# Do Historical Metrics and Developers Communication Aid to Predict Change Couplings?

I. S. Wiese, R. T. Kuroda, R. Ré, R. S. Bulhões, G. A. Oliva and M. A. Gerosa

***Abstract*— Developers have contributed to open-source projects by forking the code and submitting pull requests. Once a pull request is submitted, interested parties can review the set of changes, discuss potential modifications, and even push additional commits if necessary. Mining artifacts that were committed together during history of pull-requests makes it possible to infer change couplings among these artifacts. Supported by the Conway's Law, whom states that "organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations", we hypothesize that social network analysis (SNA) is able to identify strong and weak change dependencies. In this paper, we used statistical models relying on centrality, ego, and structural holes metrics computed from communication networks to predict co-changes among files included in pull requests submitted to the Ruby on Rails project. To the best of our knowledge, this is the first study to employ SNA metrics to predict change dependencies from Github projects.**

***Keywords*— structural holes metrics, social network analysis, change coupling, communication network, Conway's law.**

## I. Introdução

DURANTE a evolução do software, desenvolvedores se comunicam assincronamente durante a realização de tarefas relacionadas aos demandas reportadas por usuários, bem como, interagem para modificar artefatos. Como resultado das modificações, alguns artefatos evoluem conjuntamente durante o desenvolvimento do software [1]. Ball et al. [2] introduziram o conceito de acoplamento de mudança, que captura a noção de que alguns artefatos frequentemente mudam juntos, sugerindo que a força desse acoplamento está relacionada a quantidade de modificações casadas entre esses dois artefatos. Assim, a quantidade de mudanças conjuntas nos dois artefatos é diretamente proporcional ao acoplamento entre eles.

Alguns benefícios da análise dos acoplamento de mudança foram discutidos por D'Ambros e colegas [3]. Por exemplo, os acoplamentos de mudança podem revelar relacionamentos entre artefatos que, a princípio, estão escondidos, e não podem ser vistos no código ou na documentação do projeto. Outros pesquisadores mostraram que os acoplamentos de mudança podem afetar a qualidade de software, especialmente se relacionando com a quantidade de defeitos propagados no sistema. Zimmermann et al. [6] implementaram uma ferramenta para recomendar propagação de mudança a partir da alteração de um artefato. O pressuposto na sua abordagem é que as entidades que mudaram juntas muitas vezes no passado são propensas a mudar novamente no futuro.

Apesar dos benefícios do uso dos acoplamentos de mudança, pouco se sabe sobre as suas origens. Alguns pesquisadores [4–6] tentaram relacionar a ocorrência de acoplamentos de mudança com outros tipos de dependência, como por exemplo a estrutural. Eles concluíram que somente uma parte dos acoplamentos de mudança correspondem à dependência do tipo estrutural. No entanto, essas pesquisas não relacionaram a ocorrência de acoplamentos lógicos à estrutura de comunicação dos desenvolvedores e métricas obtidas a partir do histórico de modificação dos artefatos que formam o acoplamento de mudança.

Entender como os acoplamentos de mudança ocorrem pode ajudar desenvolvedores no planejamento do desenvolvimento da próxima release, no rastreamento dos efeitos da ocorrência das dependências de mudanças na arquitetura do software, bem como facilitar uma análise mais profunda dos acoplamentos de mudança para caracterizar o seu impacto na qualidade do software [3], [7].

Neste trabalho, nosso foco consiste em predizer a ocorrência de dois tipos de acoplamentos de mudanças: fortes e fracos. Por acoplamento de mudança forte, entende-se o subconjunto dos acoplamentos de mudança que mais ocorrem a partir de um limiar determinado. Uma vez que os acoplamentos de mudança identificados nesse trabalho são realizados a partir das mudanças nos artefatos, realizadas em tarefas de correção de defeitos e novas funcionalidades, os acoplamentos de mudanças classificados como "fortes" representam o grupo de artefatos que consumiram mais esforço de revisão de código e testes no passado.

