implementação oficial do GitHub, para acessar a API. Assim, foi possível coletar todos as *Issues/pull* requests do projeto. Para cada *issue/pull requests*, foi recuperado o autor e todos os usuários que comentaram. A partir desses dados, foram gerados os pares de nós no total de 6.500 arestas que se conectam. Esses pares foram gerados no formato .csv para abrir no formato suportado pelo Gephi.

Essas informações foram escolhidas, pois elas demonstram a interação entre os colaboradores do projeto. Esses dados podem ser interpretados em forma de grafos para obter informações importantes, sobre os colaboradores que mais interagem no projeto e as métricas obtidas podem ser utilizadas para melhorar a produtividade do projeto em geral.

4.1.4 Análise das Métricas do Projeto Rails

Foi escolhida a métrica de centralidade de grau, a qual é utilizada para analisar a as relações dos nós do grafo, permitindo assim identificar e caracterizar os nós que possuem o maior nível de importância a fim de melhorar a comunicação geral do projeto.

A métrica de centralidade de grau também permite a soma dos pesos das arestas de cada nó dependendo das suas relações e definir qual a distancia de determinado nó o número de todos os nós que podem ser ligados em um caminho da rede em questão.

Analisando a um grafo não direcional, o grau de um nó é simbolizado por $C_D(ni) = d(n_i)$, onde $d(n_i)$ representa o número de linhas incidentes em um nó, ou ainda de forma equivalente, o número de nós adjacentes a ele. Vale ressaltar que o grau de um nó pode variar de 0, caso no qual o nó é isolado, até g-1, caso no qual nó está contato com todos os demais nós do grafo, g representa o número total de nós (WASSERMAN E FAUST, 1994).

A métrica Modularidade, é possível encontrar várias comunidades suas sub-redes. Analisando a métrica permite medir a quantidade de ligação entre os nós entre diferentes grupos de comunidades. Funciona da seguinte forma.

- Cálculo dos caminhos de todas as ligações da rede;
- Procurar e remover a ligação com o maior valor de ligações;
- Recalcular as medidas do caminho para as restantes ligações da rede.

$$Q= \sum_i (e_{i\,i} - a_i^2)$$

A Modularidade é definida como em que $e = \sum_i e_{i\,i}$ é a fracção de ligações na rede que ligam nós na mesma comunidade, sendo que esta medida deve ser alta se houver uma boa decisão de comunidades. $a_i = \sum_i e_{i\,i}$ representa a fracção de ligações que se ligam os vértices na comunidade i (GIRVAN E NEWMAN, 2004).

A ferramenta Gephi foi utilizada na análise dos dados colhidos no GitHub, para construir o grafo de interações a partir dos dados coletados e para calcular a centralidade de grau dos nós.

Com o objetivo de aumentar a credibilidade e organização do estudo, este capítulo mostrou como o processo de pesquisa descrito foi planejado e referencial teórico foi elaborado com o objetivo de facilitar a tarefa de observação e análise, melhorando a credibilidade do estudo. O instrumento de coleta de dados utilizados na pesquisa foi planejado de forma a garantir a integridade dos resultados esperados.

No próximo capitulo vamos apresentar o desenvolvimento e resultados obtidos após a análise de interação.

5 ANÁLISE DE INTERAÇÃO ENTRE COLABORADORES EM PROJETO DE CÓDIGO ABERTO

Este capítulo irá abordar o desenvolvimento dos resultados e o detalhamento de como a metodologia descrita no capítulo anterior foi utilizada para que fossem tiradas as conclusões desse trabalho.

5.1 MÉTRICAS DE ANÁLISE DE REDES SOCIAIS

Análise de Redes Sociais estabelece um novo paradigma na pesquisa sobre a estrutura social. Para esse trabalho, destacamos duas métricas que podem ser utilizadas para analisar a interação entre os colaboradores do projeto. A rede[6] foi analisada de acordo com a Modularidade e em seguida foi utilizado o Centralidade de Grau.

Quando analisada de acordo com a Modularidade foi possível encontrar várias comunidades em que a rede podia ser subdividida. Em uma Rede Social, é de suma importância o conhecimento do grau individual de um ator. Vale ressaltar que um ator nada mais é do que, uma unidade discreta, ou seja, um colaborador.

A centralidade de um ator significa a identificação da posição em relação as trocas e a comunicação na rede. Desta maneira, o grau de centralidade de um nó, Centralidade Grau, é definido pela quantidade de enlaces diretos que um ator possui.

Baseado nisso, esta métrica foi utilizada para analisar a relevância dos colaboradores da rede descrita neste trabalho, analisando a importância dos nós, baseada nas conexões que são estabelecidas entre nós vizinhos.

Dessa forma no contexto da interação entre colaboradores, objetivo deste trabalho, o Grau de Centralidade de um colaborador individualmente pode não representar muito, entretanto, ao comparamos os graus de centralidade dos diferentes colaboradores de um grupo, podemos determinar se a interação está centralizada em um pequeno grupo de colaboradores, ou distribuída uniformemente. Com resultado analisado a maioria tendem a ter padrões de comunicação centralizada e descentralizada

[6] A rede vai ser gerada utilizado a ferramenta Gephi, a análise realizada foi com os dados colhidos no GitHub, para construir o grafo de interações a partir dos dados coletados e para calcular a centralidade de grau dos nós.

Se o primeiro caso for verdadeiro, isso pode demonstrar uma fragilidade da estrutura de interação. Caso um colaborador de alto grau de centralidade deixe de participar da rede por algum motivo, toda a interação do grupo pode ser comprometida.

Neste trabalho as redes são modeladas como grafos não dirigidos onde os nós representam os colaboradores e as arestas indicam as relações entre eles.

