UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE INFORMÁTICA
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

VIDAL DANIEL DA FONTOURA

PREDIÇÃO DE FALHAS EM PROJETOS DE SOFTWARE LIVRE BASEADA EM MÉTRICAS DE REDES SOCIAIS

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2011

VIDAL DANIEL DA FONTOURA

PREDIÇÃO DE FALHAS EM PROJETOS DE SOFTWARE LIVRE BASEADAS EM MÉTRICAS DE REDES SOCIAIS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso, do Curso Superior de Tecnologia em Sistemas para Internet da Coordenação de Informática da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito para aprovação na disciplina de TCC 2.

Orientador: Prof. Me. Igor Scaliante Wiese

CAMPO MOURÃO
2011

ATA DA DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO

As **vinte e duas horas** do dia **vinte e cinco de novembro de dois mil e onze** foi realizada na sala F103 da UTFPR-CM a sessão pública da defesa do Trabalho de Conclusão do Curso Superior de Tecnologia em Sistemas para Internet do acadêmico **Vidal Daniel da Fontoura** com o título **PREDIÇÃO DE FALHAS EM PROJETOS DE SOFTWARE LIVRE BASEADA EM MÉTRICAS DE REDES SOCIAIS**. Estavam presentes, além do acadêmico, os membros da banca examinadora composta pelo professor **Me. Igor Scaliante Wiese** (Orientador-Presidente), pelo professor **Me. Gabriel Costa Silva** e pelo professor **Me. Igor Fábio Steinmacher**. Inicialmente, o aluno fez a apresentação do seu trabalho, sendo, em seguida, arguido pela banca examinadora. Após as arguições, sem a presença do acadêmico, a banca examinadora o considerou **Aprovado** na disciplina de Trabalho de Conclusão de Curso e atribuiu, em consenso, a nota _____**.** Este resultado foi comunicado ao acadêmico e aos presentes na sessão pública. A banca examinadora também comunicou ao acadêmico que este resultado fica condicionado à entrega da versão final dentro dos padrões e da documentação exigida pela UTFPR ao professor Responsável do TCC no prazo de **quinze dias**. Em seguida foi encerrada a sessão e, para constar, foi lavrada a presente Ata que segue assinada pelos membros da banca examinadora, após lida e considerada conforme.

Observações:

_____

_____

_____

_____

_____

Campo Mourão, 25 de novembro de 2011.

Prof. Me. Gabriel Costa Silva Membro	Prof. Me. Igor Fábio Steinmacher Membro	Prof. Me. Igor Scaliante Wiese Orientador

A folha de aprovação assinada encontra-se na coordenação do curso.

RESUMO

Fontoura, Vidal D. PREDIÇÃO DE FALHAS EM PROJETOS DE SOFTWARE LIVRE BASEADAS EM MÉTRICAS DE REDES SOCIAIS. 2011. Trabalho de Conclusão de Curso – Graduação em Tecnologia em Sistemas para Internet. Universidade Tecnológica Federal do Paraná. Campo Mourão 2011.

Este trabalho foi desenvolvido com objetivo de analisar fatores humanos (comunicação e cooperação) como parâmetros para predição de falhas, visto que problemas de comunicação e cooperação levam a falhas de projetos. Muitos trabalhos de engenharia de software analisam a comunicação e a cooperação dos desenvolvedores, extraindo informações de repositórios de código fonte, e mensagens de mecanismos de comunicação, dentre eles, alguns trabalhos utilizam redes sociais e SNA para analisar a comunicação e a cooperação dos desenvolvedores. O experimento realizado neste trabalhou consistiu na geração de matrizes (desenvolvedor vs desenvolvedor) com base em dados referentes à comunicação e cooperação extraídos de repositórios de software livre, estas matrizes representam redes sociais dos desenvolvedores, em seguida foram calculados os valores para métricas de SNA e estas métricas foram utilizados como conjunto de treinamento e validação de classificadores bayesianos que efetivamente realizam a predição e por fim foram utilizadas a medidas estatísticas *Recall/Precision/F-Mesaure*, para avaliar o desempenho das predições. A partir do experimento realizado foi possível verificar que a cooperação pode auxiliar na predição de falhas, e que a comunicação não obtém um desempenho satisfatório para predição de falhas.

Palavras - chave: Redes Sociais, SNA, comunicação, cooperação, classificador bayesiano, predição de falhas e métricas de redes sociais.

ABSTRACT

Fontoura, Vidal D. FAILURE PREDICTION IN OPEN SOURCE PROJECTS BASED ON SOCIAL NETWORKS METRICS. Final Work – Graduation In Technology Systems for the Internet. Federal Technological University of Paraná. Campo Mourão 2011.

This work was developed with the goal of analyze human factors (communication and cooperation) as parameters for failure prediction, seeing that problems with communication and cooperation can lead to failure of projects. Many works of software engineering analyze communication and cooperation among the developers, by extraction data from control version systems and of communication mechanisms, among then, some use social networks and SNA to analyze the communication and the cooperation among the developers. The experiment realized in this work, consisted in the generation of matrices (developers vs developers) based on data of communication and cooperation extracted from the repository of open source software, this matrices represents social networks of the developers, then was calculated the values of metrics of SNA, and this metrics were utilized in the training set and validation set of Bayesian classifiers that actually realizes the prediction and lastly was used the statistical measures Recall/Precision/F-measure to evaluated the performance of the predictions. From the experiment realized was possible verify that cooperation can be used to failure predict, and that communication may not get a satisfactory performance to failure prediction.

Key - Words: Social Networks, SNA, communication, cooperation, Bayesian classifier, prediction failure, and metrics of social networks.

LISTA DE SIGLAS

ASF - Apache Software Foundation.

SNA - Social Network Analysis (Análise de rede sociais).

SVN - Servidores *Subversion* controladores de versão.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

SUMÁRIO

1. INTRODUÇÃO

Pesquisadores de engenharia de software têm uma variedade de abordagens para criar redes sociais e estudar fenômenos da engenharia de software. Dentre elas, Meneely e Williams(2011) mencionam aquelas baseadas em mineração de *logs* de sistemas gerenciadores de versão (Bird et al. 2008),(Meneely et al. 2010), (Nia et al. 2010),(Shin et al. 2010), (Wolf et al. 2009) e mineração de mensagens de diferentes meios de comunicação, (Meneely et al. 2010), (Wolf et al. 2009), (Biçer et al. 2011).

Dentre os diversos fenômenos que podem ser estudados na engenharia de software, destacam-se aqueles relacionados à comunicação e cooperação de membros de uma ou mais equipes de software no desenvolvimento de suas tarefas. Para analisar aspectos de cooperação e comunicação é possível utilizar recursos relacionados à criação e geração de redes sociais, (Meneely e Williams,2011). O uso de SNA (*Social Network Analysis*) possibilita descrever, caracterizar e entender redes sociais por meio da exploração de suas propriedades, (Magdaleno et al. 2010).

Assim, pesquisadores têm conduzido experimentos na tentativa de descobrir abordagens para predição de falhas utilizando técnicas de SNA. Zimmerman e Nagappan (2008) fazem predição sobre grafos de dependência em código fonte; Wolf et al. (2009) utilizam dados extraídos de mensagens de *commits* de sistemas de controle de versão (SVN) e comentários existentes nestes *commits* para construir redes sociais; Meneely et al. (2008) aplicam métricas de SNA em redes sociais construídas a partir dos arquivos modificados pelos desenvolvedores. Biçer et al. (2011) estuda qual o benefício de utilizar SNA para analisar redes sociais de comunicação extraídas de repositórios de tarefas (*issues-trackers)* para predizer falhas.

Diante deste cenário este trabalho estuda a comunicação e a cooperação, considerando as métricas de SNA e redes sociais utilizadas nos trabalhos relacionados, com o objetivo de verificar se é possível melhorar a predição de falhas em projetos de software livre. Foi selecionado o projeto Felix GOGO[1] do repositório

[1] Felix GOGO: É um sub-subprojeto do projeto Apache Felix, que é uma implementação de um Shell padrão para ambientes baseados em OSGi.

ASF (*Apache Software Foundation*). Esta escolhab foi feita a partir de uma investigação prévia, que permitiu identificar que é possível minerar informações sobre falhas, comunicação e cooperação.

Neste repositório a comunicação foi minerada a partir das mensagens trocadas nos *mailing-lists* de desenvolvedores e fóruns de discussão. A cooperação foi extraída dos arquivos modificados por um desenvolvedor (*commits*).

Uma vez mineradas estas informações, foram geradas redes sociais representadas por matrizes que relacionam os desenvolvedores e suas interações. As matrizes foram analisadas com métricas de SNA e as métricas foram utilizadas como entrada para o modelo de predição de falhas.

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta os conceitos de comunicação, cooperação e métricas de SNA comparando as abordagens dos trabalhos relacionados. O Capítulo 3 descreve a metodologia de coleta de dados para a realização do estudo empírico, as redes sociais que serão criadas e como será realizada a predição de falhas. O Capítulo 4 apresenta os resultados O Capítulo 5 apresenta uma discussão sobre os resultados. O Capítulo 6 apresenta as ameaças de validação. O Capítulo 7 apresenta os trabalhos futuros e por fim as referencias.

2. REFERENCIAL TEÓRICO

Com o objetivo de predizer falhas em projeto de software livre com base nas estruturas de comunicação e de cooperação dos desenvolvedores, foi realizada uma revisão bibliográfica. Neste Capítulo serão abordados conceitos de cooperação, comunicação, métricas de SNA e classificadores bayesianos utilizados para predição de falhas.

2.1 COOPERAÇÃO

Esta seção apresentará como os trabalhos relacionados que utilizam dados de cooperação entre os desenvolvedores criam redes sociais e utilizam métricas de SNA para interpretar dados destas redes e predizer falhas.

Wolf et al. (2009) desenvolveu um estudo para predizer falhas utilizando SNA. No projeto de software estudado as tarefas e os objetivos de cada *release* são agrupados em itens de trabalho (*work itens)* e atribuídos a equipes de desenvolvimento. Os *work itens* são desenvolvidos de forma paralela por mais de uma equipe sendo necessária a sua integração para a definição de uma *build* do projeto. Desta forma as equipes tem que cooperar entre si para concluírem os objetivos das *releases*. Assim, conexões são criadas em uma rede social considerando aqueles desenvolvedores que trabalharam em um mesmo *work item.*

Já Meneely et al. (2008) analisou o histórico dos arquivos de um projeto verificando os responsáveis pelas atualizações de cada arquivo. Identificando todos os desenvolvedores que atualizaram um arquivo X, ele cria conexões entre esses desenvolvedores. Estas conexões indicam quais foram os desenvolvedores que cooperaram para o desenvolvimento de um artefato. Por exemplo, se dois desenvolvedores, A e B, atualizaram um mesmo arquivo, podemos concluir que A e B interagiram indiretamente e/ou diretamente, visto que compartilharam atualizações em um mesmo arquivo.

Tanto Wolf et al. (2009) e Mennely et al. (2008) identificam quais foram o desenvolvedores que trabalharam em conjunto, atualizando arquivos

compartilhados. Isto foi possível investigando o *log* do sistema gerenciador de versão de documentos, a partir da mineração dos históricos dos arquivos que foram modificados pelos desenvolvedores.

Fuks et al. (2003), define cooperação como membros de um grupo que operam em conjunto para manipular e organizar informações, construindo e refinando objetos. Segundo Steinmacher et al. (2010) é possível identificar a cooperação quando um ou mais desenvolvedores interagem com artefatos compartilhados de maneira síncrona ou assíncrona. Assim, mesmo que Wolf et al. (2010) e Mennely et al. (2008) não tenham tratado explicitamente sobre cooperação em seus trabalhos, é possível inferir que os dados minerados são relativos a cooperação, uma vez que eles observaram o histórico de atualização dos arquivos e verificaram quais desenvolvedores trabalharam em conjunto em arquivos compartilhados.

Desta forma, este trabalho analisa informações que refletem a cooperação entre os desenvolvedores. Estas informações serão extraídas dos históricos de atualizações dos arquivos contidos nos repositórios *SVN* da *ASF*.

2.2 COMUNICAÇÃO

Esta seção apresentará como os trabalhos relacionados consideram a comunicação no seu contexto, e como ela será considerada neste trabalho.

Wolf et al. (2009) utilizou os comentários deixados nos *work itens* como principal forma de comunicação entre desenvolvedores. O seu estudo considera que os desenvolvedores que comentam em um *work item* leram todos os comentários que podem ter sido deixados por outros desenvolvedores anteriormente. Identificando os desenvolvedores que comentaram é possível gerar a estrutura de comunicação que se formou com base no comportamento dos comentários deixados pelos desenvolvedores.