Em um estudo anterior [8], nós identificamos a relação de quatro propriedades da rede de comunicação dos desenvolvedores com a ocorrência dos acoplamentos de mudanças fortes. A fundamentação do trabalho se apoiou na chamada "Lei de Conway" [9], que diz que as organizações que projetam sistemas são restringidas a produzirem projetos que são cópias de suas próprias estruturas de comunicação.

Baseado nessas descobertas, o objetivo deste trabalho é investigar quais outras métricas relacionadas tanto ao histórico

I. S. Wiese, Universidade Tecnológica Federal do Paraná (UTFPR), Campo Mourão, Paraná, Brasil, igor@utfpr.edu.br

R. T. Kuroda, Universidade Tecnológica Federal do Paraná (UTFPR), Cornélio Procópio, Paraná, Brasil, rodrigokuroda@gmail.com

R. Ré, Universidade Tecnológica Federal do Paraná (UTFPR), Campo Mourão, Paraná, Brasil, reginaldo@utfpr.edu.br

R. S. Bulhões, Universidade Federal da Bahia (UFBA), Salvador, Bahia, Brasil, rsbulhoes@gmail.com

G. A. Oliva, Universidade Estadual de São Paulo (USP), São Paulo, São Paulo, Brasil, goliva@ime.usp.br

M. A. Gerosa, Universidade Estadual de São Paulo (USP), São Paulo, São Paulo, Brasil, gerosa@ime.usp.br

das modificações conjuntas entre os artefatos, quanto à comunicação dos desenvolvedores, são úteis para a predição de acoplamentos de mudança fortes. Dessa forma, a seguinte questão de pesquisa foi investigada: *Quais métricas históricas e sociais podem auxiliar a predição de acoplamentos de mudança?*

Os principais resultados obtidos a partir da análise de três versões do projeto *Ruby on Rails* foram:

- Diferentes métricas obtidas a partir da rede de comunicação dos desenvolvedores podem ser úteis para prever acoplamentos de mudança. Entretanto, a dinâmica e intensidade da comunicação durante a evolução de software pode ser modificada e, consequentemente, influenciar os resultados.
- Releases com mais trocas de mensagens entre os desenvolvedores favoreceram a escolha das métricas sociais para construção do modelo de predição. No entanto, a quantidade de mudanças anteriores entre os acoplamentos de mudança foi o único preditor selecionado nas três versões.
- No âmbito da comunicação, as métricas da rede ego (*ego network*) e buracos estruturais (*structural holes*) foram as métricas mais frequentemente selecionadas.
- Inicialmente utilizamos 20 métricas nesse estudo, entretanto em torno de 1/3 das métricas foram suficientes após a execução do método AIC.

As próximas seções deste artigo estão organizadas da seguinte maneira. Na seção II, apresentamos o método de pesquisa, incluindo a coleta de dados, a construção das redes sociais, a identificação dos acoplamentos, as métricas selecionadas e o modelo de regressão logística. Na seção III, apresentamos e discutimos os resultados. Na seção IV, discutimos como este trabalho se compara com outros estudos da área. Na seção V, discutimos as ameaças à validade. Por fim, na seção VI, apresentamos nossas considerações finais e planos para trabalhos futuros.

## II. Método de Pesquisa

Nessa seção, descrevemos o procedimento de coleta dos dados, a técnica de identificação de acoplamentos de mudanças e como propomos a separação de acoplamentos de mudança em fortes e fracos. Também são descritos o conjunto de métricas e as técnicas estatísticas utilizadas em cada experimento.

### A. Coleta de Dados

Para este trabalho foi realizada a coleta de dados de três releases do projeto Ruby on Rails. O projeto foi escolhido considerando o número de "estrelas" e "*forks*" que tem atualmente no repositório Github. Essas duas informações são indicativos de popularidade. As estrelas indicam a quantidade de usuários que adicionaram o projeto na sua lista de "projeto favorito". O número de *forks* indica a quantidade de usuários que obtiveram pelo menos algum trecho do código do projeto. Normalmente, usuários que desejam contribuir com o projeto, realizam *fork* antes de submeter suas contribuições.

