# Uma abordagem para classificação de construção de software utilizando redes de comunicação e cooperação.

**Vidal D. da Fontoura[1], Igor Wiese[1], Igor Steinmacher [1], Reginaldo Ré[1] , José Teodoro da Silva e Marco Aurélio Gerosa[2]**

[1]Coordenação de Informática – Universidade Tecnológica Federal do Paraná (UTFPR)
Campo Mourão – PR – Brasil

[2]Departamento de Ciência da Computação – Universidade de São Paulo (USP)
São Paulo – SP – Brasil.

vidalfontoura@hotmail.com,{igorfs,igor,reginaldo.re}@utfpr.edu.br,{jteodoro,gerosa}@ime.usp.br

***Abstract.*** *Communication and Cooperation have been subject of several studies in Software Engineering. Social Network Analysis (SNA) metrics have been used to analyze networks built from the interaction of developers during the software development process. This work aims to predict software build failures, using classification models with three different Bayesian algorithms. The models use the SNA metric values calculated for the communication and cooperation networks, besides the sum of the interactions between them. The results indicate that communication networks were not suitable to classify builds which have failed. Nevertheless, cooperation networks obtained f-measure rates higher than 0.8, indicating more reliable results. In some cases the sum of the interactions increased the f-measure, recall and precision rates. These results show that notification systems could be built based on this kind of classification to alert software engineers on probable software build failures based on the developer's way of interaction.*

***Resumo.*** *A comunicação e a cooperação têm sido objeto de vários estudos em engenharia de software. Métricas de análise de redes sociais (Social Network Analysis - SNA) têm sido utilizadas para analisar redes construídas a partir da interação dos desenvolvedores durante o desenvolvimento do software. Este trabalho tem o objetivo predizer falhas em construções do software, utilizando modelos de classificação com três diferentes algoritmos bayesianos. Os modelos utilizam os valores das métricas SNA calculadas para as redes de comunicação, cooperação e a soma das interações entre as duas redes. Os resultados indicam que redes de comunicação não foram boas para classificar construções que falharam. No entanto, redes de cooperação obtiveram valores de f-measure superiores a 0.8, o que indicam resultados mais confiáveis. Em alguns casos a soma das interações das duas redes melhorou o valor de f-measure, recall e precision. Estes resultados indicam que sistemas de notificação poderiam ser construídos com a base neste tipo de classificação para alertar engenheiros de software sobre prováveis falhas nas construções do software baseada na forma de interação entre os desenvolvedores.*

## 1. Introdução

Uma grande quantidade de dados é produzida durante o desenvolvimento de software. Desenvolvedores utilizam diversas ferramentas para coordenar suas tarefas, trabalhar cooperativamente em artefatos e comunicar informações sobre o projeto. Controladores de versão (SVN, GIT, entre outros), gerenciadores de *issues* e defeitos (Jira, Bugzilla) e listas de e-mails (Google Groups, Yahoo Grupos), são exemplos de ferramentas que contém fontes de dados que podem ser minerados e interpretados em busca de informações de coordenação, cooperação e comunicação entre desenvolvedores. [Hassan 2008]

A comunicação e a cooperação têm sido estudadas de diversas formas a partir de dados obtidos por estas ferramentas. Dentre estas formas, destacam-se aquelas que utilizam redes sociais para capturar a interação dos desenvolvedores, construindo redes de comunicação ou cooperação [Meneely and Williams, 2011], [Bird et. al, 2009], [Nia et. al, 2010], [Shin et. al, 2010], [Wolf et. al, 2009], [Meneely et. al, 2011] e [Biçer et. al, 2011].

Redes sociais são analisadas com diversas métricas, conhecidas como SNA (*Social Network Analysis*). Com estas métricas é possível descrever, caracterizar e explicar uma rede por meio da exploração de suas propriedades, tais como, centralidades, grau, densidade, entre outros [Magdaleno et. al, 2010].