O estudo desenvolvido por Biçer et al. (2011) propôs a inclusão de métricas de *SNA* para predição de falhas em software. Para calcular os valores para as métricas foram analisados dois projetos que tem uma estrutura de comunicação similar. Comentários feitos em repositórios de tarefas dos projetos foram o meio de

comunicação considerado no estudo, pois neste local os desenvolvedores expressavam suas opiniões e por fim acabavam se comunicando.

Fuks et al. (2003) cita que pessoas se comunicam com intenção de transmitir informação. Ferramentas de comunicação mediadas por computador dão suporte a interações entre os participantes, podendo gerenciar as transições de estados, os eventos de diálogo e os compromissos de cada participante.

Este trabalho considera os meios de comunicação disponíveis em projetos da *ASF* que em alguns casos incluem *mailing-lists*.

2.3 MÉTRICAS SNA

Esta seção apresentará os trabalhos relacionados que consideram o uso de SNA com o intuito de predizer falhas. Como o desenvolvimento de software na maioria dos casos é um trabalho para ser feito em grupo, torna-se importante estudar cooperação e comunicação. Como visto nas seções 2.1 e 2.2, a literatura indica que estudos podem ser realizados analisando a comunicação e cooperação para predizer falhas. Este trabalho propõe uma análise da comunicação e cooperação utilizando métricas de redes sociais (*SNAs*) com intuito de obter dados que possibilitem a identificação de padrões para previsão de falhas, uma vez que uma rede social pode ser analisada com base em suas propriedades. (Santos et al. 2010)

A Quadro 1 relaciona os estudos e as métricas de *SNA* utilizadas nos trabalhos descritos nas seções 2.1 e 2.2. A definição de cada uma das métricas pode ser encontrada nos trabalhos correlatos.

	Wolf et. al (2009)	Mennely et. al (2008)	Serdar Biçer et al (2011)	Zimmermann et. al (2008)
Centrality	x	x	x	x
Density	x		x	
Strutural Holes	x			x
Diameter			x	
Clustering Coefficient			x	
Bridge			x	
Characteristic Path Length			x	
Ego Networks				x
Global Networks				x

Quadro 1 – Comparação de métricas de SNA utilizadas nos trabalhos correlatos.
Fonte: Autoria Própria

Observa-se no Quadro 1 que diferentes métricas foram utilizadas para medir as propriedades extraídas das redes sociais dos trabalhos relacionados identificados. Em todos os trabalhos estes valores calculados a partir da SNA foram utilizados como entrada nos diferentes modelos de predição.

Para este trabalho foram escolhidas dentre as métricas apresentadas no Quadro 1, as métricas que foram possíveis de ser calculadas com a ferramenta de SNA *Ucinet*. Portanto as métricas calculadas foram:

- **Métricas de centralidade**: Segundo Menneley et al. (2008) métricas de centralidade são utilizadas para medir o quanto os desenvolvedores são indiretamente conectados ao resto da rede. A seguir uma breve descrição de algumas métricas de centralidade de acordo com a definição de Borgatti et al. (2002).
    - ***Degree***: É o numero de nós adjacentes a um nó, *degree* é o numero de vizinhos que um nó tem na rede.
    - ***Closeness***: É o numero de passos requerido para um nó alcançar todos os outros nós. Maior valor para um nó significa que é mais fácil para o nó divulgar informações através da rede.
    - ***Betweenness***: O valor da centralidade *betweenness* de um nó x é o número de caminhos mais curtos entre pares de outros nós que percorrem x. É uma medida da influência de um nó sobre o fluxo de informações entre os outros nós, especialmente nos casos onde o fluxo de informações por uma rede basicamente segue o caminho mais curto disponível.
    - ***Eigenvector***:É a métrica expressa em porcentagem, que mede a centralidade de um nó baseada nas ligações a outros nós que são

centrais.

- **Métricas de ego-network:** Segundo Borgatti et, al (2002) são métricas que isolam um nó e levam em consideração apenas este nó e aqueles nós que estão diretamente conectados a ele. A seguir uma breve descrição de algumas métricas de ego-network de acordo com a definição de Borgatti et, al (2002) :
    - ***Size***: É o numero de nós que estão a um passo de um nó.
    - ***Ties***: É o numero de conexões entre todos os nós da *ego-network*.
    - ***Pairs***: É o numero de conexões possíveis entre cada nó da *ego-network*.
    - ***Density***: É o número de *Ties* dividido pelo o número de *Pairs*. É a porcentagem de todas as possíveis conexões que está realmente presente.
    - ***Average geodesic distance***: É a media dos caminhos mais curtos entre todos os pares conectados na *ego-network*.
    - ***Diameter***: É o comprimento do caminho mais longo entre atores conectados.
    - ***Weak Components***: Um *weak component* é o maior número de atores que estão conectados.
    - ***Weak Components Divided By Size***: É o valor de *Weak Components* dividida pela *Size***.**
    - ***2-step reach***: É a porcentagem de todos os nós da rede toda que estão conectados a dois passos da *ego-network*.
    - ***Reach efficiency***: É o valor da medida de *Two-setp reach* dividida pela medida Size.
    - ***Brokerage***: Numero de pares não conectados diretamente.
    - ***Normalized brokerage***: É o valor de *Brokerage* dividido pelo numero de *Pairs***.**
    - ***Betweenness*:** Mesma definição de centralidade *Betweenness*, mas aplicada a *ego-network*. É o numero de caminhos mais curtos entre pares de *ego-network*.
    - ***Normalized Betweenness***: É o valor de *Betweenness* dividido pelo o maior valor possível de *Betweeenness*.

- **Métricas de structural holes**: De acordo com a definição de Hanneman e Riddle (2005), são métricas que se referem a alguns aspectos muito importantes de vantagem/desvantagem posicional de nós, relativos à forma que estão incorporados nas *ego-networks.* As métricas de *ego-network* de acordo com a definição Hanneman e Riddle (2005) são:
    - ***Degree*:** É o valor da centralidade *Degree* relativo a ego-network
    - ***Effective***: É o numero de nós que uma *ego-network* tem, menos o numero médio de conexões que cada nó tem entre si.
    - ***Efficiency***: É a proporção do tamanho da *ego-network* em relação ao tamanho real da rede completa.
    - ***Constraint*:** é uma medida resumo em que as conexões ego são os nós que estão ligados uns aos outros
- **Densidade da rede:** Segundo Borgatti et al. (2002) densidade é proporção de arestas relativas ao o numero total de possíveis.
- **Coeficiente de Clusterização:** Segundo Borgatti et al. (2002) é uma medida de grau de que cada nó tende a se clusterizar a outro.

2.4 MODELO DE PREDIÇÃO

Esta seção irá apresentar os modelos de predição utilizados nos trabalhos relacionados.

Wolf et al. (2009) utilizou um classificador/filtro bayesiano para avaliar se o conjunto de métricas selecionadas em seu estudo, obteria sucesso em predizer falhas. Para predizer se um *build* X irá falhar, um classificador foi treinado, com os resultados de um conjunto de *builds* de um projeto e os valores das métricas de SNA respectivos a cada *build*. Feito o treinamento, o próximo passo consistiu em inserir os valores das métricas relativas ao *build* X e o classificador respondeu a probabilidade do *build* X obter sucesso ou falha baseando-se no treinamento. Wolf et al. (2009) comparou o resultado que o classificador respondeu, com o resultado real e calculou os coeficientes de *Recall* e *Precision* para avaliar a qualidade da classificação.

Para predizer Biçer et al. (2011) também utilizou um classificador bayesiano

mais especificamente o algoritmo de *Naive Bayes,* pois apresenta um desempenho significativamente melhor do que outros algoritmos para predição de defeitos (Menzies et al. 2007). Biçer et al.(2011) utilizou como entrada para o algoritmo, valores das métricas de SNA que ele calculou a partir de redes sociais extraídas dos comentários das tarefas. Para validar os resultados que o algoritmo respondeu foi utilizada uma analise de custo-benefício. Biçer et al. (2011) utilizou WEKA Software para executar o algoritmo e realizar experimentos.

Assim como Biçer et al. (2011) e Wolf et al. (2009) este trabalho utilizará a abordagem de classificação bayesiana, utilizando a ferramenta WEKA Software para realizar o procedimento de predição, no entanto, irá comparar os resultados de *Recall/Precision* com diferentes configurações do conjunto de treinamento e validação de dados, combinando estes treinamentos com três diferentes algoritmos. Nas seções 2.4.1, 2.4.2 e 2.4.3 são apresentados detalhes gerais sobre cada um dos algoritmos testados.

2.4.1. Algoritmo *Naive Bayes*

Segundo Martins e Costa(2009) este algoritmo é um dos mais simples classificadores probabilísticos. O modelo construído por este algoritmo é um conjunto de probabilidades que são estimadas pela contagem da freqüência de cada valor de característica para as instâncias dos dados de treino. O classificador estima a probabilidade de uma nova instância pertencer a uma classe específica, baseada no produto das probabilidades condicionais individuais para os valores característicos da instância.

O calculo utiliza o teorema de Bayes por essa razão que o algoritmo é denominado um classificador de Bayes. O algoritmo também é denominado *Naive*, uma vez que todos os atributos são independentes, dado o valor da variável da classe. Apesar deste pressuposto, o algoritmo apresenta um bom desempenho em muitos dos cenários de predição de classes.

2.4.2. Algoritmo *Bayes-Net*

Segundo Heckerman(1996) este algoritmo codifica relações probabilísticas

entre variáveis de interesse. Quando usado em conjunto com técnicas de estatística, o algoritmo *Bayes-Net* tem varias vantagens para análise de dados. O modelo codifica dependências entre todas as variáveis e possibilita entradas de valores nulos. Uma rede bayesiana pode ser usada para aprender relações casuais e, portanto, pode ser usada para ganhar compreensão sobre o domínio do problema e para prever as conseqüências da intervenção.

2.4.3. Algoritmo *Bayesian Logistic Regression*

Segundo Genkin et al. (2007) este algoritmo utiliza uma transformada de Laplace para evitar *overfitting* definido por Everitt (2002) é quando modelos estatísticos contém erros aleatórios e ruído em modelos excessivamente complexos. Com isto o algoritmo deve produzir esparsos modelos preditivos para dados nominais. Esta abordagem produz compactos modelos preditivos, pelo menos tão eficazes quanto os produzidos por classificadores que utilizam vetor de maquina ou de regressão logística.

2.5 AVALIAÇÂO DOS MODELOS DE PREDIÇÂO

Biçer et al. (2011) menciona que a melhoria da performance dos modelos de predição não esta associado a implementação de novos algoritmos, mas sim, na melhoria da qualidade do conjunto de dados minerados ou até mesmo, de métricas que ainda não foram utilizadas para predição.

Desta forma, não faz parte do escopo deste trabalho, discutir se a utilização de rede bayesiana oferece a melhor forma de predizer falhas em projetos de software. Espera-se com este trabalho testar se a predição de falhas pode ser potencializada a partir da melhoria do conjunto de dados de entrada para o modelo de predição. Isto será obtido com a mineração de dados adicionais em relação aos trabalhos relacionados, que serão explicados detalhadamente na Seção 3.1.1, 3.1.2, 3.1.3 e 3.2 e a partir da combinação do conjunto de métricas apresentadas na Quadro 1.

Dado que os modelos de predição são criados para tentar predizer dados diferentes daqueles foram utilizados para construir-lo, existe a necessidade de

avaliar o desempenho do modelo em realizar predições corretas. A avaliação normalmente é feita dividindo a amostra de dados disponível em dois conjuntos: um de treinamento e outro de validação. O modelo é construído com base no conjunto de treinamento, depois é aplicado aos dados do conjunto de validação, e realiza predições com base nos dados que foram utilizados no conjunto de treinamento. Esta técnica serve para avaliar o modelo e estimar a incerteza das suas previsões. (Han et al. 2006).

Existem varias medidas para avaliar as predições realizadas por modelos de predição, a seguir são apresentadas algumas medidas, e maneiras para avaliar os modelos de predição:

- **Matriz de Confusão:** É uma tabela para visualização dos resultados utilizada para avaliar modelos de predição que utilizem algoritmos de classificação. Cada linha da matriz representa as instâncias preditas de uma classe, enquanto cada coluna da matriz representa as instâncias reais de uma classe, assim a diagonal principal representa os valores que foram preditos corretamente, e a diagonal secundária representa os valores que foram preditos incorretamente. Para modelos que tem como objetivo realizar predições para duas classes deve-se considerar que a matriz de confusão é uma tabela com duas linhas e duas colunas que registra o número de *True Negatives* (TN), *False Positives* (FP), *False Negatives* (FN) e *True Positives* (TP). (Anacleto 2009). A Figura 1 apresenta o esquema de uma matriz de confusão, neste caso as classes são *Positive* e *Negative*.