O tamanho do nó é proporcional ao número de relações entre os colaboradores e a cor dos nós identifica a qual sub-rede o colaborador pertence. Na Figura 08 está ilustrada a maneira como o grau dos atores está distribuído na rede, é destacado nesse ponto que existe um nó com grau[7] maior que 500.

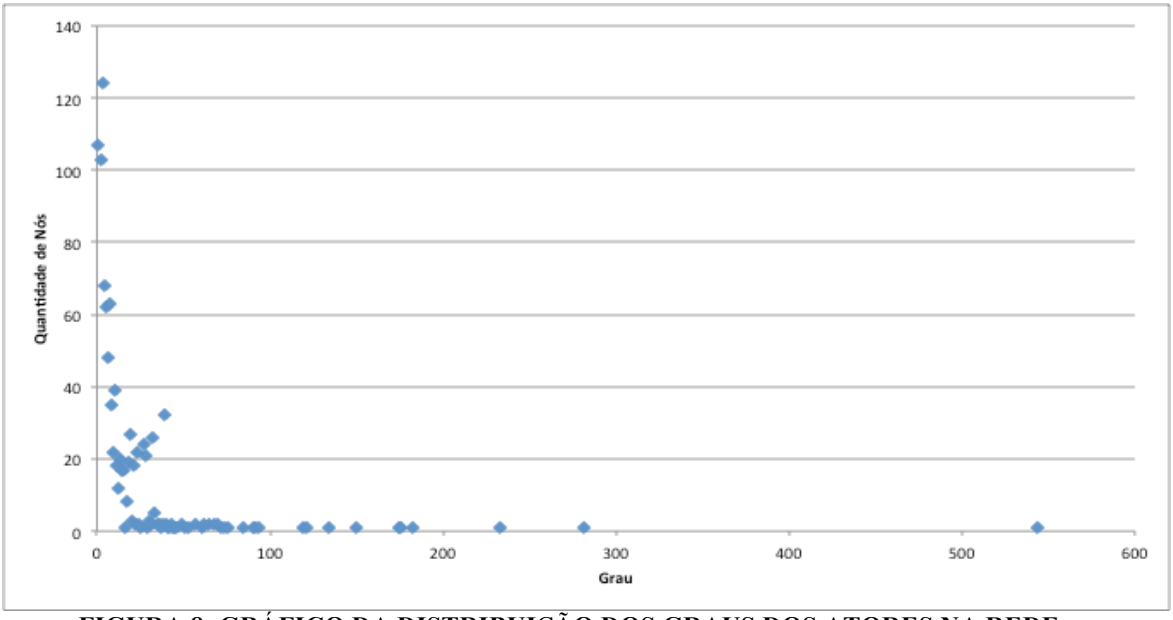
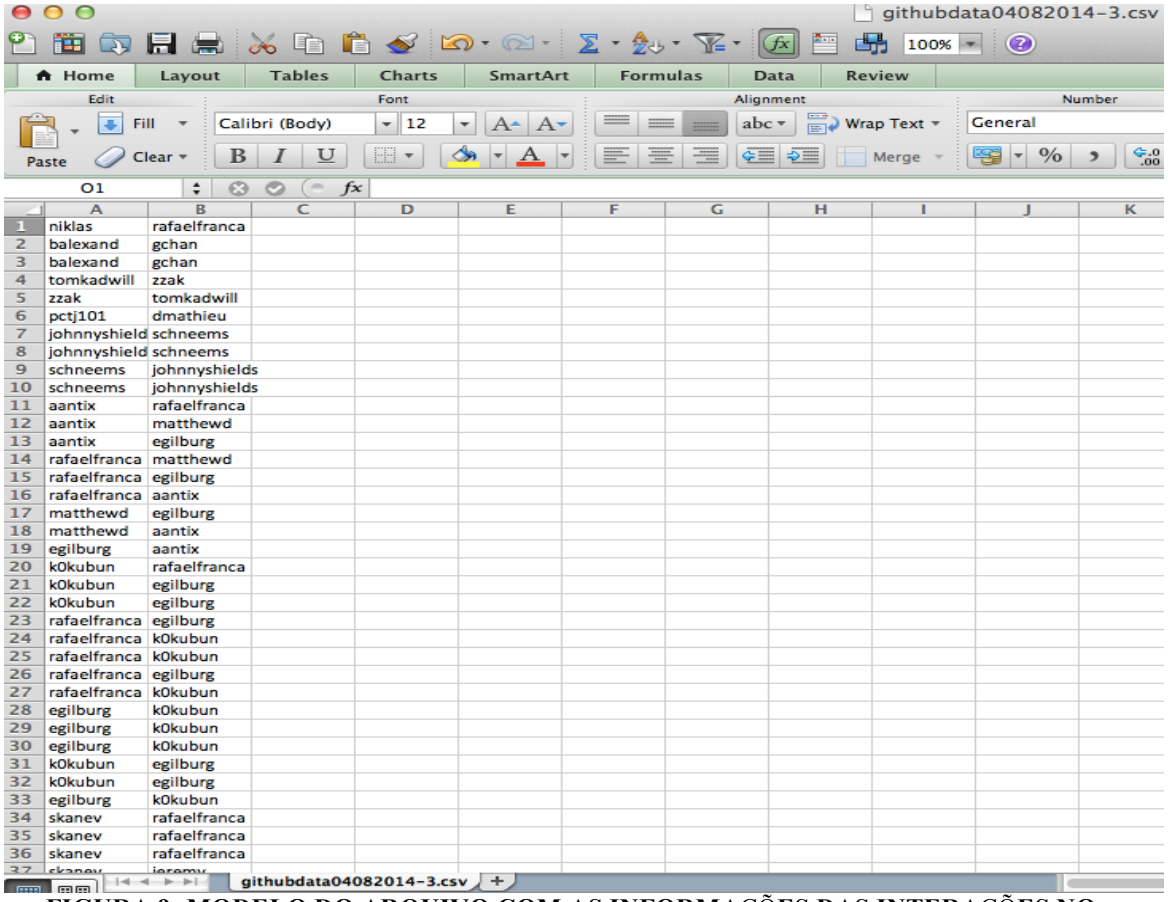


FIGURA 8: GRÁFICO DA DISTRIBUIÇÃO DOS GRAUS DOS ATORES NA REDE

As ligações estudadas através da análise de redes sociais dentro deste projeto de código aberto foram capazes de identificar fluxos de informação entre os atores, sendo assim possível avaliar as interações entre atores (nós) para a obtenção de informações vantajosas, que se utilizadas de maneira correta podem aumentar a produtividade do projeto.