Para este trabalho foram coletados dados de comunicação (listas de e-mail) e cooperação (obtidos a partir do *log* dos controladores de versão), hospedados no repositório da *apache foundation,* com o objetivo de predizer falhas em construções do software (do inglês, *builds*) a partir de redes de comunicação e cooperação. Para a classificação são comparados três algoritmos bayesianos que permitem fazer a classificação de construções que estejam no estado de "falha".

Os modelos de classificação utilizam como parâmetros de entrada métricas SNA calculadas a partir das redes de comunicação ou cooperação e possibilitam a identificação de uma classe. Para guiar a metodologia deste trabalho três questões de pesquisa foram investigadas: Q1: Redes de comunicação podem ser utilizadas para classificar construções propensas a falhar? Q2: Redes de cooperação podem ser utilizadas para classificar construções propensas a falhar? Q3: a união das redes permite melhorar os resultados de classificação obtidos individualmente em Q1 e Q2 para classificação de construções propensas a falhar?

Identificar o estado da construção de um software é importante, pois ferramentas de notificação e monitoramento de projetos podem ser desenvolvidas para auxiliar gerentes, lideres de projetos e desenvolvedores a diminuir os esforços da fase de manutenção.

O presente artigo esta organizado da seguinte forma: A seção 2 apresenta os trabalhos relacionados. A seção 3 apresenta como o modelo de classificação é avaliado, a metodologia de coleta de dados e geração das redes de comunicação e cooperação é apresentada na seção 4. A seção 5 apresenta os resultados para cada uma das questões de pesquisa. Por fim, as conclusões são apresentadas na seção 6.

## 2.Trabalhos relacionados

Wolf et. al, 2009 coletaram dados do IBM JAZZ e analisaram as redes sociais formadas a partir dos desenvolvedores que trabalharam em uma mesma tarefa (do inglês, *work item*) e que se comunicaram para a realização desta tarefa. Eles utilizaram algoritmos bayesianos para classificar as construções propensas a falhar, calculando métricas SNA a partir das redes de cooperação e comunicação.

Meneely et. al, 2008 analisaram o histórico das diferentes versões dos arquivos de um projeto verificando os responsáveis pelas atualizações de cada arquivo. Todos os desenvolvedores que alteraram um arquivo foram conectados para formar a rede de contribuição do arquivo. Os autores calcularam métricas SNA sobre esta rede, para prever quais arquivos eram mais propensos a falhar.

Biçer et. al, 2011 criaram redes de comunicação baseadas na troca de comentários feitos por desenvolvedores em gerenciadores de tarefas. Os autores também utilizaram métricas SNA para predizer arquivos propensos a falhar.

Schröter et. al, 2006 construíram redes de dependência entre os arquivos, agrupando os arquivos de forma que representassem um componente do software. Diferentemente dos trabalhos anteriores, a classificação realizada foi sobre os componentes propensos a falhar. Eles utilizaram dados de um software proprietário e também utilizaram métricas SNA como entrada para o modelo de classificação.

O presente trabalho utiliza a abordagem proposta por [Wolf et. al 2009] para criar modelos de classificação para construções propensas a falhar. São utilizados dados de comunicação e cooperação entre os desenvolvedores. No entanto, ao invés de utilizar os comentários dos desenvolvedores em uma tarefa, serão utilizados os e-mails trocados entre os desenvolvedores. Dentre as diferentes métricas SNA utilizadas pelos trabalhos mencionados, escolheu-se usar as métricas de densidade da rede, coeficiente de clusterização, métricas de centralidade, métricas de *ego-network* e métricas de *structural holes*.

Estas métricas foram escolhidas por representar diferentes propriedades das redes e também utilizarmos um conjunto de métricas maior que os trabalhos relacionados.

## 3. Avaliação Dos Modelos De Classificação

Para [Biçer et. al 2011] a melhoria do desempenho dos modelos de classificação não esta associada à implementação de novos algoritmos, mas sim, à melhoria da qualidade do conjunto de dados coletados ou, até mesmo, de métricas que ainda não foram utilizadas para classificação.