		Valore Reais	
Valores Preditos		True Positive	False Positive
		False Negative	True Negative

TN – Instâncias negativas classificadas como negativas
FP – Instâncias negativas classificadas como positivas
FN – Instâncias positivas classificadas como negativas
TP – Instâncias positivas classificadas como positivas

Figura 1 – Exemplo de Matriz de Confusão
Fonte: Autoria Própria

- **Recall, Precision:** *Recall* e *Precision* são duas medidas de avaliação amplamente utilizadas para avaliar modelos de predição. *Recall* pode ser considerada como uma medida de completude enquanto *Precision* pode ser considerada como uma medida de precisão ou de fidelidade. *Recall* mede a capacidade de um modelo encontrar o que se quer e *Precision* mede a capacidade do mesmo modelo em rejeitar o que não se quer. (Anacleto 2009). Para calcular *Recall/Precision* deve se utilizar estas medidas a partir da matriz de confusão, com as seguintes fórmulas.

$$Precision = \frac{TP}{FP + TP} \qquad Recall = \frac{TP}{FN + TP}$$

- **F-Measure**: É definida como a média harmônica entre os valores de *Recall* e *Precision.* Pode ser obtida pela seguinte formula:

$$F - Measure = \frac{2 * Recall * Precision}{Recall + Precision}$$

Segundo Delgado et al. (2010) *F-Measure* combina as métricas de *Recall* e *Precision*. O resultado *F-Measure* é um indicativo de que, quanto mais próximo de 1 melhor é o desempenho do modelo de predição, e resultados mais próximos de 0, demonstram a imprecisão do modelo de predição.

3. METODOLOGIA

A metodologia deste trabalho foi previamente publicada no VIII Simpósio Brasileiro de Sistemas Colaborativos em Paraty – RJ em 2011.

Baseando-se na metodologia descrita por Fontoura et al. (2011) para alcançar o objetivo de predizer falhas em projetos de software livre, por meio da mineração de dados de comunicação e cooperação dos desenvolvedores envolvidos em um projeto, três questões de pesquisa foram investigadas neste trabalho: (Q1) Métricas de SNA aplicadas a redes sociais extraídas de mecanismos de comunicação dos desenvolvedores podem auxiliar a predição de falha em projetos de software? (Q2) Métricas de SNA aplicadas a redes sociais extraídas de mecanismos de cooperação dos desenvolvedores podem auxiliar a predição de falha em projetos de software? (Q3) Resultados combinados a partir da analise das redes sociais criadas podem melhorar os resultados de predição? Para responder estas questões um experimento foi realizado contendo os seguintes passos: Coleta de Dados, Análise das Redes Sociais e Predição do Estado das Builds.

3.1 COLETA DE DADOS

A coleta de dados foi realizada em um projeto da ASF. Assim, o primeiro passo consistiu em selecionar qual projeto seria estudado neste trabalho. Após investigação do conjunto de projetos da ASF e verificado se existia a possibilidade de extrair informações relativas à comunicação, cooperação e o histórico dos estados das *builds*.

A ASF utiliza uma ferramenta de integração continua chamada *Hudson*. O agendamento dos *builds* pode variar de projeto para projeto. Alguns executam o agendamento a cada compilação, alguns diariamente, outros a cada mês. Uma *build* constrói o código fonte compilando-o a partir da ultima versão do código fonte encontrada no SVN. Ele também executa os testes automatizados verificando se esta nova construção continua válida em relação aos testes já criados. Quando ocorre algum erro durante a compilação e execução dos testes, o Hudson indica que a *build* falhou, caso contrário, ela esta no estado de sucesso.

A escolha do projeto foi feita a partir da analise do histórico das *builds* gerados pela configuração de agendamento do *Hudson*. A grande maioria dos projetos não guardam o histórico completo, somente apresentando as ultimas 5 *builds.* Isto limitou a possibilidade de escolha dentre os projetos da ASF.

Figura 2 – Pagina de histórico de construções para o projeto maven-parent (*builds*)
Fonte: Autoria Própria

A Figura 2 mostra o histórico de construção do projeto *maven-parent*. Pode-se observar do lado esquerdo uma caixa de diálogo indicando o histórico de *builds* contendo somente 5 entradas. Cada bola representa a execução de uma *build.* Também é possível observar o seu *timestamp* que indica a data que a *build* foi construída. As bolas de cor azul indicam que o *build* foi executado com sucesso. A cor vermelha indica falha.

Desta forma, projetos que não continham um intervalo com duração de tempo grande de *builds* com sucesso e falhas foram descartados. Considerando este pré-requisito o projeto selecionado foi o Felix GOGO.

Após a seleção do projeto para o experimento, iniciou-se a extração dos dados do projeto Felix GOGO. Foram minerados todos os dados de comunicação e cooperação entre as datas de 25-07-2009 e de 19-09-2011. Este intervalo foi definido com base no histórico de *builds* do projeto.

Foram identificados 26 intervalos com duração de um mês aproximadamente. Cada intervalo é caracterizado como uma *build* do projeto e foi classificado como sucesso ou falha. Destes 26 intervalos, 13 *builds* encontravam-se no estado de sucesso e 13 no estado de falha.

Foram coletadas 765 mensagens da lista de emails do projeto, enviadas por 86 usuários. Para a cooperação foram minerados 4369 *commits* do SVN do projeto, realizados por 25 usuários. Observa-se preliminarmente que existem mais usuários discutindo a respeito do projeto do que propriamente realizando *commits*. Provavelmente isto ocorreu porque usuários da lista de email podem estar participando do projeto para reportar *bugs* existentes ou simplesmente acompanhar o seu desenvolvimento. Outro fator de restrição é que normalmente, projetos de software livre, contêm uma lista de usuários que podem realizar *commit* diretamente no SVN do projeto.

Para a coleta de dados foram construídos 2 mineradores que extraíram as informações dos projetos e as armazenavam em um banco de dados local apresentado na Figura 3. Os dados de comunicação minerados foram: os autores, os assuntos, as datas e os conteúdos das mensagens. Os dados de cooperação foram: os autores, as revisões, as datas, e os arquivos adicionados/modificados/excluídos dos *commits*. A Figura 3 apresenta o banco de dados que foi construído para armazenar estas informações.

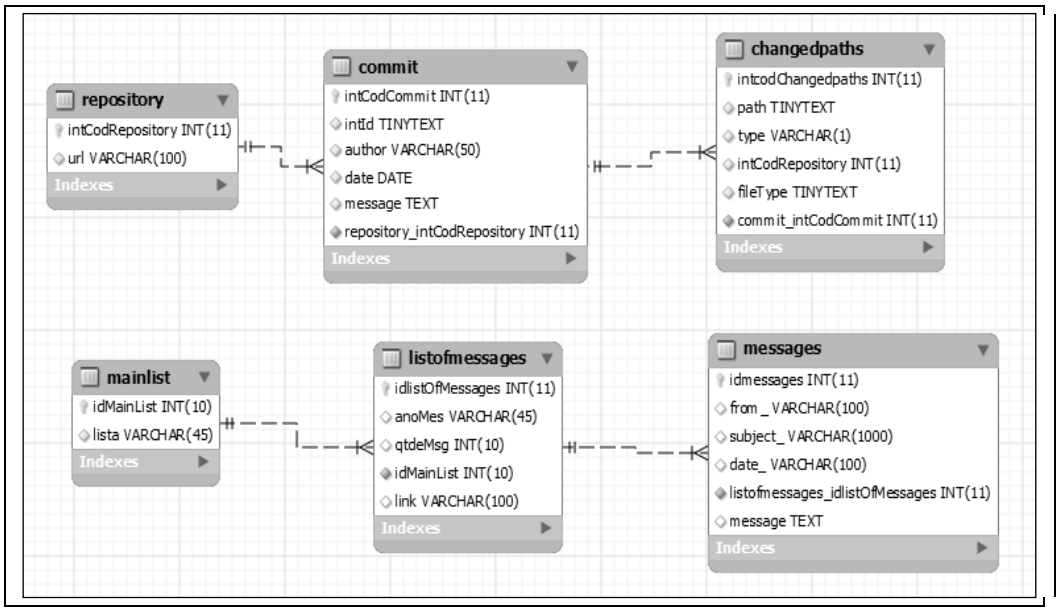
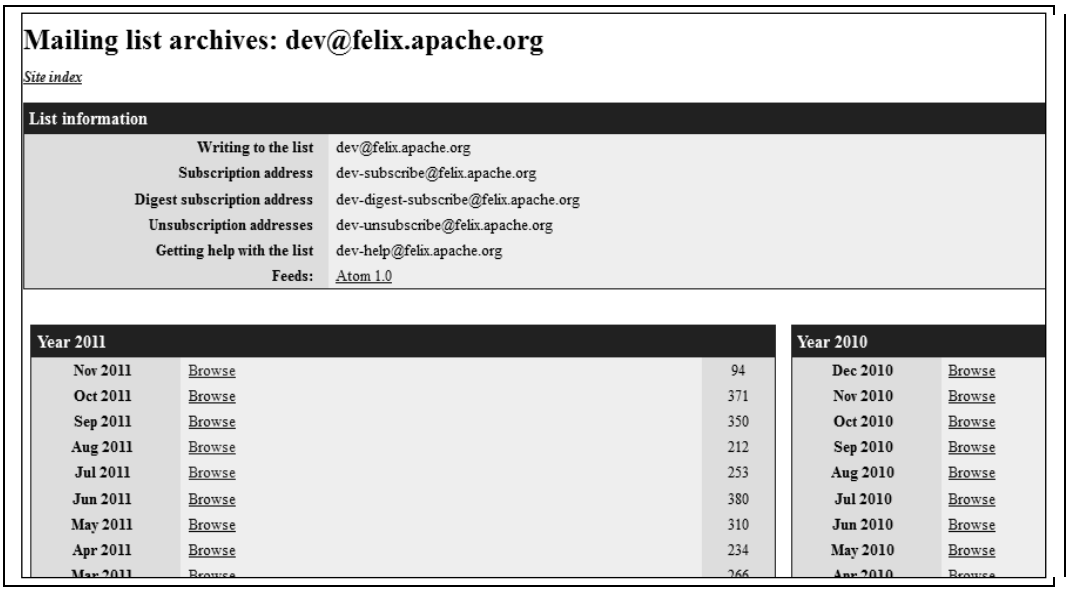


Figura 3 – Esquema do banco de dados para armazenar os dados de cooperação e comunicação
Fonte: Autoria Própria

As seções a seguir apresentam como os dados foram retirados das fontes de informações, como eles foram minerados e armazenados no esquema da Figura 3, e como estes dados foram utilizados para gerar matrizes para cálculo das métricas SNA.

3.1.1 Dados De Comunicação e Geração De Matrizes De Comunicação

Dados referentes à comunicação foram extraídos de listas de emails (*mailing-list*) hospedados nos portais dos projetos da *ASF*. As listas contêm os usuários que se comunicaram enviando mensagens para lista de email do projeto, a data, os assuntos (*subject*) e os conteúdos das mensagens. A Figura 4 apresenta a pagina principal do arquivo de *mailing-list* do projeto Felix GOGO.