A partir dos dados coletados, foi gerado como saída um arquivo no formato CSV contendo nome dos colaboradores que fazem a implementação do projeto e outros colaboradores ao qual eles se ligam no quesito de comunicação. A Figura 09 mostra os dados do arquivo que vai ser usado no Gephi para visualização dos grafos.

[7] Grau: é a quantidade de conexões que um nó possui.



	A	B	C	D	E	F	G	H	I	J	K
1	niklas	rafaelfranca									
2	balexand	gchan									
3	balexand	gchan									
4	tomkadwill	zzak									
5	zzak	tomkadwill									
6	pctj101	dmathieu									
7	johnnyshield	schneems									
8	johnnyshield	schneems									
9	schneems	johnnyshields									
10	schneems	johnnyshields									
11	aantix	rafaelfranca									
12	aantix	matthewd									
13	aantix	egilburg									
14	rafaelfranca	matthewd									
15	rafaelfranca	egilburg									
16	rafaelfranca	aantix									
17	matthewd	egilburg									
18	matthewd	aantix									
19	egilburg	aantix									
20	k0kubun	rafaelfranca									
21	k0kubun	egilburg									
22	k0kubun	egilburg									
23	rafaelfranca	egilburg									
24	rafaelfranca	k0kubun									
25	rafaelfranca	k0kubun									
26	rafaelfranca	egilburg									
27	rafaelfranca	k0kubun									
28	egilburg	k0kubun									
29	egilburg	k0kubun									
30	egilburg	k0kubun									
31	k0kubun	egilburg									
32	k0kubun	egilburg									
33	egilburg	k0kubun									
34	skanev	rafaelfranca									
35	skanev	rafaelfranca									
36	skanev	rafaelfranca									

FIGURA 9: MODELO DO ARQUIVO COM AS INFORMAÇÕES DAS INTERAÇÕES NO GITHUB

5.2 GERAÇÃO DE GRAFOS NA REDE GEPHI

Utilizando o Gephi, pode-se visualizar e analisar a rede presente no arquivo CSV, demonstrando em um grafo claramente quem são as subredes dentro da rede, permitindo explorar o fluxo das informações que agrupam colaboradores, montado as subredes e medindo a distancia entre nós.

A Figura 10 ilustra que grafo gerado que ainda não sofreu nenhum tipo de arranjo por um dos vários algoritmos do software e nem opções de exibição.

A única percepção que temos na imagem é que cada nó aparece com um circulo e suas ligações, como linhas que muitas vezes se sobrepõe.

Cada colaborador que adiciona uma *Issue* no GitHub ganha um nó, os colaboradores que comentam uma *Issue* são interligados, inclusive o que cria a mesma.