Assim como [Biçer et. al 2011] e [Wolf et. al 2009], o modelo proposto neste trabalho utiliza algoritmos de classificação bayesiana, no entanto, utilizamos um maior conjunto de métricas e diferentes combinações de redes em comparação aos trabalhos relacionados. Para execução dos testes de classificação foi usada a ferramenta WEKA, que implementa vários algoritmos de aprendizagem de máquina.

A avaliação de um modelo de classificação é feita dividindo a amostra de dados em dois conjuntos: um de treinamento e outro de validação. O modelo aprende com o conjunto de treinamento e utiliza o conjunto de validação para verificar se o aprendizado é capaz de classificar estas instancias. Esta técnica serve para avaliar o modelo e estimar a incerteza das suas classificações [Han and Kamber 2006]. Existem várias medidas para avaliar as classificações realizadas. Neste trabalho foram utilizadas as medidas de *Recall*, *Precision* e *F-Measure*.

*Recall* representa o número de instâncias corretamente classificadas sobre todas as instâncias que representam construções que falharam. *Precision* representa o número de instâncias corretamente classificadas como construções que falharam, dividido pelo total do número de instâncias que estão na classe de falha. *Recall* indica a sensibilidade do modelo, enquanto *Precision* indica a sua corretude. *F-Measure* é calculada com a média harmônica entre estas duas métricas e indica a relação entre a sensibilidade de detectar uma classe e a corretude da sua classificação. [Menzies et. al, 2007]

Diferentes abordagens de avaliação podem ser adotadas. Optamos por dividir os conjuntos de treino e validação aleatoriamente, utilizando o método *hold-out.* O método

realiza a separação aleatória de acordo com um valor configurado, por exemplo, dividir 50% dos dados para teste e 50% dos dados para treinamento. A construção do modelo foi executada com três algoritmos *Naive bayes* [Martins and Costa 2009], *Bayes-Net* [Heckerman 1995], *Bayesian Logistic Regression* [Genkin et. al 2007].

## 4. Metodologia

Para responder as questões de pesquisa os seguintes passos foram seguidos: (i) coleta de dados e criação das redes sociais; e (ii) análise das redes sociais.

### 4.1 Coleta de dados e criação das redes sociais

A coleta de dados foi realizada no projeto Felix GOGO hospedado na *apache software foundation*. A ferramenta de integração contínua Hudson foi utilizada pelo projeto para construir o código fonte compilando-o a partir da última versão do código fonte encontrado no repositório de código. Quando ocorre algum erro durante a compilação e execução dos testes ou do código-fonte, a ferramenta indica que a construção falhou.

Foi construída para este trabalho uma ferramenta para coletar os dados disponíveis do Hudson. O *crawler* recebe o endereço da página do projeto e coleta os períodos de construção do projeto. São identificados para cada construção, a data e o estado da construção (falha ou sucesso).

Foram coletados todos os dados de comunicação e cooperação entre as datas de 25-07-2009 e de 19-09-2011. Vinte e seis construções foram coletadas neste intervalo, com duração de aproximadamente um mês. Treze construções encontravam-se no estado de sucesso e treze no estado de falha.

Para reconstrução das interações foram coletadas informações de comunicação das listas de e-mail e do controle de versão do código fonte. Da comunicação foram coletados o autor da mensagem, o assunto, a data e o conteúdo da mensagem. Do repositório de código fonte foram coletados dados dos autores dos *commits*, dos arquivos alterados e das mensagens.

Para a construção das redes de comunicação e rede de cooperação, foram identificados os intervalos de data existentes entre duas construções. A rede de cooperação foi construída a partir do relacionamento entre os desenvolvedores que realizaram *commit* de um mesmo arquivo neste intervalo. Para a rede de comunicação, dois desenvolvedores eram relacionados quando trocavam mensagens neste mesmo intervalo de data.