Mailing list archives: dev@felix.apache.org

Site index

List information

Writing to the list	dev@felix.apache.org
Subscription address	dev-subscribe@felix.apache.org
Digest subscription address	dev-digest-subscribe@felix.apache.org
Unsubscription addresses	dev-unsubscribe@felix.apache.org
Getting help with the list	dev-help@felix.apache.org
Feeds:	Atom 1.0

Year 2011

Nov 2011	Browse	94
Oct 2011	Browse	371
Sep 2011	Browse	350
Aug 2011	Browse	212
Jul 2011	Browse	253
Jun 2011	Browse	380
May 2011	Browse	310
Apr 2011	Browse	234
Mar 2011	Browse	266

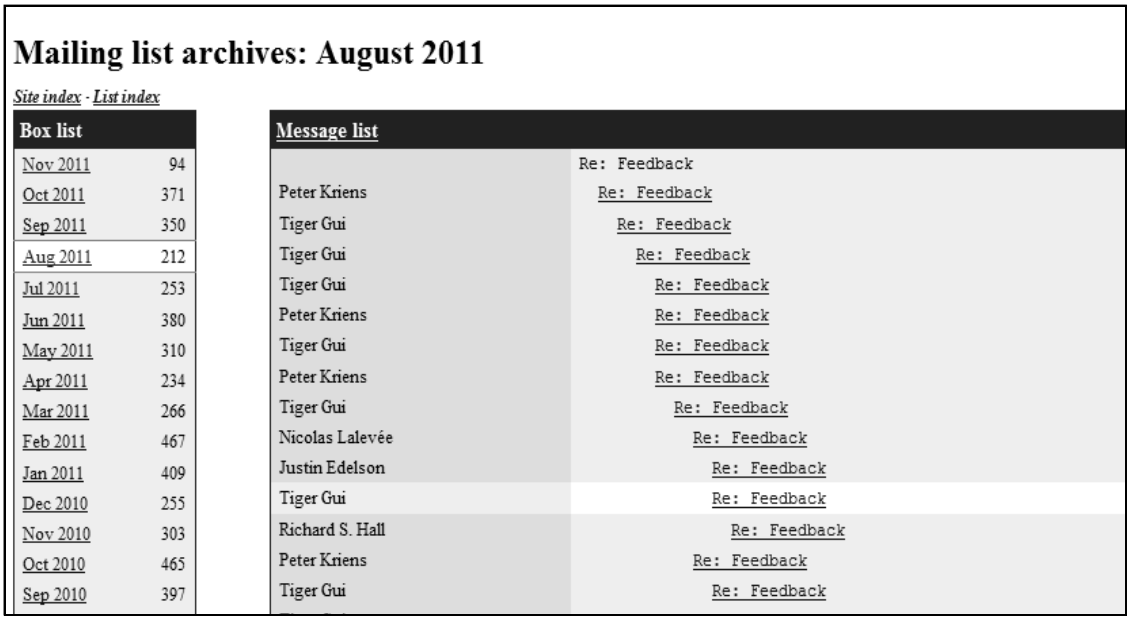
Year 2010

Dec 2010	Browse
Nov 2010	Browse
Oct 2010	Browse
Sep 2010	Browse
Aug 2010	Browse
Jul 2010	Browse
Jun 2010	Browse
May 2010	Browse
Apr 2010	Browse

Figura 4 – Pagina principal do arquivo de mailing list do projeto Felix GOGO
Fonte: Autoria Própria

Para extrair as informações o minerador requisita a pagina apresentada na Figura 4, utilizando a seguinte url: mail-archives.apache.org/mod_mbox/felix-dev/. A partir do HTML, são encontrados os *links* para as mensagens de cada mês/ano, realizando uma requisição para cada *link*. Após a requisição o minerador recebe o código HTML referente ao *link* recebido, que por sua vez contêm o *link* de todas as mensagens que ocorreram em um determinado mês. A Figura 5 exemplifica as mensagens mineradas para o mês de agosto de 2011.

Assim todas as mensagens e todas as respostas são armazenadas na estrutura de banco de dados apresentada na Figura 3.



Mailing list archives: August 2011

Site index · *List index*

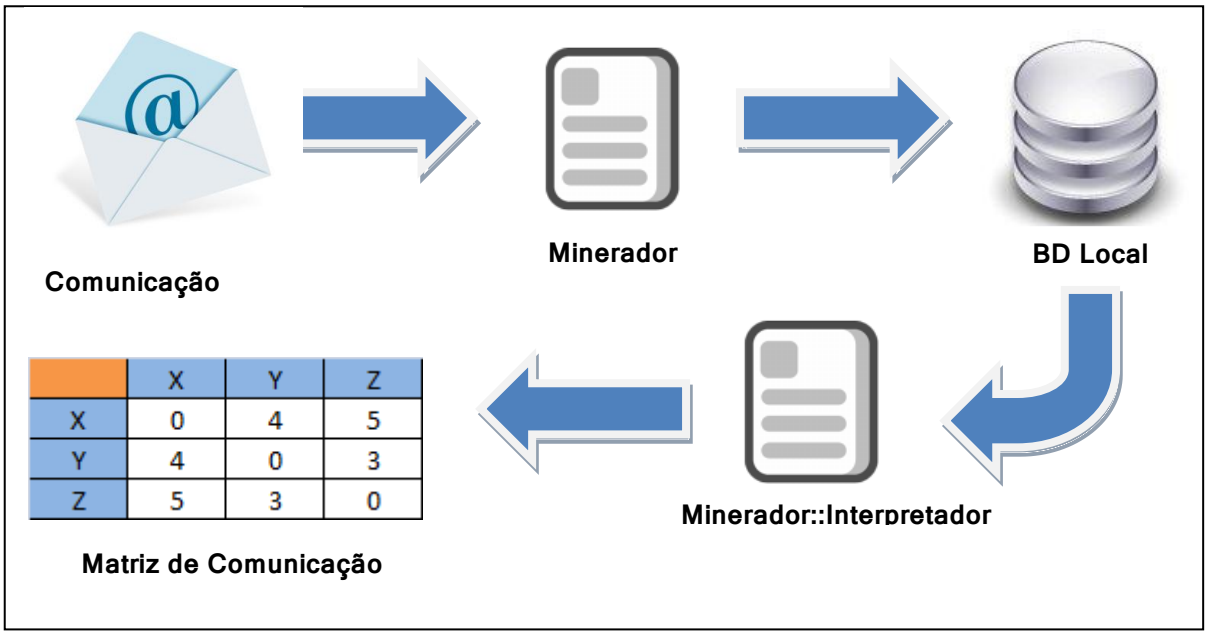
Box list	
Nov 2011	94
Oct 2011	371
Sep 2011	350
Aug 2011	212
Jul 2011	253
Jun 2011	380
May 2011	310
Apr 2011	234
Mar 2011	266
Feb 2011	467
Jan 2011	409
Dec 2010	255
Nov 2010	303
Oct 2010	465
Sep 2010	397

Message list	
	Re: Feedback
Peter Kriens	Re: Feedback
Tiger Gui	Re: Feedback
Tiger Gui	Re: Feedback
Tiger Gui	Re: Feedback
Peter Kriens	Re: Feedback
Tiger Gui	Re: Feedback
Peter Kriens	Re: Feedback
Tiger Gui	Re: Feedback
Nicolas Lalevée	Re: Feedback
Justin Edelson	Re: Feedback
Tiger Gui	Re: Feedback
Richard S. Hall	Re: Feedback
Peter Kriens	Re: Feedback
Tiger Gui	Re: Feedback

Figura 5 – Pagina que indexa todas as mensagens do mês de Agosto de 2011
Fonte: Autoria Própria

Para implementação do minerador foram utilizadas expressões regulares para encontrar os *links* e encontrar as informações sobre as mensagens. Isto foi importante para a construção dos *links* de visitação, pois os o endereço para estes utiliza caminhos relativos.

A Figura 6 apresenta os passos para a extração e criação das redes sociais, iniciado a partir do minerador, após a conclusão da tarefa de mineração e armazenamento dos dados de comunicação, são criadas as matrizes desenvolvedor *versus* desenvolvedor similares à matriz apresentada na Tabela 1.



	X	Y	Z
X	0	4	5
Y	4	0	3
Z	5	3	0

Figura 6 - Processo de extração e analise dos dados de comunicação
Fonte: Autoria Própria

Para gerar as redes sociais de comunicação (matrizes) foram encontradas todas as mensagens que ocorreram no intervalo tempo de cada *build*, em seguida foram identificados todos os autores que responderam a cada mensagem.

Para todos os participantes de uma mesma mensagem é incrementada uma interação entre eles em uma matriz. A matriz relaciona todos os autores de mensagens enviadas/respondidas dentro do intervalo de tempo da *build* indicando a quantidade de mensagens trocadas entre eles.

Para cada um dos 26 intervalos foi gerado uma matriz desenvolvedor *versus* desenvolvedor que representa a rede social de comunicação entre os desenvolvedores no período respectivo ao intervalo de tempo da *build*.

A Tabela 1 apresenta um exemplo onde o desenvolver *Richard s.* trocou 6 mensagens com *Carsten zieg.* A matriz representa a rede social da comunicação dos desenvolvedores.

	yi xiao xia	richard s. l	peter krie	tiger gui t	nicolas lal	justin ede	carsten zi	felix mesc	jean-bapt	karl pauls
yi xiao xiao	0	0	0	0	0	0	0	0	0	0
richard s. ha	0	0	0	0	0	0	6	0	0	5
peter krien	0	0	0	0	0	0	0	0	0	0
tiger gui tig	0	0	0	0	0	0	0	0	0	0
nicolas lale	0	0	0	0	0	0	0	0	0	0
justin edels	0	0	0	0	0	0	0	0	0	0
carsten zieg	0	6	0	0	0	0	0	0	1	6
felix mesch	0	0	0	0	0	0	0	0	0	0
jean-baptis	0	0	0	0	0	0	1	0	0	1
karl pauls k	0	5	0	0	0	0	6	0	1	0

Tabela 1 - Exemplo de matriz desenvolvedor versus desenvolvedor
Fonte: Autoria Própria

3.1.2 Dados De Cooperação e Geração De Matrizes De Cooperação

Dados referentes à cooperação foram extraídos de repositórios *SVN* dos projetos da *ASF*. Os repositórios contêm dados de cada *commit* realizado indicando o seu autor, os arquivos alterados, a data e o número da revisão do projeto.

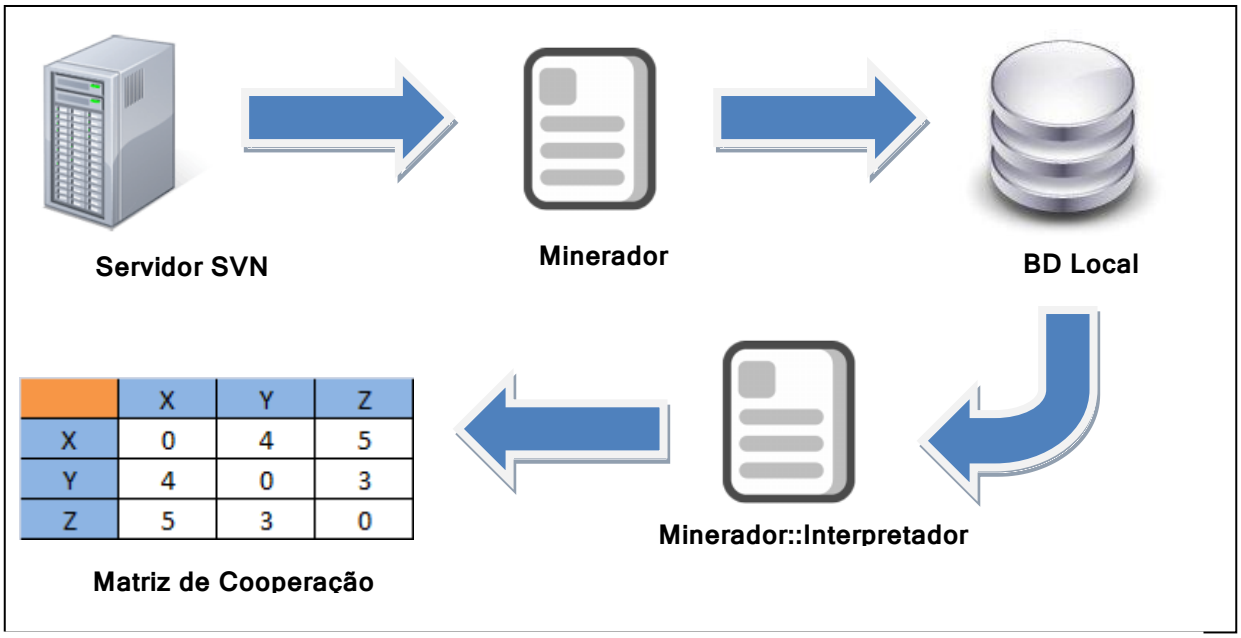
Similarmente ao processo de extração de dados da comunicação, a Figura 7, apresenta os passos para a extração dos dados de cooperação. Um minerador obtém informações do Servidor *SVN a*rmazenando-as em um banco de dados local, já apresentado na Figura 3.

O minerador foi implementando utilizando uma API Java de conexão com o

repositório SVN, chamada SVNKIT API Java Svn (2011). Está API fornece métodos para criar uma conexão com um servidor SVN e executar comandos.

Depois de feita a conexão, o minerador utiliza métodos que requisitam as linhas do registro dos *commits*, para cada linha é feita uma análise para encontrar as informações desejadas e inseri-las em um banco de dados local.

Outro minerador analisa os dados salvos no banco de dados local e cria as relações existentes entre os desenvolvedores a partir das alterações de artefatos compartilhados por eles. Assim, uma matriz é construída com os dados referentes à cooperação dos desenvolvedores. A matriz criada é idêntica a estrutura apresentada na Tabela 1.



	X	Y	Z
X	0	4	5
Y	4	0	3
Z	5	3	0

Figura 7 - Processo de extração e analise dos dados de cooperação
Fonte: Autoria Própria

Para gerar as redes de cooperação (matrizes) foram levantados todos os *commits* que ocorreram no intervalo de tempo de cada *build*, cada *commit* tem seu respectivo autor e lista de arquivos alterados. Para criar as redes sociais, foram analisados quais arquivos foram alterados por diferentes autores de *commits*. Para cada arquivo que foi alterado por dois autores diferentes, dentro de um mesmo intervalo de tempo de uma *build*, foram incrementadas interações entre os autores dos *commits* em matrizes desenvolvedor *versus* desenvolvedor.