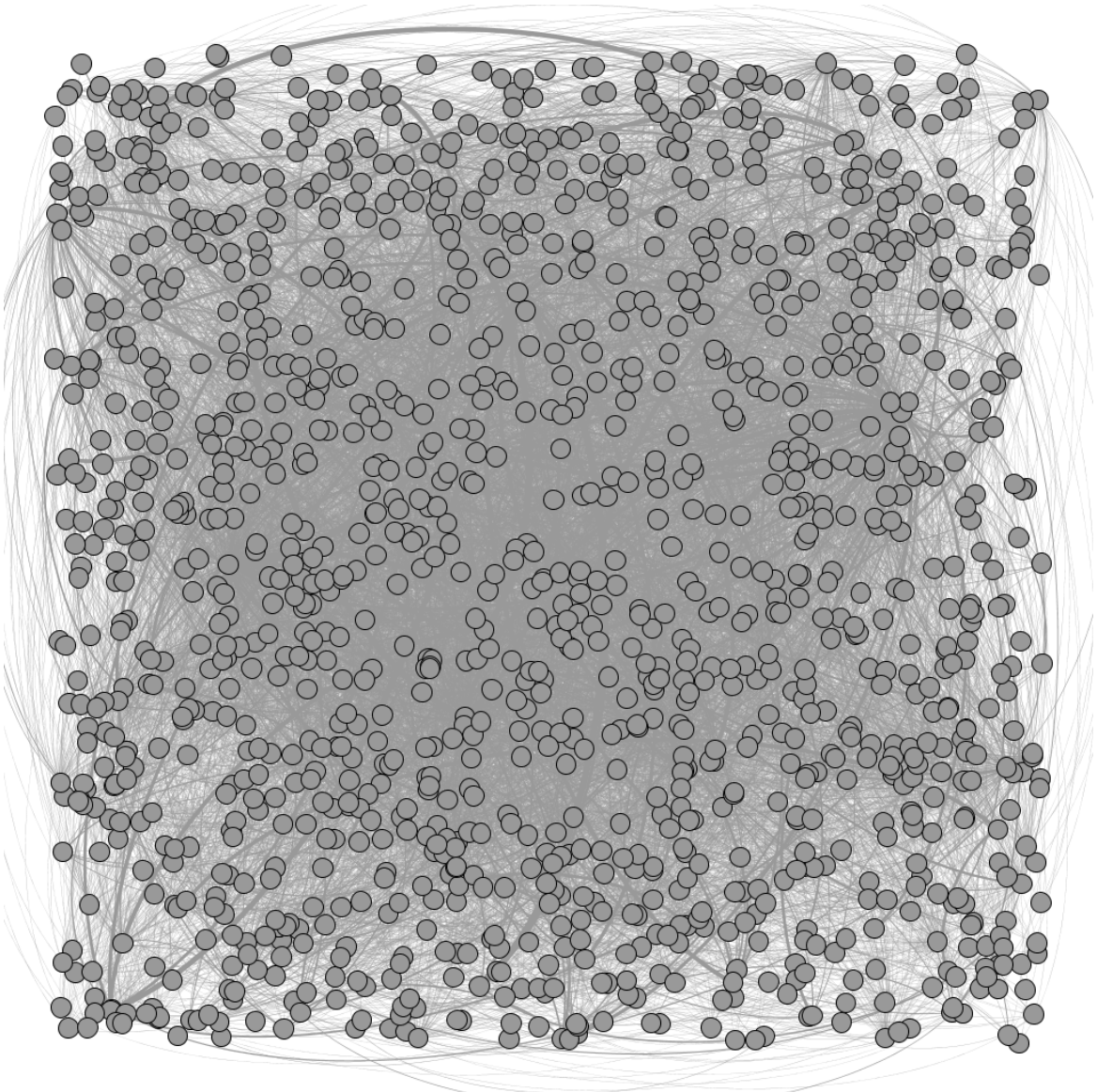


FIGURA 10: IMAGEM DO ARQUIVO SEM O TRATAMENTO PARA FACILITAR INTERPRETAÇÃO

Para realizar análise desta rede são necessários alguns ajustes na ferramenta Gephi, para que a sua visualização fique mais apresentável na interpretação dos grafo. Na aba classificação pode-se calcular diversas métricas ente elas, *"hits ", "pagerank"* e *"modularidade",* já na aba de Distribuição é possível escolher entre diversos algoritmos de arranjo, entre eles o Force Atlas, Yufan Hu e Nocerlap e definindo uma graduação de cores, temos o arranjo ilustrado na Figura 11. Com isso, a execução da Figura abaixo, mostra uma visualização muito mais esclarecedora de como está a rede social do perfil analisado.

Na Figura 11 pode-se observar o grafo das interações entre os colaboradores e o estado da rede do perfil analisado. O grafo demostra que existem sub-redes bem

definidas nesta rede. É possível analisando as ligações dos colaboradores que interagem com cada um, podemos ver que círculos sociais formam outras subredes no grafo e os colaboradores que atuam como elo entre uma rede e outra.

As ligações são importantes na disseminação ou coleta de informações, auxiliando a interação dos colaboradores no projeto e melhorando a produtividade em geral.

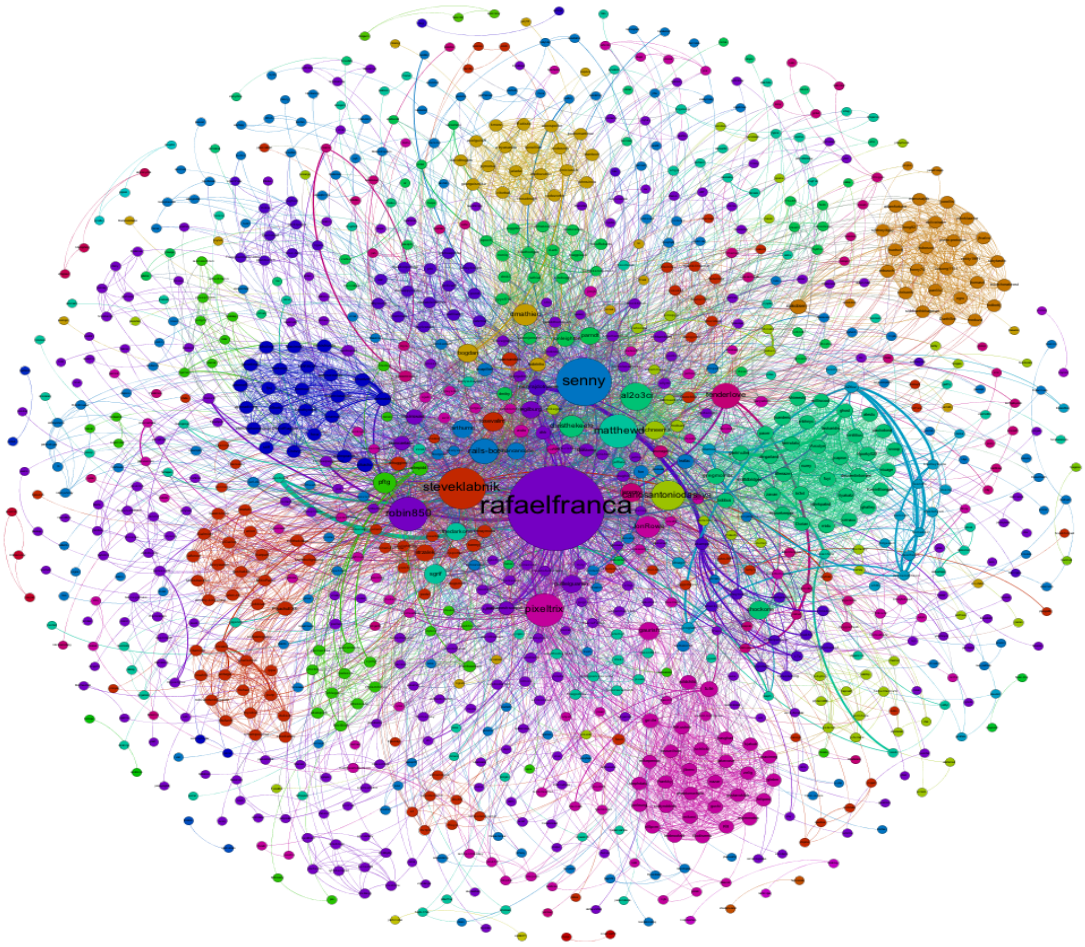


FIGURA 11: REDE DAS INTERAÇÕES ENTRE OS COLABORADORES

No grafo da Figura 11 é possível observar as ligações fracas, mais distantes, que por sua vez são os atores com menor quantidade de interações, enquanto que os elementos centrais da rede são atores que possuem maior comunicação.