As redes criadas não são direcionadas, mas contém nas arestas valores da interação entre estes desenvolvedores. Para a rede de comunicação, foi representada a quantidade de mensagens trocadas pelos desenvolvedores. Para a rede de cooperação, as arestas receberam a quantidade de vezes que dois desenvolvedores fizeram *commit* de um mesmo arquivo.

Por fim, para responder a terceira questão de pesquisa, foi realizada a soma entre os valores contidos nas arestas das matrizes de comunicação e colaboração, quando dois desenvolvedores tinham as duas interações. Por exemplo, quando o desenvolvedor *A* trocou e-mails com o desenvolvedor *B* e também realizou *commits* de arquivos que também foram alterados por *B*. Caso só exista a relação entre dois desenvolvedores em uma das matrizes, este valor é repetido na matriz da soma.

### 4.2 Análises Das Redes Sociais

Para analisar as redes de comunicação e cooperação, utilizamos a ferramenta UCINET. Ela oferece recursos de visualização e cálculo das métricas SNA utilizadas por este trabalho. Para cada uma das redes sociais (matrizes) - 26 redes de comunicação, 26 redes de cooperação, e 26 redes de comunicação e cooperação - realizamos a importação destas redes para a ferramenta.

Algumas métricas retornam valores para um nó, no entanto, a rede representa uma construção do software e os valores destes nós devem ser convertidos para esta construção. Quando as métricas possuíam esta característica utilizamos a abordagem descrita por [Biçer et. al 2011] que converte os valores das métricas individuais de todos os nós para um único valor. Para fazer esta transformação, obtêm-se do o maior valor da métrica de um nó, a média dos valores de todos os nós, o desvio padrão desta média e a soma dos valores de todos os nós.

Nós computamos as métricas para 78 matrizes, que representam os três tipos de redes construídas para os 26 construções do software coletada. Estes valores representam as entradas para o modelo de classificação, e foram exportados no formato CSV e posteriormente convertidos para o formato ARFF da ferramenta WEKA.

Com a ferramenta WEKA, executamos os algoritmos de classificação: Naive Bayes, Bayes-Net e Bayesian Logistic Regression. Os algoritmos necessitam de um conjunto de treinamento e um conjunto de validação. Para isto deve-se definir o percentual total de dados que deve ser utilizado para treinar o algoritmo. A validação foi realizada com as amostras separadas nas seguintes proporções de treino e validação: 50%/50%, 66%/34% e 80%/20%.

## 4. Resultados

Para investigar as questões de pesquisa deste trabalho, foram realizadas avaliações para verificar o desempenho dos modelos de classificação. Foram reportados os valores de *recall*, *precision* e *f-measure* obtidos para cada uma das redes. Como todas as questões de pesquisa dizem respeito à classificação de construções propensas a falhar, a avaliação foi desenvolvida analisando somente a classificação para as construções que falharam.

Um modelo de classificação é a combinação de uma matriz (comunicação, cooperação, comunicação-cooperação), um algoritmo (Bayes-Net, Naive Bayes, Bayesian Logistic Regression) e uma configuração de treinamento (50%/50%, 66/34%, 80%/20%).

Para Q1 todos os modelos de classificação relacionados a esta questão utilizaram o arquivo de treinamento com valores das métricas SNA computadas para a rede de comunicação. A Tabela 2 apresenta os resultados da combinação dos algoritmos, com as diferentes configurações de treinamento.