Nós usamos a API (GithubAPI can be access on: http://developer.github.com/) do Github para coletar as informações do projeto. Foram extraídas informações do código fonte, a partir do sistema de controle de versão, e metadados das contribuições e interações sociais dos desenvolvedores, a partir de cada *pull request* submetido ao projeto. No Github, um *pull request* representa a submissão de uma ou várias modificações nos artefatos do projeto. Esse mesmo *pull request* permite que desenvolvedores possam interagir trocando mensagens e sugerindo melhorias no código. Quando a solução final é obtida, o *pull request* é finalizado e o código é integrado no projeto original por um desenvolvedor membro do projeto.

Após a realização da coleta de dados, alguns pré-processamentos foram realizados. Primeiro, nós removemos arquivos com extensão de imagens, documentos e sem extensão. Essa remoção foi realizada, porque estamos interessados em prever o acoplamento de mudança para arquivos que representam a implementação do sistema. Também foram removidas todas as *pull requests* que não tiveram seu código integrado no projeto, uma vez que não é possível determinar se um *pull request* é importante ou não, correto ou não, se ele não tiver sido integrado ao projeto.

A

TABELA *I* apresenta a sumarização dos dados coletados. Cada versão constitui um conjunto de treinamento e de teste. As versões 3.1 e 4.2 não tem seus acoplamentos de mudanças preditos, uma vez que não temos dados da versão 3.0 para treinar o conjunto de dados, e não obtivemos dados após a release 4.1 para gerar um novo conjunto de teste.

A mesma tabela apresenta o número do primeiro e último *pull request* coletado para a release de teste (segunda linha do período de análise de cada conjunto de dados). Na versão de treino, as métricas históricas e de comunicação foram calculadas para cada acoplamento de mudança. Na versão de teste, os acoplamentos de mudança foram identificados e classificados no grupo forte ou fraco, de acordo com a quantidade de mudanças conjuntas que eles tiveram (mais detalhes na seção II.C).

Assim, o número de arquivos (# Arquivos), a quantidade total de acoplamentos de mudanças (Total AM), e a quantidade de acoplamento de mudança encontrada em cada grupo são referentes a release de teste. Abaixo da quantidade de acoplamentos de mudanças fortes e fracos (AM fracos, AM fortes) é apresentado o percentual de instâncias encontradas de acoplamentos de mudança de cada tipo. Por exemplo, no primeiro conjunto de dados, 70.98% dos acoplamentos de mudança encontrados estão no grupo de acoplamentos de mudança fracos, e consequentemente menos recorrentes.

### B. Construindo as redes sociais de comunicação.

Redes sociais são comumente representadas como um dígrafo G com um conjunto de arestas A e nós N. Cada desenvolvedor representa um nó na nossa rede. Cada aresta indica uma troca de mensagem entre dois desenvolvedores. Utilizamos o número de vezes que dois desenvolvedores trocaram mensagens como peso da aresta. Portanto, para

conectar os nós da rede, foram utilizadas a sequência de comentários dos desenvolvedores.

A Figura 1 mostra um exemplo de como construímos a rede social de comunicação a partir das mensagens escritas pelos desenvolvedores em *pull requests*. Os quadrados em azul indicam a existência de um *pull request* submetido por um contribuidor do projeto e que teve seu código integrado ao projeto. Os números e letras dentro dos quadrados indicam o número de cada *pull request* (#) e o nome de cada arquivo. Por exemplo, os arquivos A, B e D foram modificados para resolver a pendência relatada no *pull request* #2.

Os triângulos indicam o fluxo de mensagens enviadas pelos desenvolvedores durante a modificação dos arquivos. Cada triangulo, indica uma mensagem. Os números indicam qual foi o desenvolvedor que interagiu no *pull request*.