Podemos identificar os colaboradores com nós proporcionais entre si, ou seja, ao número de conexões que os colaboradores utilizam ao longo da comunicação. Logo, é percebido que na rede, as arestas (linhas) que interligam os atores (nós), mostram o fluxo de interação e as direções dos atores. Portanto as arestas (linhas), mostram o fluxo

de interação dos colaboradores, quanto mais ligações maior a dimensão das linhas(arestas).

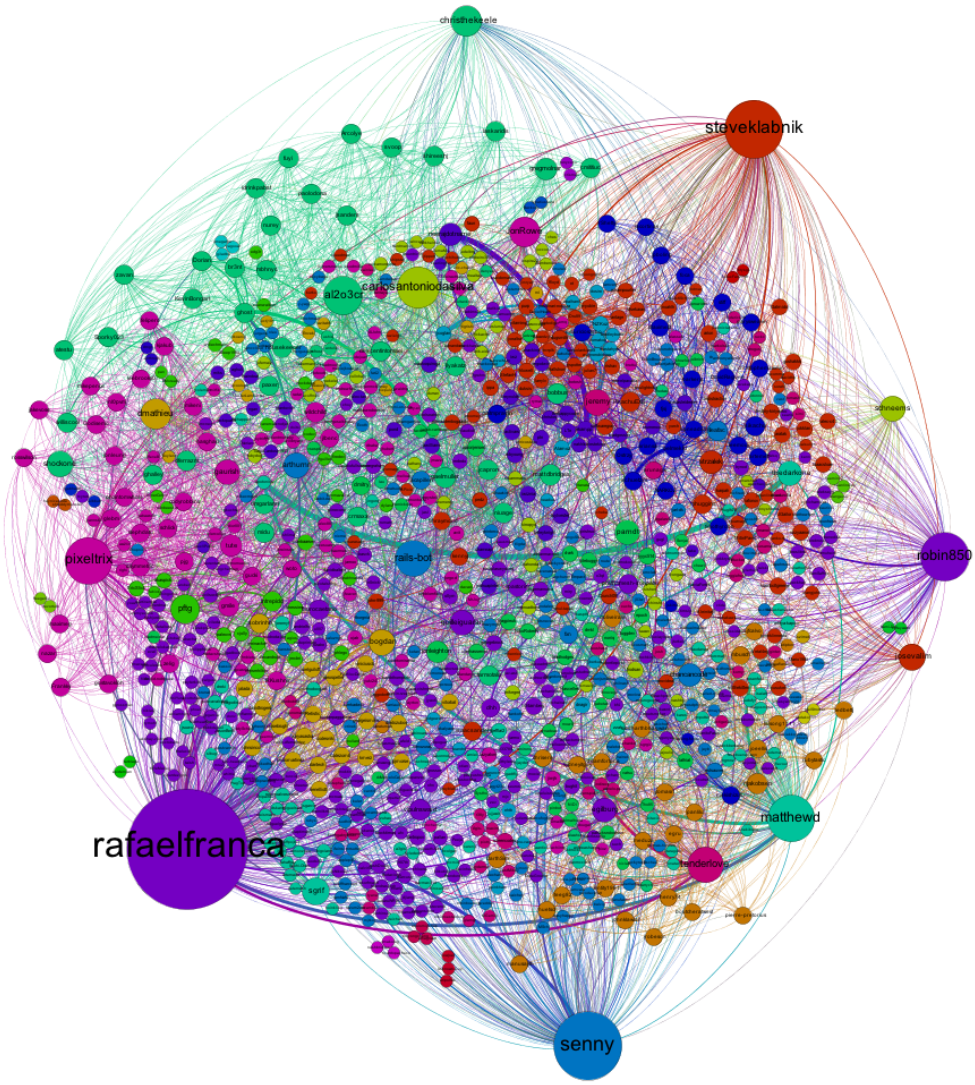


FIGURA 12: GRAFO DAS COMUNIDADES DA REDE

A rede da Figura 12 ilustra o grafo das comunidades da rede, ou seja, uma sub-rede conectando uma grande parte dos nós da rede original, o que indica que a rede de colaborador é bastante coesa. Também se podem observar os nós proporcionais ao grau de cada colaborador, ou seja, ao número de conexões que o colaborador estabeleceu ao longo da comunicação. Destaca-se na imagem, Rafael Franca, o colaborador que mais tem conexões com outros atores, o maior disseminador de conhecimento do projeto.

Portanto analisando esse grafo o resultado tendem a ter padrões de comunicação mais descentralizadas.

Após o calculo da métrica modularidade identificamos 29 (vinte e nove) comunidades onde destas destacamos as cinco maiores sendo elas descritas na tabela 03.