Estas matrizes relacionam todos os autores de *commits* que realizaram alterações em arquivos em comum em um mesmo intervalo de tempo de uma *build*, e indicam o grau de interação com artefatos compartilhados entre os

desenvolvedores.

Para cada um dos 26 intervalos foi gerada uma matriz a qual representa a rede social de cooperação entre os desenvolvedores no período respectivo ao intervalo de tempo da *build*.

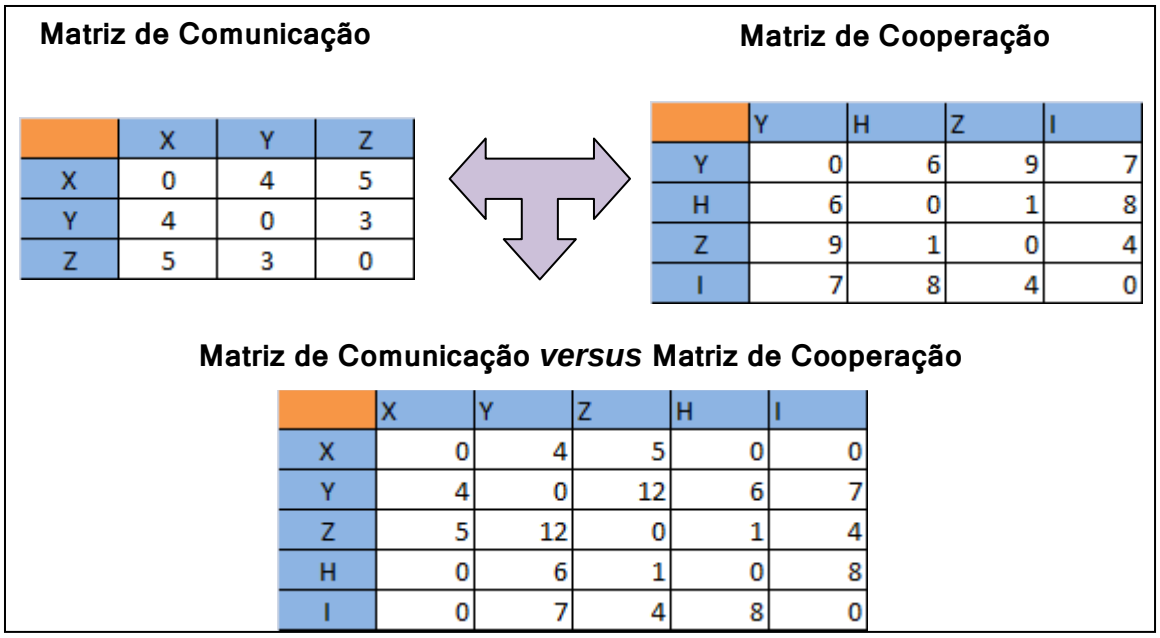
3.1.3 União Das Matrizes De Comunicação E De Cooperação

As matrizes de comunicação e cooperação foram criadas, da mesma maneira que foram criadas as matrizes anteriores, o diferencial é que tanto, as mensagens trocadas entre os autores como arquivos alterados pelos autores, foram levados em consideração. O objetivo da criação desta matriz foi somar a intensidade da comunicação com a intensidade de cooperação, resultando em uma matriz que representa uma rede social que reflete ambos os fatores.

A Figura 8 mostra um cenário de união das matrizes de cooperação e comunicação. Considere que os desenvolvedores X, Y e Z trocaram mensagens. Já os desenvolvedores Y, Z, H e I cooperaram no desenvolvimento de artefatos.

A união das matrizes será feita da seguinte maneira: se um desenvolvedor está contido nas matrizes de comunicação e cooperação, somam-se os valores correspondentes a suas interações de comunicação e cooperação. Por exemplo, na Figura 8, os desenvolvedores Y e Z estão contidos nas duas matrizes. Na matriz de comunicação ocorreram 3 trocas de mensagens. Já na matriz de cooperação os desenvolvedores interagiram com arquivos compartilhados 9 vezes. A matriz Comunicação *versus* Cooperação indica que ocorreram 12 interações entre estes desenvolvedores.

Para os casos onde os desenvolvedores estão contidos em apenas uma das matrizes, os valores de relacionamento existentes na matriz a qual ele pertence, é mantido. Já os relacionamentos que passaram a existir quando as matrizes foram unidas, ficam com os valores zerados. Observando a Matriz de Comunicação *versus* Cooperação, os valores do desenvolvedor X em relação aos desenvolvedores I e H estão zerados, porque I e H estão contidos apenas na Matriz de Cooperação enquanto o desenvolvedor X esta contido somente na Matriz de Comunicação.



	X	Y	Z
X	0	4	5
Y	4	0	3
Z	5	3	0

	Y	H	Z	I
Y	0	6	9	7
H	6	0	1	8
Z	9	1	0	4
I	7	8	4	0

	X	Y	Z	H	I
X	0	4	5	0	0
Y	4	0	12	6	7
Z	5	12	0	1	4
H	0	6	1	0	8
I	0	7	4	8	0

Figura 8 - Demonstração da união das matrizes de comunicação e de cooperação
Fonte: Autoria Própria

Para cada um dos 26 intervalos foi gerado uma matriz desenvolvedor *versus* desenvolvedor resultante da união de matrizes de comunicação e de matrizes de cooperação. Esta matriz resultante representa a rede social de comunicação e cooperação entre os desenvolvedores no período respectivo ao intervalo de tempo da *build*.

Este passo difere das abordagens de Wolf et al. (2009) e Biçer et al. (2011) porque nenhum dos dois autores gera uma rede social que abrange dados de comunicação e de cooperação.

3.2 ANÁLISES DAS REDES SOCIAIS

Para análise dos dados foi aplicado um conjunto de métricas referentes a redes sociais, conforme seção 2.3. A escolha das métricas foi feita baseada nos trabalhos relacionados e com a disponibilidade de cálculo pela ferramenta *Ucinet*. A Figura 9 apresenta um arquivo de saída do *Ucinet* relativo as medidas de centralidade de uma rede social, que pode ser obtida com o calculo de *Degree*, *Betweenness*, *Closeness* e *Eigenvector* entre outros (Wasserman e Faust 1994).

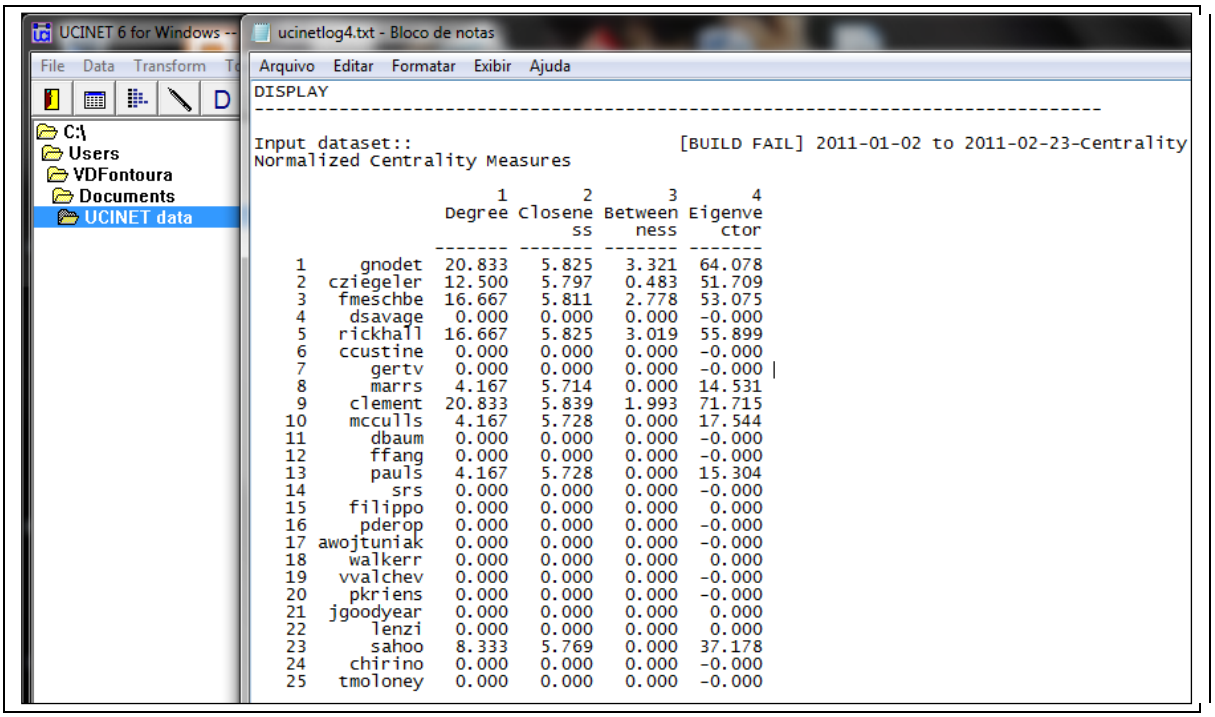


Figura 9 - Valores das métricas de centralidade da rede social.
Fonte: Autoria Própria

Os cálculos das métricas de redes sociais foram realizados de forma manual para cada uma das redes sociais (matrizes) visto que eram 26 redes de comunicação, 26 redes de cooperação, e 26 redes de comunicação e cooperação, que foram geradas conforme descrito nas seções 3.1.1, 3.1.2, 3.1.3.

Utilizando a ferramenta *Ucinet* para cada uma das redes de comunicação, cooperação, e comunicação e cooperação foram gerados 5 arquivos de saída diferentes que armazenavam os valores para as métricas.

Um arquivo continha os valores para métricas de centralidade, outro valores para as métricas de *ego-network*, e um terceiro armazenava valores para métricas de *structural holes*, um quarto arquivo continha valores para os coeficientes de clusterização e por ultimo um arquivo com os valores de densidade.

A única métrica que reflete a rede com um todo é a densidade da rede, as outras métricas são relativas aos nós da rede. Para estas métricas foram calculados os valores da soma, maior valor, média e desvio padrão de cada métrica. Esta abordagem também foi utilizada Biçer et al. (2011). A Figura 10 apresenta uma planilha que contém os valores das métricas de centralidade para os desenvolvedores e os valores calculados da soma, maior valor, média e desvio padrão de cada métrica.

Autor	Degree	Closeness	Betweeness	Eigenvector
gnodet	4.167	4.167	0	0
cziegeler	4.167	4.34	0	70.711
fmeschbe	8.333	4.348	0.362	100
dsavage	0	0	0	0
rickhall	4.167	4.167	0	0
ccustine	0	0	0	0
gertv	0	0	0	0
marrs	4.167	4.167	0	0
clement	0	0	0	0
mcculls	0	0	0	0
dbaum	0	0	0	0
ffang	0	0	0	0
pauls	0	0	0	0
srs	0	0	0	0
filippo	0	0	0	0
pderop	4.167	4.167	0	0
awojtuniak	0	0	0	0
walkerr	0	0	0	0
vvalchev	4.167	4.34	0	70.711
pkriens	0	0	0	0
jgoodyear	0	0	0	0

Degree			
Máximo	Soma	Média	Desvio Padrão
8.333	33.335	1.333	2.320

Closeness			
Máximo	Soma	Média	Desvio Padrão
4.348	29.696	1.188	1.945

Betweeness			
Máximo	Soma	Média	Desvio Padrão
0.362	0.362	0.014	0.072

Eigenvector			
Máximo	Soma	Média	Desvio Padrão
100.000	241.422	9.657	27.133

Figura 10 – Planilha exemplo com os valores de maior valor, soma, média e desvio padrão para as métricas de centralidade
Fonte: Autoria própria

Os valores 0 na planilha indicam que o valor para a métrica era zerado possivelmente porque o desenvolvedor não participou de interações com outros, ou porque o valor para métrica é equivalente a 0 mesmo.

Os valores da soma, maior valor, média e desvio padrão contidos nas planilhas do Excel e os valores da densidade das redes, foram utilizados para a geração do arquivo de entrada para o software WEKA.

3.3 PREDIÇÕES DO ESTADO DAS BUILDS

Para verificar o desempenho que métricas SNA aplicadas a redes sociais de comunicação e cooperação têm para predição do estado (sucesso,falha) de uma *build*,foi utilizada a suíte de algoritmos de aprendizagem WEKA. Esta suíte contém uma coleção de algoritmos para analise de dados e modelos de predição.

O WEKA utiliza um arquivo com extensão *.ARFF* como entrada dos algoritmos de aprendizagem. O ARFF é um arquivo de texto puro composto de três partes:

1. **Relação:** A primeira linha do arquivo deve ser iniciada com `@relation` seguida de uma palavra chave que identifica a relação ou tarefa sendo estudada.