TABELA I. SUMARIZAÇÃO DOS DADOS DO PROJETO RUBY ON RAILS.

| Conjunto de dados | Período de Análise | Primeiro *pull request* | Ultimo *pull request* | # Arquivos | Total AM | AM fracos | AM fortes |
|---|---|---|---|---|---|---|---|
| #1 | (Versão 3.1) 01/07/2010 - 16/12/2010<br>(Versão 3.2) 16/12/2010 - 01/06/2011 | 4 | 133 | 917 | 268 | 190<br>70.89% | 78<br>29.11% |
| #2 | (Versão 3.2) 16/12/2010 - 01/06/2011<br>(Versão 4.0) 01/06/2011 - 16/11/2011 | 34 | 1440 | 1195 | 105 | 76<br>72.38% | 29<br>27.62% |
| #3 | (Versão 4.0) 01/06/2011 - 16/11/2011<br>(Versão 4.1) 16/11/2011 - 01/05/2012 | 1441 | 3641 | 856 | 171 | 112<br>65.49% | 59<br>34.51% |

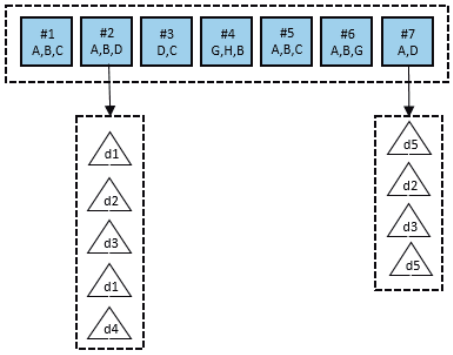


Figura 1. Coletando acoplamentos de mudança e interações dos desenvolvedores dos *pull requests*.

Para cada acoplamento de mudança foram agrupadas todas as mensagens dos *pull requests* onde eles ocorreram, uma vez que acoplamentos de mudança diferentes mudaram em um conjunto de *pull requests* diferentes. Por exemplo, os arquivos A e D foram modificados nos *pull requests* #2 e #7. No *pull request* #2, a primeira mensagem foi enviada pelo desenvolvedor d1, seguido por d2, d3 e novamente d1 e d4. Para o *pull request* #7, os desenvolvedores d5, d2, d3 e novamente d5 postaram mensagens.

Por exemplo, nos *pull requests* #2 e #7 é possível observar que 5 desenvolvedores trocaram mensagens quando o acoplamento de mudança A-D foi capturado. Para criar a rede social, nós primeiramente criamos o nó d1. Depois, foi criado o nó d2 e uma aresta foi criada entre d1 e d2 usando uma aresta direcional apontando d2 para d1 com o peso 1

Após esse momento a mensagem de d3 foi considerada parte da rede. Nós adicionamos um nó d3. Para todos os desenvolvedores anteriores, foi criada uma aresta de ligação. Nesse caso, temos uma aresta de d3 para d2 e uma aresta de d3 para d1. Seguindo esse procedimento sempre foi verificado se um nó já pertencia a rede. Se o desenvolvedor não existisse, nós adicionávamos o nó do respectivo desenvolvedor na rede e conectávamos esse novo nó com todos os nós existentes no mesmo *pull request*. A mesma verificação era realizada com as arestas. Quando uma aresta já existia o peso do número de mensagens era incrementado em "+1".

### *C. Identificando acoplamentos de mudança.*

A técnica de associação de regras tem sido utilizada para identificar acoplamentos de mudança. Essa técnica é baseada em aprendizagem não supervisionada usada para detecção de padrões [10].

Uma regra de associação $A_i \Rightarrow A_j$ entre dois arquivos significa que se uma mudança aconteceu em $A_i$, também deveria acontecer em $A_j$. Nesse trabalho, uma regra de associação significa o acoplamento de mudança entre dois artefatos que mudaram em um conjunto de *pull requests*. Similarmente a montagem da rede de comunicações, para determinar os pares de artefatos nós precisamos percorrer todos os *pull requests* que tiveram código integrados no projeto em cada release.

Por exemplo, o acoplamento de mudança A-D ocorreu nos *pull requests* #2 e #7, entretanto, o acoplamento A-B ocorreu nos *pull requests* #1, #2, #5 e #6. Dessa forma, para identificar os acoplamentos de mudança é necessário agrupar todos os *commits* realizados em cada *pull request*, combinando todos os artefatos modificados em pares, removendo os arquivos duplicados que foram modificados mais de uma vez em um único *pull request*.