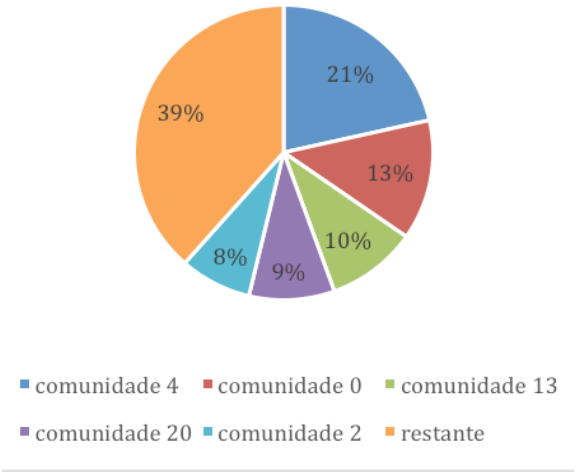


FIGURA 13: DISTRIBUIÇÃO DO NÚMERO DE COMUNIDADES DOS ATORES

TABELA 3: COMUNIDADES MAIS RELEVANTES

Identificador	Participantes
Comunidade 4	219
Comunidade 0	135
Comunidade 13	100
Comunidade 20	94
Comunidade 2	79
Restante	393

A Figura 13 apresenta a distribuição do número de comunidades dos atores. Observa-se que apenas 39% dos atores não estão relacionados nessas sub-redes, já em relação ao numero de colaboradores, podemos ver que a grande maioria dos mesmos (21%) se comunicam com a Comunidade 4, sendo essa a maior disseminadora de conhecimento no projeto (Tabela 03).

Pode-se observar na Figura 14 a interação entre as 5 maiores comunidades da rede, também é destacado que os nós com maior grau de colaboração estão presentes nesta rede.

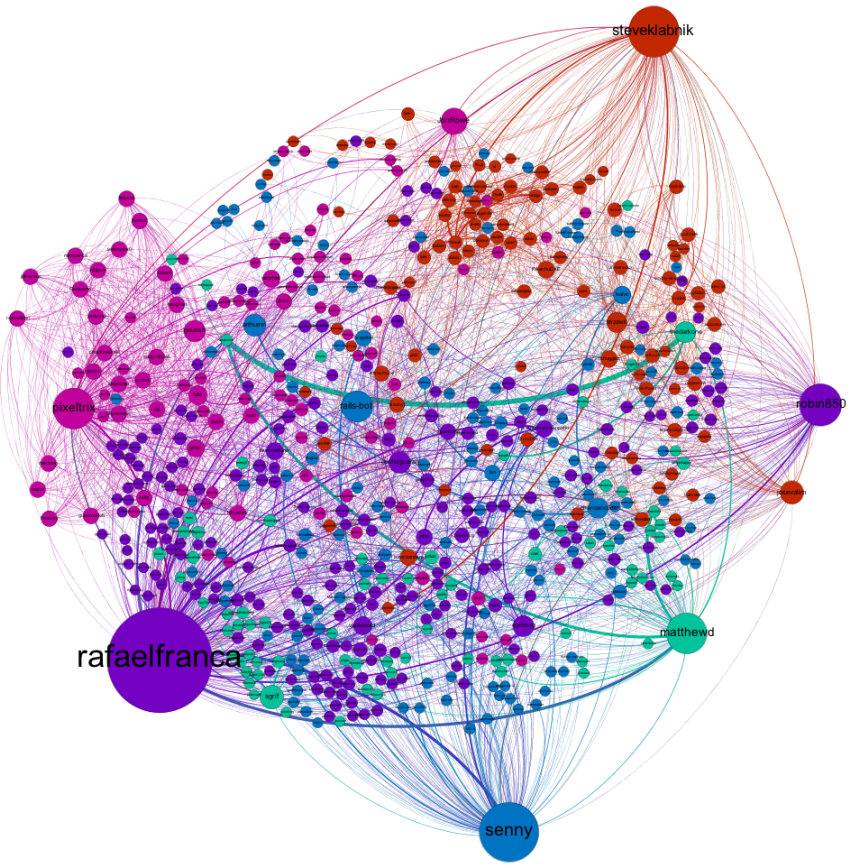


FIGURA 14: INTERAÇÃO ENTRE AS CINCO MAIORES COMUNICADES

Nas Figuras 15, 16, 17, 18, 19 são apresentados os grafos das 5 maiores comunidades da rede separadamente, destacando com melhor visibilidade as interações entre os colaboradores das sub-redes como: Rafael Franca, Senny, pixeltrix, steveklabnik e Matthewd