**Tabela 2. Resultados da classificação a partir de valores de *Recall, Precision e F-Measure* para a Rede de Comunicação**

| Modelo de Classificação | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Algoritmo | Bayes – Net | | | Naive Bayes | | | Bayesian Logistic Regression | | |
| % de Treinamento | 50% | 66% | 80% | 50% | 66% | 80% | 50% | 66% | 80% |
| Recall | 1 | 1 | 1 | 0.6 | 0 | 0.5 | 1 | 0.67 | 0.5 |
| Precision | 0.38 | 0.33 | 0.4 | 0.42 | 0 | 0.33 | 0.38 | 0.25 | 0.25 |
| F-Measure | 0.55 | 0.5 | 0.57 | 0.4 | 0 | 0.4 | 0.55 | 0.36 | 0.33 |

Os modelos 1,2,3 e 7 apresentaram altos valores de *recall* para as construções que falharam. No entanto, os valores de *precision* são inferiores. Provavelmente este comportamento da classificação indica a dificuldade destes modelos em rejeitar uma construção com um estado diferente de falha. O melhor modelo construído para Q1 obteve *f-measure* de 0.57. Esta medida é calculada a partir da relação entre *Recall* e *Precision,* e mede quão calibrado o modelo de classificação pode estar. Segundo [Genkin et. al 2007], um modelo pode ser considerado ajustado às classes que se deseja prever, se o valor de *f-measure* for igual ou superior a 0.75.

O baixo desempenho de classificação para Q1 pode ser atribuído aos prováveis ruídos que as mensagens trocadas na lista de discussão possuem. Os participantes podem trocar mensagens dos mais variados assuntos, podendo conter mensagens que não são realmente efetivas ao desenvolvimento do projeto e estejam relacionadas diretamente com o motivo da falha da construção.

Estes ruídos podem criar relações inválidas na rede social. Mesmo considerando somente as mensagens de intervalos de tempo de uma construção, não foram aplicadas outras heurísticas para retirar mensagens que não fossem específicas sobre os arquivos alterados durante aquele intervalo de tempo.

Para responder Q2 foram realizados os mesmos testes, no entanto, as métricas SNA foram computadas a partir da rede de cooperação. A Tabela 3 apresenta os resultados para os nove modelos de classificação definidos com as matrizes de cooperação.

**Tabela 3. Resultados da classificação a partir de valores de *Recall, Precision e F-Measure* para a Rede de Cooperação**

| Modelo de Classificação | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|
| Algoritmo | Bayes – Net | | | Naive Bayes | | | Bayesian Logistic Regression | | |
| % de Treinamento | 50% | 66% | 80% | 50% | 66% | 80% | 50% | 66% | 80% |
| Recall | 0 | 0 | 1 | 0.71 | 1 | 1 | 0.85 | 0.8 | 0.66 |
| Precision | 0 | 0 | 0.75 | 0.62 | 0.71 | 0.75 | 0.54 | 0.67 | 1 |
| F-Measure | 0 | 0 | 0.85 | 0.66 | 0.83 | 0.85 | 0.66 | 0.73 | 0.8 |

Observa-se que, em geral, todos os modelos de classificação têm valores satisfatórios para as três medidas de avaliação do modelo de classificação proposto. Os modelos 10 e 11 apresentaram os piores desempenhos, estranhamente, retornando valores zerados para as métricas. Os modelos 12 e 15 apresentaram valores bem consistentes para *f-measure*, que indica a capacidade dos modelos de discernir entre construções no estado de falha ou de sucesso.

Por fim, repetiu-se os testes para a rede de comunicação e colaboração para responder Q3. A Tabela 4 apresenta os resultados obtidos quando as matrizes de comunicação e cooperação foram somadas.

**Tabela 4. Resultados da classificação a partir de valores de *Recall, Precision e F-Measure* para as Redes de Comunicação e Cooperação somadas.**

| Modelo de Classificação | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|
| Algoritmo | Bayes – Net | | | Naive Bayes | | | Bayesian Logistic Regression | | |
| % de Treinamento | 50% | 66% | 80% | 50% | 66% | 80% | 50% | 66% | 80% |
| Recall | 1 | 1 | 1 | 0.8 | 1 | 1 | 0.6 | 1 | 1 |
| Precision | 0.38 | 0.42 | 0.4 | 0.66 | 0.42 | 0.66 | 0.42 | 0.6 | 0.66 |
| F-Measure | 0.55 | 0.6 | 0.57 | 0.72 | 0.6 | 0.8 | 0.5 | 0.75 | 0.8 |

Quando os modelos foram combinados, esperava-se que a capacidade preditiva fosse

melhorada. Os modelos 21, 23, 24, 25 e não apresentaram melhora quando comparados os valores de *f-measure.*

Nos modelos 19, 20, 22, 26 foram constatados melhoras nos valores de f-measure. O modelo 22 apresentou o melhor índice de melhora.