Regras de associação utilizam medidas de interesse e significância que usualmente são dadas por limiares de *suporte* e *confiança*. Nesse estudo, a medida de suporte denota o número de vezes que dois artefatos foram modificados conjuntamente. A medida de confiança define o grau em que artefatos estão logicamente conectados, caracterizando assim a força da relação. Assim, utilizando essas duas medidas é possível saber a força e significância de um acoplamento de mudança com base no histórico de modificações dos artefatos considerando o conjunto de *pull requests* de cada release. As formulas são definidas a seguir [11] :

$$Suporte = \text{NM conjuntas de Ai e Aj}$$

$$Confiança = \frac{\text{NM conjuntas de Ai e Aj}}{\text{NM Fi ou NM Fj}}$$

onde, NM é o número de modificações conjuntas entre dois artefatos, e Ai e Aj são dois artefatos distintos. Para calcular o suporte basta somar a quantidade de modificações conjuntas em um conjunto de *pull requests* que tiveram seu código integrado ao projeto em uma release.

Para calcular a confiança, o número de modificações conjuntas de dois artefatos é dividido pelo número de modificações do artefato que tem o maior número de modificações. É importante observar que os valores de confiança para $A_i \Rightarrow A_j$ e $A_j \Rightarrow A_i$ podem ser diferentes. Por exemplo o artefato A e B foram modificados conjuntamente 4 vezes. Entretanto A foi modificado em 5 *pull requests*, enquanto B foi modificado em 4 *pull requests*. Nesse caso a confiança de A ⇒ B = 4/5=80%, enquanto a confiança de B ⇒ A = 4/4 = 100%. Nesse caso podemos dizer que todas as vezes que A foi modificado, B também foi modificado conjuntamente [11].

Como limiar de suporte mínimo nós consideramos a mudança conjunta em pelo menos dois *pull requests*. Dessa forma, acoplamentos de mudança que ocorreram somente uma vez foram removidos. Acoplamentos de mudança com confiança inferior a 40% também foram removidas.

Para classificar os pares de artefatos em fortes e fracos, nós usamos a distribuição e análise dos quartis sobre a quantidade de co-mudanças entre os artefatos (valor de suporte). Acoplamentos de mudança com valores de suporte igual ou acima do terceiro quartil foram classificados como fortes.

### *D. Métricas históricas e de comunicação*

Para esse trabalho dois diferentes grupos de métricas foram calculados. Primeiramente, nós utilizamos os metadados do histórico de modificações para calcular as métricas históricas.

Para as métricas históricas foram calculados o número de vezes que cada acoplamento de mudança, mudou conjuntamente na release anterior (*updates*), o número de desenvolvedores distintos que enviaram mensagens no conjunto de *pull requests* que formou o acoplamento de mudança (*commenters*), o número de comentários que um acoplamento de mudança teve (*comments*), a soma do número de linhas adicionadas e removidas de cada um dos artefatos que formam um acoplamento de mudança (*codeChurnFile, codeChurnFile2*), e a média de linhas adicionadas e removidas dos dois artefatos (*codeChurnAVG*).

As métricas de comunicação foram calculadas com base na rede social de comunicação extraída do conjunto de *pull requests* onde o acoplamento de mudança foi formado. Para realizar a construção das redes e o cálculo das métricas foi utilizada a API JUNG (JUNG API can be access on: jung.sourceforge.net/).

Algumas das métricas são utilizadas para caracterizar um simples nó e seus vizinhos (*ego networks*), enquanto outras medidas são relativas a estrutura da rede, como por exemplo os buracos estruturais (*structural hole*). Mais detalhes sobre métricas de análise de redes sociais podem ser obtidos em Wassermman e Faust [12].