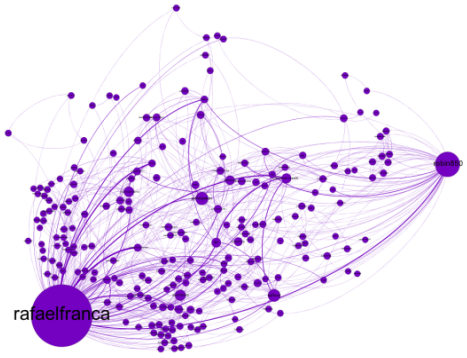


FIGURA 15: RAFAELFRANCA

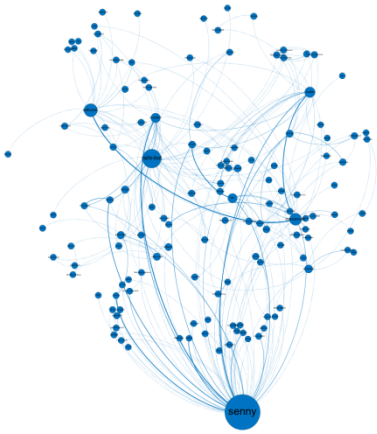


FIGURA 16: SENNY

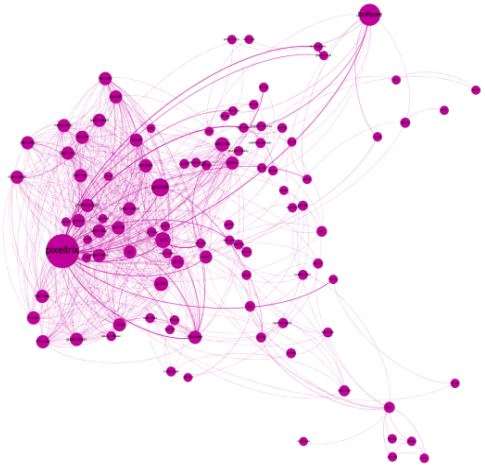


FIGURA 17: PIXELTRIX

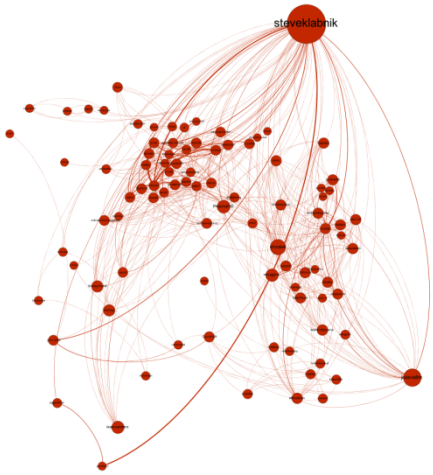


FIGURA 18: STEVEKLABNIK

FIGURA 19: MATTHEWD

Com os dados das redes coletados o calculo da centralidade de grau foi possível, identificando os colaboradores mais relevantes do projeto, o resultado com os 10 principais colaboradores pode ser visualizado na Tabela 04.

Tabela 4: Principais colaboradores

Colaborador	Centralidade Grau
rafaelfranca	544
Senny	282
steveklabnik	233
robin850	183
pixeltrix	176
matthewd	175
carlosantoniodasilva	150
al2o3cr	134
rails-bot	121

Baseado na métrica de centralidade de grau, listamos na tabela 04 os colaboradores considerados mais centrais os mais relevante na rede. O grau de um nó representa a quantidade de conexões, no nosso caso, de relações entre os colaboradores. Sendo assim, os nomes na coluna da esquerda da Tabela 04 indicam os colaboradores com o maior numero de comunicações. Este número indica que tais colaboradores tiveram papel fundamental no andamento do projeto, os demais componentes conectados presentes na rede porém não deixa claro qual sua importância na rede uma vez que colaborador podem estabelecer conexões com outros colaboradores nós na rede e podem ser utilizados de forma a melhorar a comunicação entre os membros menos ativos, auxiliando a melhorar a produtividade do projeto.

6 CONCLUSÃO E SUGESTÕES DE TRABALHOS FUTUROS

Neste trabalho foi apresentado conceitos de software livre, destacando suas vantagens em questão de custo, possibilidade de customização, e facilidade de utilização. Foram comentadas também as diferentes licenças utilizadas por esse tipo de software e as diferenças entre sistemas com código livre e de código aberto.

Com os conceitos em mente essa pesquisa utilizou informações sobre a Rede Social dos colaboradores do projeto Rails, um dos projetos mais populares de código aberto no repositório GitHub a fim de analisar os padrões de comunicação e gerar métricas utilizáveis na conclusão.

As métricas obtidas incluíram a modularidade da rede para encontrar as comunidades ou sub-redes do projeto e o grau de centralidade, a fim de destacar os níveis de comunicação entre os participantes e colaboradores.

Os dados obtidos foram tratados e representados na forma de grafos, cada colaborador que adiciona uma *Issue* no GitHub ganha um nó e os colaboradores que comentam uma *Issue* são interligados.

Com essas estruturas, foi possível destacar os colaboradores que mais disseminaram conhecimento no projeto, mostrar as sub-redes do projeto, focadas nos colaboradores principais que tinham maior fluxo de comunicação e identificar participantes que não comentaram muito durante o projeto.

A análise também mostrou que o projeto era descentralizado com relação a disseminação de conhecimento o que auxiliou na produtividade do mesmo visto que facilita a participação de um maior número de pessoas relevante.

Visto que o foco deste trabalho foi a comunicação entre os integrantes atuais do projeto Rails, como trabalhos futuros, a participação de novos integrantes também pode ser analisada afim de aumentar o nível de colaboração entre os integrantes e outros projetos também podem ser analisados a fim de determinar padrões de comunicação nas diferentes fases do desenvolvimento de software. Também como trabalho futuro, uma análise da evolução da comunicação durante o tempo pode ser feita, de forma a auxiliar a criar um repositório de lições aprendidas para novos projetos de código livre.

REFERÊNCIAS

CAMPOS, A. **O que é software livre. BR-Linux**. Florianópolis, março de 2006. Disponível em <http://br-linux.org/linux/faq-softwarelivre>. Acesso em: 03 de Abril de 2014.

COSTA, R. C. da; SANTOS, R. F. O. dos. **Conhecendo o software livre**. Disponível em: <http://www.periodicos.letras.ufmg.br/index.php/ueadsl/article/viewFile/2504/2456>. Acesso em: 10 mai. 2014.

CARMEL, E., TIJA, P. (2005), **Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce**. UK: Cambridge.

CLARA, M. **Análise e visualização de redes: o gephi**. Disponível em: <http://humanidadesdigitais.org/2013/08/16/analise-e-visualizacao-de-redes-o-gephi/>. Acesso em: 26 mai. 2014

CROWSTON, K.; HOWISON, J. **The social structure of free and open source software development.** Disponível em: <http://firstmonday.org/ojs/index.php/fm/article/viewArticle/1207/1127>.Acesso em: 22 abril 2014.

DEMO, P. **Metodologia do conhecimento científico**. São Paulo: Atlas, 2000. Disponível em: <http://www.pedagogiaemfoco.pro.br/met02a.htm>. Acesso em: 26 abr. 2014.

DAMIAN, D.; ZOWGHI, D. **An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations**. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 36., Hawaii, 2003. Anais Eletrônicos. Disponível em: <http://www.computer.org/>. Acesso em: 08 de abr. 2014.

EIS, D. **Iniciando no GIT – Parte 1**. Disponível em: <http://tableless.com.br/iniciando-no-git-parte-1/>. Acesso em: 14 fev. 2014.

FILHO, S. S.; STEFANUTO, G. N.; LUCCA, J. E. de.; ALVES, A. M. **O impacto Software livre e de código aberto (SL/CA) nas condições de apropriabilidade na indústria de software brasileira.** Tema: Propriedad industrial e intelectual; Información em C&T. XI seminário Latino, ALTEC 2005.