2. **Atributos:** Um conjunto de linhas onde cada uma inicia com `@attribute` seguida do nome do atributo, e o tipo do valor, que pode ser:
    - Nominal: Neste caso os valores possíveis para o atributo devem aparecer com uma lista separada por vírgulas, entre um par de chaves.
    - Numérico: Neste caso deve ser apenas informado que o tipo do valor é `real`.

   Os valores obtidos para as métricas das redes sociais respectivas a um intervalo de tempo de uma *build* são todos numéricos. Um único atributo nominal existe para indicar o estado da *build* (sucesso ou falha).
3. **Dados:** Depois de uma linha contendo `@data`. Cada linha deve corresponder a uma instância de valores separados por vírgula estes valores devem estar na mesma ordem definida na seção de atributos.

A Figura 11 apresenta o conteúdo de um arquivo ARFF exemplo que foi utilizado, para compreender como os ARFF para este estudo deviam ser gerados.

```
1  @relation weather
2
3  @attribute outlook {sunny, overcast, rainy}
4  @attribute temperature real
5  @attribute humidity real
6  @attribute windy {TRUE, FALSE}
7  @attribute play {yes, no}
8
9  @data
10 sunny,85,85,FALSE,no
11 sunny,80,90,TRUE,no
12 overcast,83,86,FALSE,yes
13 rainy,70,96,FALSE,yes
14 rainy,68,80,FALSE,yes
15 rainy,65,70,TRUE,no
16 overcast,64,65,TRUE,yes
17 sunny,72,95,FALSE,no
18 sunny,69,70,FALSE,yes
19 rainy,75,80,FALSE,yes
20 sunny,75,70,TRUE,yes
21 overcast,72,90,TRUE,yes
22 overcast,81,75,FALSE,yes
23 rainy,71,91,TRUE,no
```

Figura 11 – Exemplo de um arquivo ARFF.
Fonte: Autoria Própria

Foram gerados três arquivos ARFF, que representam a comunicação, a cooperação, e a união da comunicação e cooperação. Os arquivos *ARFF* contêm os estados das 26 *builds*, e os respectivos valores das métricas das redes sócias de cada *build*.

Cada linha do ARFF refere-se a uma *build*. As linhas iniciam com o estado de uma *build* (sucesso, falha) e em seguida separado por vírgulas, os valores da soma, máximo, media e desvio padrão de cada métrica das redes sociais e o valor da densidade da rede.

A Figura 12 apresenta a declaração de alguns atributos, o primeiro atributo apresentando na linha 3 é o atributo status e seus possíveis valores *fail* – falha, *success* – succeso. Na linha 5 até a linha 8 são declarados os atributos máximo,soma,media e desvio padrão para métrica de centralidade *degree*, todos do tipo real.

```
1  @relation test
2
3  @attribute status {fail,success}
4
5  @attribute degreeMax real
6  @attribute degreeSum real
7  @attribute degreeMed real
8  @attribute degreeDesv real
9
10 @attribute betwenessMax real
11 @attribute betwenessSum real
12 @attribute betwenessMed real
13 @attribute betwenessDesv real
14
15 @attribute closenessMax real
16 @attribute closennesSum real
17 @attribute closennesMed real
18 @attribute closenessDesv real
19
20 @attribute eigenvectorMax real
21 @attribute eigenvectorSum real
22 @attribute eigenvectorMed real
23 @attribute eigenvectorDesv real
```

Figura 12 – Parcial do conteúdo de um arquivo ARFF utilizado neste estudo.
Fonte: Autoria Própria

Todas as métricas descritas na seção 2.3 tem a sua respectiva declaração nos arquivos ARFF gerados.

A Figura 13 apresenta a terceira seção de um arquivo ARFF utilizado neste estudo, que representa os valores para os atributos declarados na segunda seção do ARFF.

Cada linha da seção `@data` representa uma *build.* O primeiro dado é o estado da *build* seguido dos valores das métricas de redes sociais respectivas à *build*.

```
153 @data
154
155 fail,4.167,16.668,0.667,1.559,4.167,16.668,0.667,1.559,0.000,0.000,0.000,0.000,0.000,-2
156 fail,12.500,41.667,1.736,3.457,4.545,26.481,1.059,1.926,0.725,0.725,0.029,0.145,86.490,
157 success,20.833,50.001,2.000,4.520,5.252,36.410,1.456,2.384,5.072,6.884,0.275,1.063,97.3
158 success,25.000,100.002,4.000,5.828,5.543,67.496,2.700,2.661,6.522,11.595,0.464,1.390,90
159 fail,8.333,25.001,1.000,2.178,4.348,21.362,0.854,1.745,0.362,0.362,0.014,0.072,100.000,
160 success,12.500,41.669,1.667,2.946,4.545,34.797,1.392,2.074,1.087,1.087,0.043,0.217,100.
161 fail,8.333,33.335,1.333,2.320,4.348,29.696,1.188,1.945,0.362,0.362,0.014,0.072,100.000,
162 fail,4.167,8.334,0.333,1.154,4.167,8.334,0.333,1.154,0.000,0.000,0.000,0.000,0.000,-200
163 fail,20.833,108.334,4.333,7.165,5.839,52.036,2.081,2.833,3.321,11.594,0.464,1.055,71.71
164 fail,4.167,8.334,0.333,1.154,4.167,8.334,0.333,1.154,0.000,0.000,0.000,0.000,0.000,-200
165 success,16.667,50.003,2.000,3.633,4.762,40.366,1.615,2.204,2.174,2.174,0.087,0.435,100.
166 success,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,141.421
167 success,37.500,175.001,7.000,8.733,7.101,96.159,3.846,3.308,14.130,23.188,0.928,2.930,7
168 success,4.167,16.668,0.667,1.559,4.167,16.668,0.667,1.559,0.000,0.000,0.000,0.000,100.0
169 fail,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,141.421,14
170 fail,8.333,33.333,1.333,2.877,4.348,21.378,0.855,1.746,0.000,0.000,0.000,0.000,81.650,2
171 success,16.667,50.002,2.000,3.826,4.990,38.065,1.523,2.274,3.261,4.710,0.188,0.703,95.7
172 fail,4.167,16.668,0.667,1.559,4.167,16.668,0.667,1.559,0.000,0.000,0.000,0.000,0.000,-2
173 success,29.167,91.667,3.667,6.513,5.556,52.321,2.093,2.639,6.341,6.522,0.261,1.267,88.8
174 fail,4.167,16.668,0.667,1.559,4.167,16.668,0.667,1.559,0.000,0.000,0.000,0.000,0.000,-2
175 fail,8.333,16.667,0.667,1.969,4.348,13.028,0.521,1.440,0.362,0.362,0.014,0.072,100.000,
176 fail,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,141.421,14
177 success,37.500,175.001,7.000,8.733,7.101,96.159,3.846,3.308,14.130,23.188,0.928,2.930,7
178 success,4.167,16.668,0.667,1.559,4.167,16.668,0.667,1.559,0.000,0.000,0.000,0.000,0.000
179 success,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,141.421
180 success,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,141.421
```

Figura 13 – Parcial do conteúdo da seção @data de um arquivo ARFF utilizado neste estudo. Fonte: Autoria Própria

Após a geração dos arquivos de entrada para o software WEKA, foi executado o algoritmos de classificação: *Naive-Bayes*, *Bayes-Net* e *Bayesian-Logistic-Regression*. Os algoritmos necessitam de um conjunto de treinamento e um conjunto de validação.

O software WEKA é capaz de dividir o conjunto de dados de um ARFF em conjunto de treinamento e conjunto de validação, para isto deve-se definir a porcentagem do total de dados que deve ser utilizado para treinar o algoritmo, e o restante será utilizado como conjunto de validação.

Foram utilizadas varias porcentagens para a divisão dos conjuntos, como 50%, 66% e 80% do total de *builds* descritas no arquivo ARFF, para treinamento e o restante o WEKA utiliza como conjunto de validação.

4. RESULTADOS.

Para investigar as questões de pesquisa deste trabalho, foram realizadas avaliações para verificar o desempenho dos modelos de predição foi realizado. Para avaliar foram utilizadas as medidas de avaliação descritas na seção 2.5. Como todas as questões de pesquisa dizem respeito à predição de falhas, a avaliação foi desenvolvida analisando a classe falha.

Foram definidos modelos de predição para cada combinação entre os arquivos ARFF disponíveis, os algoritmos executados e as porcentagens de divisão dos dados das *builds* contidos nos arquivos ARFF.

Para melhor organizar este capitulo, os resultados serão apresentados separados baseando em cada questão de pesquisa deste estudo, e por fim uma conclusão sobre os resultados e as questões de pesquisas:

4.1 (Q1) MÉTRICAS DE SNA APLICADAS A REDES SOCIAIS EXTRAÍDAS DE MECANISMOS DE COMUNICAÇÃO DOS DESENVOLVEDORES PODEM AUXILIAR A PREDIÇÃO DE FALHA EM PROJETOS DE SOFTWARE?

Todos os modelos de predição relacionados a esta questão utilizaram o arquivo ARFF relativo aos dados das redes sociais de comunicação.

Modelo de predição 1, utilizou algoritmo *Bayes-Net* e porcentagem de 50% dos dados para treinamento e outros 50% para o conjunto de validação. A Tabela 2 apresenta a matriz de confusão para o modelo 1.

	falha	sucesso
falha	5	0
sucesso	8	0

Tabela 2 – Matriz de confusão para o modelo de predição 1
Fonte Autoria Própria

O valor da célula superior esquerda representa o valor de TP que pare este

caso é o numero de *builds* com estado falha, classificadas corretamente pelo algoritmo. A soma da primeira linha representa o numero de *builds* que realmente estão no estado falha. Neste caso o algoritmo conseguiu predizer todas as *builds* com estado falha, e obtendo um valor para Recall de 1 (100%).

Contudo isto não é suficiente para afirmar que o modelo de predição proposto tem um desempenho alto. *Precision* mede a capacidade de um modelo em rejeitar o que não se quer e neste caso o algoritmo classificou 8 *builds* com estado de sucesso como *builds* do estado de falha, como pode ser verificado observando a célula inferior esquerda. Isto indica um resultado de *Precision* de 0.385 (38,5%).

O valor obtido para medida de *F-Measure* foi de 0.556 que mostra que a classificação obteve desempenho baixo quando comparadas as medidas de *Recall* e *Precision*.

A Tabela 3 agrupa as diferentes combinações de algoritmos e divisão de dados para as redes sociais criadas a partir da comunicação dos desenvolvedores.

Modelos de Predição	1	2	3	4	5	6	7	8	9
Algoritmo	Bayes - Net			Naive Bayes			Bayesian Logistic Regression		
Porcetagem da divisão dos dados	50%	66%	80%	50%	66%	80%	50%	66%	80%
Recall para Falha	1	1	1	0.6	0	0.5	1	0.67	0.5
Precision para Falha	0.385	0.333	0.4	0.429	0	0.33	0.385	0.25	0.25
F-Measure para Falha	0.556	0.5	0.571	0.4	0	0.4	0.556	0.36	0.333

Tabela 3 – Resultados para os modelos de predição que utilizam dados de comunicação
Fonte: Autoria Própria

Observando a Tabela 3 verificar que todos os modelos que utilizam o algoritmo *Bayes-Net* conseguem classificar todas *builds* falhas corretamente baseado na analise do valor de *Recall* para falha o qual é 1 ou 100%, contudo nem um deles conseguiu rejeitar de forma satisfatória as *builds* com estado de sucesso, classificando-as como falha quando na realidade são *builds* com estado de sucesso.

Isto provoca uma diminuição dos valores de *Precision*, que acabam impactando na média harmônica calculada por *F-Measure* em valores baixos, não passando de 0.571, o que demonstra um desempenho baixo para os modelos de predição.

Para os modelos que utilizam os outros algoritmos os valores de *F-Measure* para estes são abaixo do valor de 0.571, implicando em um desempenho baixo nas combinações de *Bayes-Net* com 50% e *Bayesian Logistic Regression* com 50%.

O Gráfico 1 apresenta uma relação entre as *builds* que realmente falharam, com as *builds* preditas como falha, e *builds* preditas como falha que na realidade são sucesso, para o algoritmo *Bayes – Net*. Optou-se por apresentar a comparação deste algoritmo porque ele apresentou os melhores valores para *F-Measure*.