É importante destacar que os modelos 19 e 20 estavam com valores estranhamente zerados em Q2, logo, a melhora deles se deu em função da comparação de f-measure de Q1 com o valor obtido em Q3. Nestes dois modelos, o modelo 20 melhorou em relação a *f-measure* de Q1, enquanto o modelo 19 ficou com o mesmo resultado já existente em Q1. O modelo 27 também não adicionou melhora quando foi realizada a combinação, no entanto, o resultado de *f-measure* também não foi reduzido.

Os melhores modelos propostos, para as três questões de pesquisa foram obtidos com conjuntos de treinamento 80/20. Este resultado era esperado, uma vez que uma maior quantidade de instâncias era utilizada para o treinamento o que permite potencializar o aprendizado do modelo de classificação.

## 5. Conclusão

Pesquisadores de engenharia de software têm coletado dados de interações entre desenvolvedores. Estes dados representam a comunicação e cooperação durante o desenvolvimento do software. Classificação é uma das técnicas utilizadas para entender a dinâmica do desenvolvimento do software e descobrir evidências que possam melhorar o processo de desenvolvimento.

Neste trabalho apresentamos uma abordagem para classificar construções de software utilizando redes de comunicação e cooperação. Mostramos que as redes de comunicação, pelo menos para o projeto Felix Gogo, tiveram os valores de *recall*, *precision* e *f-measure* abaixo de 0.60. Associamos este baixo desempenho em Q1 a prováveis ruídos existentes nas mensagens. Também mostramos que redes de colaboração (Q2) apresentaram resultados mais confiáveis para predizer construções no estado de falha, obtendo em alguns modelos valores superiores a 0.8 para classificação. Esperávamos que a unificação das duas redes (Q3) proporcionasse uma melhoria na classificação. Esta questão de pesquisa foi atendida parcialmente, pois alguns modelos melhoraram sua classificação e outros não.

Como trabalhos futuros, pretendemos estudar formas de diminuir o ruído sobre os dados da comunicação e repetir o experimento para mais projetos, adicionando também novos algoritmos de classificação realizando a avaliação por método de validação-cruzada.

Ferramentas poderiam ser construídas com base nesta abordagem para notificar engenheiros de software, líderes de projetos e desenvolvedores sobre potenciais problemas em construções futuras, o que poderia auxiliar a etapa de manutenção do software.

## Agradecimentos

Os autores deste artigo agradecem a Fundação Araucária pelo suporte financeiro.

## Referencias


SCHROTER, Adrian; ZIMMERMANN, Thomas; ZELLER, Andreas.(2006) Predicting Componentes Failures at Design Time". In Proceedings of the 5th International Symposium on Empirical Software Engineering (ISESE 2006), Rio de Janeiro, Brazil, September, pp.18-27.

BIÇER, Serdar; BENER, Ayse; CAGLAYAN, Bora. (2011) "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.

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

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

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

HASSAN, Ahmed E.(2008) The Road Ahead for Mining Software Repositories, In: Proceedings of the Future of Software Maintenance (FoSM), Beijing, China.

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

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

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

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

MENEELY, Andrew; WILLIAMS, Laurie.(2011) 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.

MENEELY, Andrew; WILLIAMS, Laurie; SNIPES, Will; Osborne, Jason.(2008) 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.

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

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

SHIN, Yonghee; MENEELY, Andrew; WILLIAMS, Laurie; OSBORNE Jason.(2010). Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities. In IEEE Transactions in Software Engineering (TSE).

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