NEWMAN, M.E.J; M GIRVAN (2004). **Finding and evaluating community structure in networks, Physical Review.**

GIL, A.C. **Métodos e técnicas de pesquisa social**. São Paulo: Atlas, 1999. Disponível em: <http://www.administradores.com.br/informe-se/artigos/pesquisa-qualitativa-exploratoria-e-fenomenologica-alguns-conceitos-basicos/14316/>. Acesso em: 20 abr. 2014.

GARCIA, M. N.. S.; SILVANA, M. B. dos.; PEREIRA, R. da S.; ROSSI, G.; **Software livre em relação ao software proprietário: aspectos favoráveis e desfavoráveis percebidos por especialistas**. Free software in connection with the proprietary software: favorable and unfavorable aspects perceived by experts. Data de aprovação: 06/12/2010. Disponível em: <http://seer.uscs.edu.br/index.php/revista_gestao/article/view/1061/847>. Acesso em: 09 de abril de 2014.

GAMA, A. **GitHub e Git – Colaboração e Organização**. Disponível em: <http://www.devmedia.com.br/github-e-git-colaboracao-e-organizacao/24150 >. Acesso em: 23 mar. 2014

HEATH, O. V. S. **A estatística na pesquisa cientifica**. São Paulo: EPU: Ed. Da Universidade de São Paulo, 1981. 95p.

HEXSEL, R.A. **Propostas de ações de Governo para incentivar o uso de software livre**. Relatório Técnico do Departamento de Informática da UFPR, n. 004/2002, Curitiba, outubro, 2002. Disponível em: <http://www.inf.ufpr.br/info/techrep/RT_DINF004_2002.pdf>. Acesso em: 18 de abril de 2014.

HERBSLEB, J. D., MOITRA, D. (2001), **Global Software Development**, IEEE Software,
March/April, EUA, p. 16-20.

LUZ, C. C. F. **Projeto de Migração para Software livre em Micro e Pequenas Empresas.** Caso Cor de Rosa Moda Feminina. Universidade Federal de Santa Catarina. Florianópolis-sc. 2007/01. Disponível em: <http://projetos.inf.ufsc.br/arquivos_projetos/projeto_445/Relatorio%20Final.pdf>. Acesso em: 15 abr. 2014

MINAYO, M.C. de S. (Org.) **Pesquisa social: teoria, método e criatividad**e. 22 ed. Rio de Janeiro: Vozes, 2003. . Disponível em: <http://www.administradores.com.br/informe-se/artigos/pesquisa-qualitativa-exploratoria-e-fenomenologica-alguns-conceitos-basicos/14316/>. Acesso em: 20 abr. 2014.

MARTELETO, R. M.. **Análise de redes sociais: aplicação nos estudos de transferência da informação**. Ciência da Informação, Brasília, v. 30, n. 1, p. 71-81, jan./abr. 2001.

MACEDO, T. M. B.. **Redes informais nas organizações:a co-gestão do conhecimento.** Ciência da Informação, Brasília, v. 28, n.1, p. 94-100, jan. 1999.

MURAKAMI, T. **Introdução ao gephi.** Disponível em: <http://bsf.org.br/2011/10/18/introducao-ao-gephi/>. Acesso em: 19 fev. 2014

OPENSUSE. **GIT**. Disponível em: <http://pt.opensuse.org/Git>. Acesso em: 17 fev. 2014

PEREIRA, I. **O movimento do Software livre.** Disponível em: < http://www.ces.uc.pt/lab2004/pdfs/InesPereira.pdf >. Acesso em: 10 mai. 2014

PEREIRA, F.; MARINHO, I.; OLIVEIRA, N. **Open Source Software Development.** Disponível em: <http://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/es_final_11.pdf>. Acesso em: 30 jun 2014.

PRIKLADNICKI, R.; AUDY, J.; EVARISTO, R. **Distributed Software Development: Toward an Understanding of the Relationship Between Project Team, Users and Customers.** In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, 5., Angers, 2003. Anais. v.3. p.417-423.

PINTO, P. **Git – Sistema de controle de versões de software(parte II).** Disponível em: <http://pplware.sapo.pt/linux/git-sistema-de-controlo-de-versoes-de-software-parte-ii/>. Acesso em: 13 jan 2014.

SABINO,V.; KON, F.. **Licenças de Software Livre História e Características.** Relatório Técnico RT-MAC-IME-USP 2009-01. Centro de Competência em Software Livre Departamento de Ciência da Computação. Instituto de Matemática e Estatística. Universidade de São Paulo, Março de 2009. Disponível em: <http://ccsl.ime.usp.br/files/relatorio-licencas.pdf>. Acessado em: 15 de maio de 2014.

SILVA, G. **Software Livre – uma mudança de modelo.** Disponível em: <http://www.infosites.com.br/artigo.html?id=8a1db1fd3bd80d64013bd8237fc1004d >. Acesso em: 15 jun 2014

SANT'ANNA, M. **As contradições do Open Source**. Disponível em:<http://www.linhadecodigo.com.br/artigo/379/as-contradicoes-do-open-source.aspx>. Acesso em: 25 mai 2014

SCOTT, J. **Social Network Analysis**. Ahandbook.2nd ed. London, UK: Sage Publications,2000.

TOMAÉL, M. I.; ALCARÁ, A. R.; CHIARA, I. G. Di. **Das redes sociais à inovação.** Ci. Inf, Brasilia, v 34, n. 2, p, 93-104, maio/ago. 2005

WELLMAN, B. **Are personal communities local? a dumptarian reconsideration.**Social Networks, Amsterdam, v. 18, p. 347-354, 1996.

WASSERMAN, S., FAUST, K., 1994. Social Network Analysis: Methods and Applications. 8. Ed. Cambridge University Press, New York, EUA.