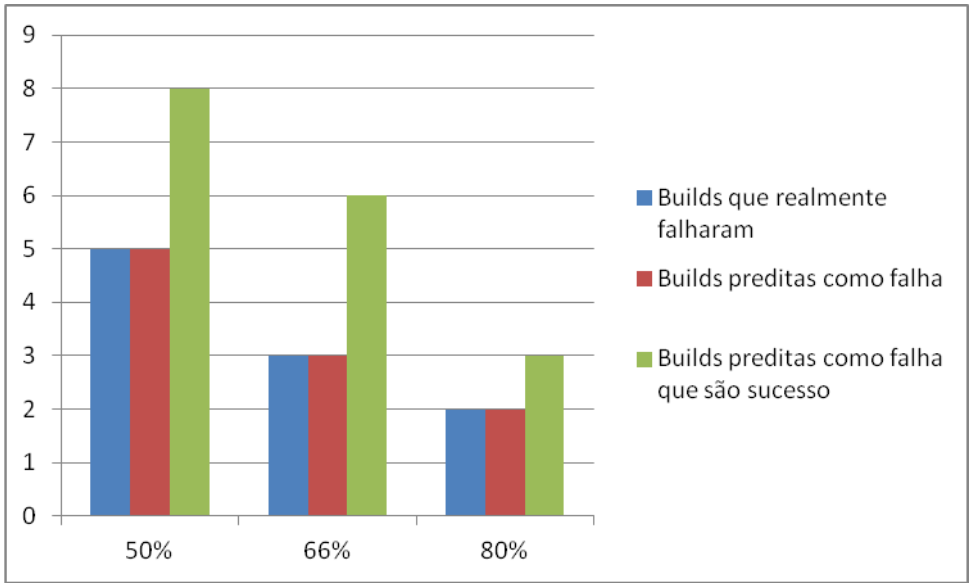


Gráfico 1 – Relação entre *builds* preditas para modelos que utilizam dados de comunicação.
Fonte: Autoria Própria

O Gráfico 1 indica que em qualquer divisão do conjunto de dados as *builds* que realmente falharam foram classificadas corretamente, no entanto, muitas *builds* que obtiveram sucesso são classificadas como *builds* com estado de falha.

Após a análise é possível responder a questão de pesquisa Q1, indicando que, para o projeto Felix GOGO métricas de SNA aplicadas a redes sociais de comunicação são modelos de predição com desempenho mediano quando utilizado o algoritmo *Bayes-Net* e configuração de 80% para treinamento-validação dos dados. Não se pode afirmar que a comunicação pode auxiliar na predição de falhas. Talvez o baixo desempenho das métricas de SNA aplicadas redes sociais de comunicação para predição seja devido aos possíveis ruídos que as mensagens trocadas no *mailing-list* possam conter, pois os participantes podem trocar mensagens dos mais variados assuntos, podendo conter mensagens que não são realmente efetivas ao desenvolvimento do projeto.

Retomando a idéia de Biçer et al.(2011), que afirmar que o conjunto de dados utilizado determina o impacto da predição, é necessário realizar novas combinações entre as métricas de SNA (adicionando e removendo métricas) em projetos distintos

do repositório ASF, incluindo e removendo outras fontes de dados de comunicação como os comentários em tarefas do *issue tracker* e dos comentários existentes nos *commits*, para sugerir o impacto da comunicação da predição de falhas.

4.2 (Q2) MÉTRICAS DE SNA APLICADAS A REDES SOCIAIS EXTRAÍDAS DE MECANISMOS DE COOPERAÇÃO DOS DESENVOLVEDORES PODEM AUXILIAR A PREDIÇÃO DE FALHA EM PROJETOS DE SOFTWARE?

Para responder esta questão de pesquisa foi utilizado o arquivo ARFF referente, aos dados das redes sociais de cooperação, como entrada para os modelos de predição, e os mesmo algoritmos e porcentagens de divisão que foram utilizados para responder a questão de pesquisa Q1. Também foram definidos modelos de predição para cada combinação entre algoritmos e as porcentagens de divisão.

A Tabela 4 apresenta a matriz de confusão para o modelo de predição 14, que utilizou o algoritmo *Naive Bayes* com porcentagem de divisão 66% para treinamento e o restante dos dados para validação.

	falha	sucesso
falha	5	0
sucesso	2	2

Tabela 4 – Matriz de confusão para o modelo de predição 14.
Fonte: Autoria Própria

Observando a Tabela 4 baseando nas definições apresentadas na seção 2.5, verifica-se que este modelo conseguiu predizer corretamente todas as *builds* que realmente estão no estado de falha, obtendo um *Recall* de 1 ou 100%.Para este caso, o valor de *Precision* obtido foi de 5/ 5 +2 = 0.714 ou 71%. Este valor de *Precision* significa que modelo obteve algum êxito em rejeitar *builds* com estado de sucesso.

Calculando a medida *F-Measure* descrita na seção 2.5, é obtido o valor 0.833, que indica um desempenho satisfatório para o modelo de predição 14. Este modelo classificou todas as *builds* falhas corretamente e conseguiu apresentar uma precisão de 71%.

A Tabela 5 apresenta os modelos de predição definidos para responder está questão.

Modelos de Predição	10	11	12	13	14	15	16	17	18
Algoritmo	Bayes - Net			Naive Bayes			Bayesian Logistic Regression		
Porcetagem da divisão dos dado	50%	66%	80%	50%	66%	80%	50%	66%	80%
Recall para Falha	0	0	1	0.714	1	1	0.857	0.8	0.667
Precision para Falha	0	0	0.75	0.625	0.714	0.75	0.545	0.67	1
F-Measure para Falha	0	0	0.857	0.667	0.833	0.857	0.667	0.73	0.8

Tabela 5 - Resultados para os modelos de predição que utilizam dados de cooperação
Fonte: Autoria Própria

Observa-se que em geral todos os modelos de predição tem valores satisfatórios para *Recall*, *Precision* e *F-Measure* excluindo os modelos 10 e 11, que obtiveram 0 para as 3 medidas de avaliação.

Todos os modelos de predição apresentados na Tabela 5 exceto o 10 e 11, tem valores para Recall igual ou acima 0.667 (67%), valores para *Precision* igual ou acima de 0.545 (55%) e valores para *F-Measure* igual ou acima de 0.667 (67%). Para estão questão de pesquisa os valores para *Precision* são melhores em comparação com os valores encontrados para os modelos de predição da questão de pesquisa Q1.

Os modelos de predição que obtiveram o melhor valor para *F-Measure* foram os modelos 12 e 15 alcançando 0.857 para *F-Measure*, os quais neste caso são os melhores modelos de predição..

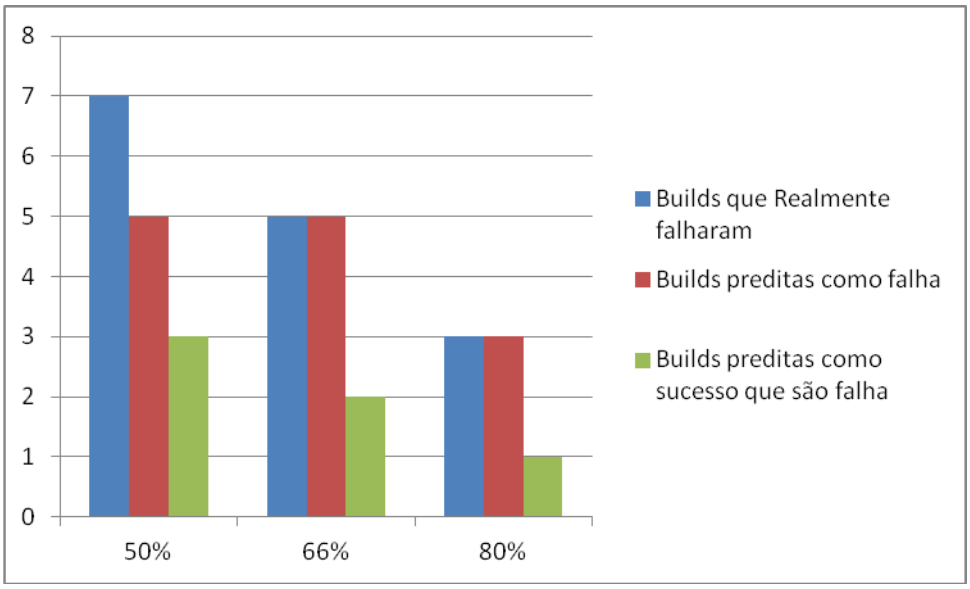


Gráfico 2 – Relação entre *builds* preditas para modelos que utilizam dados de cooperação
Fonte: Autoria Própria

O Gráfico 2 apresenta uma comparação entre a quantidade de *builds* que realmente falharam, *builds* que foram preditas como falha e *builds* que obtiveram sucesso que foram classificadas erroneamente como *builds* que falharam, para os modelos que utilizaram o algoritmo *Naive Bayes*

Após a analise é possível responder a questão de pesquisa Q2, indicando que métricas de SNA aplicadas a redes sociais de cooperação podem auxiliar na predição de falhas, pois possibilitam a geração de modelos de predição com alto desempenho.

Possivelmente o melhor desempenho em relação à comunicação esteja relacionado com a possibilidade de haver ruídos nas mensagens.

4.3 (Q3) RESULTADOS COMBINADOS A PARTIR DA ANALISE DAS REDES SOCIAIS CRIADAS PODEM MELHORAR OS RESULTADOS DE PREDIÇÃO?

Para responder esta questão de pesquisa assim como as questões Q1 e Q2, foram definidos modelos de predição que neste caso foi utilizado o arquivo ARFF que contém dados das métricas de SNA relativas às redes sociais de comunicação e cooperação. Os modelos definidos utilizam os mesmo algoritmos e as mesmas porcentagens de divisão dos conjuntos de treinamento e validação utilizados nas questões Q1 e Q2.

A Tabela 6 apresenta a matriz de confusão para modelo de predição 27 que o utilizou o algoritmo *Bayesian – Logistic – Regression* utilizando 66% dos dados para treinamento e o restante para validação.

	falha	sucesso
falha	3	0
sucesso	2	4

Tabela 6 – Matriz de confusão para o modelo de predição 27.
Fonte: Autoria Própria

Analisando a Tabela 6 e baseando – se nas definições descritas na seção 2.5 identifica-se que o modelo de predição 27 conseguiu predizer corretamente todas as *builds* falhas e classificou incorretamente duas *builds* com estado de sucesso como *builds* falhas. Assim alcançando um *Recall* de 1 (100%), e *Precision* de 3 / 3+2 = 0.6 (60%), este valor de *Precision* indica que o modelo obteve algum êxito em rejeitar

builds com estado de sucesso, que para este caso foram 4 *builds* sucesso que ele rejeitou e classificou corretamente como sucesso, e classificando incorretamente duas *builds* sucesso como falha. O valor de *F-measure* para teste caso é de 0.75 o que indica que modelo 27 tem um desempenho bom para predição segundo Delgado et, al (2010), pois está próximo de 1.

A Tabela 7 apresenta todos os modelos definidos e os respectivos valores de *Recall*, *Precision* e *F-Measure* para responder a questão de pesquisa Q3.

Modelos de Predição	19	20	21	22	23	24	25	27	28
Algoritmo	Bayes - Net			Naive Bayes			Bayesian Logistic Regression		
Porcetagem da divisão dos dado	50%	66%	80%	50%	66%	80%	50%	66%	80%
Recall para Falha	1	1	1	0.8	1	1	0.6	1	1
Precision para Falha	0.385	0.429	0.4	0.667	0.429	0.667	0.429	0.6	0.667
F-Measure para Falha	0.556	0.6	0.571	0.727	0.6	0.8	0.5	0.75	0.8

Tabela 7 - Resultados para os modelos de predição que utilizam dados de comunicação e cooperação
Fonte: Autoria Própria

Observando a Tabela 7 é possível verificar que os modelos que utilizam o algoritmo *Bayes – Net* não conseguem atingir um bom desempenho para predizer falhas, pois os valores de *F-Measure* para estes modelos estão igual ou abaixo de 0.6. Já para os modelos que utilizam o algoritmo Naive Bayes todos obtiveram um valor de *F-Measure* maior ou igual a 0.6, o que implica em modelos bons para predição. E por ultimo os modelos que utilizam o algoritmo Bayesian – Logistic – Regression exceto utilizando 50% para divisão dos dados, obtiveram valores iguais ou acima de 0.75, implicando em modelos de predição com desempenho alto para predição de falhas.

Em geral todos os modelos tem valores para *F-Measure* entre 0.5 e 0.8 indicando que para maioria dos casos os valores de *F-Measure* estão mais próximos de 1 do que 0, o que segundo Delgado et al. (2010) implica em modelos com desempenho bom para predição.

O Gráfico 3 apresenta uma comparação entre a quantidade de *builds* que realmente falharam, *builds* que foram preditas como falha e *builds* que obtiveram sucesso que foram classificadas erroneamente como *builds* que falharam, para os modelos que utilizaram o algoritmo *Bayesian – Logistic – Regression*.

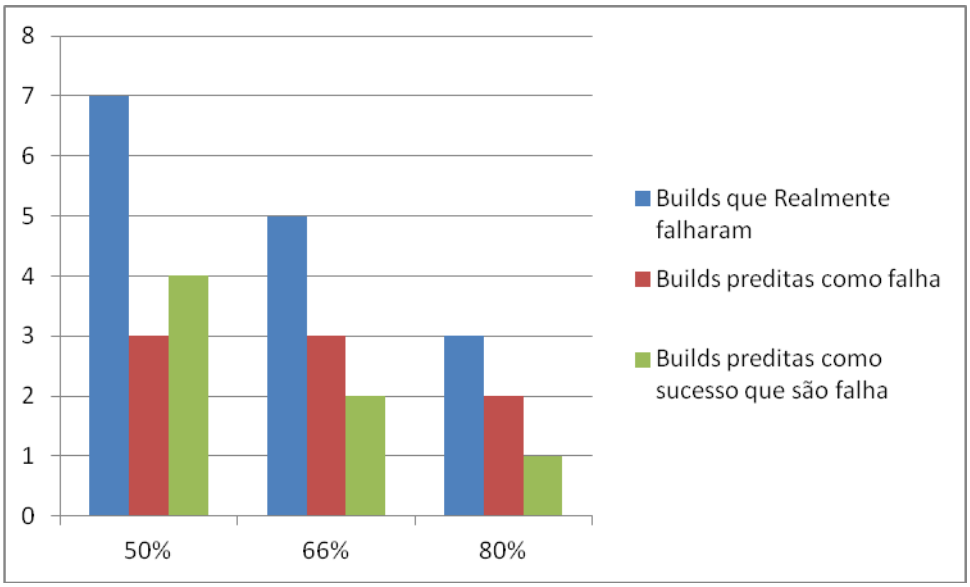


Gráfico 3 – Relação entre *builds* preditas para modelos que utilizam dados de comunicação e cooperação
Fonte: Autoria Própria

Após a analise podemos concluir que a resposta para questão de pesquisa Q3 é que métricas de SNA aplicadas a rede sociais de comunicação e cooperação podem auxiliar na predição de falhas do projeto Felix GOGO.

5. DISCUSSÃO

Vale ressaltar que os valores de *F-Measure* para os modelos definidos para Q3 em geral são altos, porém menores do que os valores de *F-Measure* para os modelos definidos pra questão Q2, como pode ser verificado na Tabela 8.

Algoritmo	Bayes - Net			Naive Bayes			Bayesian Logistic Regression		
Porcetagem da divisão dos dado	50%	66%	80%	50%	66%	80%	50%	66%	80%
F-Measure dos modelos para Q3	0.556	0.6	0.571	0.73	0.6	0.8	0.5	0.75	0.8
F-Measure dos modelos para Q2	0	0	0.857	0.67	0.83	0.857	0.667	0.73	0.8

Tabela 8 – Comparação de F-Measure dos modelos de predição para questões de pesquisa Q2 e Q3
Fonte: Autoria Própria

Na Tabela 8 estão destacados os valores de F-Measure iguais ou maiores que 0.8, podemos verificar que a maior quantidade de valores iguais ou maiores que 0.8 são para modelos de predição definidos para a questão de pesquisa Q2. Para questão Q3 apenas dois valores são iguais a 0.8 e, para questão Q3, 3 valores são maior que 0.8 e um valor é igual a 0.8. A partir da análise da Tabela 8 é possível

concluir que métricas de SNA aplicadas a redes sociais de cooperação, obtêm um melhor desempenho na predição de falhas do que métricas de SNA aplicadas a redes sociais de comunicação e cooperação.

Possivelmente esta diferença de desempenho deve-se à união de comunicação com cooperação visto que as métricas de SNA aplicadas a rede sociais de comunicação não são uteis para predizer falhas como descrito anteriormente, e unindo redes de comunicação e cooperação, talvez as redes de comunicação impactam na redução do desempenho das redes cooperação para predição de falhas.

Como citado anteriormente as métricas de SNA aplicadas a redes de comunicação obtiveram um baixo desempenho para predição de falhas, pois podem existir ruídos nas mensagens que são utilizadas para gerar as redes sociais de comunicação, por esta razão as redes sociais de cooperação e comunicação tiveram o desempenho reduzindo em comparação a utilização de redes sociais apenas de cooperação.

6. AMEAÇAS DE VALIDAÇÃO

Este capítulo irá apresentar as ameaças de validação identificadas para este estudo.

Após a execução dos modelos de predição foi identificado que o conjunto de dados extraídos para a geração das redes sociais é relativamente pequeno e talvez se a quantidade de dados for maior, os resultados possam ser diferentes. Em conseqüência as respostas para as questões de pesquisas podem ser outras.

Esta ameaça deve-se à quantidade reduzida de *builds* armazenadas pelo Projeto Felix GOGO na ferramenta de integração Hudson, foram pesquisados outros projetos que armazenassem uma quantidade maior de *builds*, contudo o Projeto Felix GOGO é o projeto que armazena o maior histórico das *builds*.

Outra ameaça identificada é que a metodologia foi desenvolvida apenas para o Projeto Felix e não foi testada em outro projeto pelo fato de outros projetos armazenarem um histórico de *builds* muito pequeno.

Em relação aos dados foi identificada outra ameaça, pois neste estudo somente foram levados em consideração *mailing-list* como mecanismo de

comunicação dos desenvolvedores, sendo que possivelmente os desenvolvedores utilizam outros mecanismos como *issue-trackers*, *bug-trackers*, comentários dos *commits* entre outros.

Ainda em relação aos dados outra ameaça identificada foi a possível existência de ruídos nas mensagens extraídas das *mailing-list,* em outras palavras talvez nem todas as mensagens trocadas sejam efetivamente referentes ao processo de desenvolvimento do projeto Felix – GOGO, estas podem poluir os dados, o que pode resultar em baixos desempenhos para modelos de predição quando utilizados como conjunto de entrada.

A ultima ameaça identificada é em relação a seleção das métricas de SNA utilizadas, pois estas foram selecionadas analisando os trabalhos relacionados e não foi verificado o impacto de cada métrica em relação a predição de falhas.

7. TRABALHOS FUTUROS

Para trabalhos futuros deseja-se ampliar o conjunto de dados e adicionar dados relativos a todos os meios de comunicação que possam ser utilizados pelos desenvolvedores, submeter os dados de comunicação a um filtro que tente eliminar as mensagens consideradas como ruído, assim resultando em um conjunto de dados livre de poluição contendo dados de vários mecanismos de comunicação.

Também é desejável realizar o experimento para mais projetos software e realizar uma comparação entre os resultados para cada projeto para verificar se a metodologia pode ser validada para outros projetos de software, bem como testar combinações de métricas de SNA diferentes.

REFERÊNCIAS

API Java Svn. Disponível em:

http://svnkit.com/documentation.html Acessado em Agosto de 2011.

ANACLETO, Ana Cristina da Silva; **Aplicação de Técnicas de Data Mining em Extracção de Elementos de Documentos Comerciais,** 2009 - Faculdade de Economia - Universidade do Porto, Porto Portugal, 2009.

BIÇER, Serdar; BENER, Ayse; CAGLAYAN, Bora. **Defect Prediction Using Social Network Analysis on Issue Repositories.** In ICSSP '11 Proceedings of the 2011 International Conference on on Software and Systems Process. New York, NY, USA, 2011.

BIRD, Christian; PATTISON, David; D'SOUZA, Raissa; FILKOV, Vladimir; DEVANBU, Premkumar. **Latent Social Structures in Open Source Projects**. In SE, Atlanta, GA, 2008, p.p24-36.

BORGATTI, S.P., EVERETT, M.G. and FREEMAN, L.C. 2002. **Ucinet for Windows: Software for Social Network Analysis**. Harvard, MA: Analytic Technologie.

DELGADO, Carlos Henrique; VIANNA, Caroline Evangelista; GUELPELI, Marcus Vinicius C. **Comparando Sumários De Referência Humanos Com Extratos Ideais No Processo De Avaliação De Sumários Extrativos**. In: IADIS Ibero-Americana WWW/Internet 2010, p. 293 - 300 Algarve, Portugal, 2010.

EVERITT B.S. (2002) **Cambridge Dictionary of Statistics**, CUP. ISBN 0-521-81099-x

FOUTOURA, V. ; STEINMACHER, I. ; WIESE, I. S. ; GEROSA, M. A. . **Predição de falhas em projetos de software baseado em métricas de redes sociais de comunicação e cooperação**. In: SBSC e CRWIG, 2011, Parati - RJ. Anais do VIII Simpósio Brasileiro de Sistemas Colaborativos, 2011. p. 37-40.

FUKS, Hugo; RAPOSO, Alberto; GEROSA, Marco. **Do Modelo de Colaboração 3C à Engenharia de Groupware.** In WEBMIDIA 2003 -Simpósio Brasileiro de Sistemas Multimídia e Web, Trilha especial de Trabalho Cooperativo Assistido por Computador, Novembro 2003, Salvador-BA, pp. 445-452.

GENKIN, Alexander; MADIGAN, David; LEWIS David D. **Large-Scale Bayesian Logistic Regression for Text Categorization**. Vol. 49, No. 3, pp. 291-304. (August 2007).

HAN, Jiawei; KAMBER, Micheline. **Data Mining: Concepts and Techniques** Morgan Kauffman, Second Edition. Illinois 2006.

HANNEMAN, Robert A; RIDDLE, Mark. 2005**. Introduction to social network methods**. Riverside, CA: University of California, Riverside (published in digital form at http://faculty.ucr.edu/~hanneman/)

HECKERMAN, David**. A Tutorial on Learning With Bayesian Networks. Microsoft Research,** Page 57. number MSR-TR-95-06, March 1995.

MAGDALENO, Andréa; WERNER, Cláudia; Araujo, Renata. **Estudo de Ferramentas de Mineração, Visualização e Análise de Redes Sociais**. Rio de Janeiro, ES-735/10, 2010.

MARTINS, António Cardoso; COSTA, Paulo Dias. **Estudo de três algoritmos de machine learning na classificação de dados eletrocardiográficos**. 2009, Tese de mestrado – Faculdade de Medicina da Universidade do Porto, Porto Portugal, 2009.

MENEELY, Andrew; WILLIAMS, Laurie; SNIPES, Will; Osborne, Jason. **Predicting failures with developer networks and social network analysis.** In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering (SIGSOFT '08/FSE-16). ACM, New York, NY, USA, 2008.

MENEELY, Andrew; WILLIAMS, Laurie. **Socio-technical developer networks: should we trust our measurements?**. In Proceeding of the 33rd international conference on Software engineering (ICSE'11). ACM, New York, NY, USA, 281-290. DOI=10.1145/1985793.1985832, 2011.

MENEELY, Andrew; WILLIAMS, Laurie; CORCORAN, Mackenzie. **Improving Developer Activity Metrics with Issue Tracking Annotations**. In Workshop on Emerging Trends in Software Metrics (WETSoM), Cape Town, South Africa, 2010.

MENZIES, Tim; GREENWALD, Jeremy; FRANK, Art. **Data Mining Static Code Attributes to Learn Defect Predictors**. In Ieee Transactions On Software Engineering, West Virginia Univ., Morgantown, WV, Jan. 2007.

NIA, Roozbeh; BIRD, Christian; DEVANBU, Premkumar; FILKOV, Vladimir. **Validity of Network Analyses in Open Source Projects.** In Mining Software Repositories, Cape Town, South Africa, 2010.

SARR M. **Improving precision and recall using a spellchecker in a search engine**. Royal Institute of Technology, Sweden, 2003.

SHIN, Yonghee; MENEELY, Andrew; WILLIAMS, Laurie; OSBORNE Jason. **Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities.** In IEEE Transactions in Software Engineering (TSE), vol. to appear, no. p. 2010.

STEINMACHER, Igor; CHAVES, Ana; GEROSA, Marco. **Awareness Support in Global Software Development: A Systematic Review Based on the 3C Collaboration Model.** In: 16th CRIWG Conference on Collaboration and Technology, 2010, Maastricht. Lecture Notes in Computer Science, 2010. v. 6257. p. 185-201

WASSERMAN, Stanley; FAUST, Katherine; **Social Network Analysis: Methods and Applications**. 1 ed. Cambridge, United Kingdom, Cambridge University Press. 1994.

Weka Software is a collection of machine learning algorithms for solving real-world data mining problems. Disponível em **http://sourceforge.net/projects/weka/**

WOLF, Timo; SCHROTER, Adrian; DAMIAN, Daniela; NGUYEN, Thanh. **Predicting build failures using social network analysis on developer communication.** In Proceedings of the 2009 IEEE 31st International Conference on Software Engineering, 2009.

ZIMMERMANN, Thomas; NAGAPPAN, Nachiappan; **Predicting defects using network analysis on dependency graphs,**. In Proceedings of the 30th international conference on Software engineering, Leipzig, Germany , May 10-18, 2008.