TABELA II. CONJUNTO DE MÉTRICAS OBTIDAS DA REDE SOCIAL DE COMUNICAÇÃO.

| Métrica | Tipo de Métricas | Definição |
|---|---|---|
| AVG das medidas de centralidade | Betweenness Centrality (btwAvg), Closeness (clsAvg) Degree Centrality (dgrAvg) | Quantifica quão perto os nós são indiretamente ligados a outros nós na rede. |
| AVG das medidas de nó (ego measures) | Ego Density (egoDensityAvg), Ego Ties (egoTiesAvg) Ego Size (egoSizeAvg) Ego Betweenness *(egoBtwAvg)* | Medidas ego baseiam-se na vizinhança de qualquer nó específico. O nó que está sendo avaliado é denotado ego. O "Bairro" inclui o conjunto de nós ligados ao ego por arestas mais o próprio ego. As medidas de ego são obtidas das subredes formadas a partir de todos os nós. |
| AVG das medidas de Burados Estruturais (Structural Hole Metrics-SHM) | Effective Size (efvSizeAvg), Efficiency (efficiencyAvg), Hierarchy(HierarchyAvg) | SHM diz respeito à noção de redundância de redes e o grau em que existem falta de ligações entre nós. SHM denota as lacunas entre os nós de uma rede social ou representa as pessoas de ambos os lados do "buraco" indicando que existe uma diversidade de fluxo de informação na rede. |
| Diâmetro | Propriedade da Rede | O diâmetro é definido como sendo a máxima, sobre todos os pares de vértices *u*, *v*, do comprimento do percurso mais curto a partir de *u* para *v* |
| Densidade | Propriedade da Rede | A densidade é calculada como a percentagem das ligações existentes para todas as ligações possíveis na rede. |

mais detalhes dessas métricas podem ser encontradas em: [13–16]

Uma vez que as métricas de rede social são calculadas por nó, é necessário agregar os valores de todos os participantes da discussão de um acoplamento de mudança em uma única medida. Dessa forma, muitas das medias apresentadas utilizam a média da medida de todos os nós para converter o valor para um acoplamento de mudança.

### *E. Modelo de Regressão Logística*

Modelos de regressão logística são usados para produzir uma estimativa da probabilidade para uma variável dependente assumir um determinado valor. Nessa classe de modelos de regressão, a variável dependente deve ser dicotômica, tornando necessária a separação dos acoplamentos de mudança em dois grupos. Para esse trabalho, nós categorizamos os acoplamentos lógicos em fortes e fracos, com base no valor de suporte de um acoplamento de mudança.

Todas as métricas calculadas e descritas na subseção anterior foram usadas como preditores candidatos no modelo. Entretanto, para usar a regressão logística com múltiplos preditores é necessário assumir que todos os preditores são independentes. Na prática, sabe-se que muitas medidas têm níveis de correlação altos, o que torna duas medidas dependentes uma da outra podendo levar o modelo a problemas de sobreajustamento (*overfitting*) e consequentemente a perda de poder preditivo [17].

Para resolver esse problema nós analisamos a correlação de *Spearman* entre as métricas calculadas de cada acoplamento de mudança e usamos um método de seleção de modelos chamada *Akaike Information Criterion* (AIC). O método AIC é uma medida relativa da qualidade estatística do modelo para um determinado conjunto de modelos candidatos e conjunto de dados. O melhor modelo, apresentará o menor valor de AIC. Esse método não somente trata o ajustamento do modelo (*goodness of fit*), mas também penaliza modelos que necessitam de muitas variáveis independentes para produzir bons resultados. Nesse sentido, o AIC é útil para reduzir o risco de sobreajustamento dos modelos gerados. Para calcular o AIC a seguinte fórmula deve ser utilizada: $AIC = 2k - \ln(L)$, onde *k* é o número de parâmetros ou variáveis independentes no modelo e *L* é o valor maximizado da função de verossimilhança para o modelo [18].

Desta forma, para cada modelo válido em cada release nós reportamos o percentual de desvio explicado do modelo ($\chi 2$). O Desvio é definido como -2 vezes o *log* da probabilidade do modelo. O percentual de desvio explicado é a taxa do desvio do modelo nulo contendo somente o intercepto (quando o modelo não tem nenhuma métrica) e o modelo final obtido. Nós testamos e reportamos os valores de p usando o teste de qui-quadrado [19].

Ao invés de reportar os coeficientes de regressão, nós reportamos as razões de chance (*odds ratio*). Essa abordagem é amplamente usada em engenharia de software no que diz respeito a construção de modelos de regressão logística [7], [20]. A razão de chance é obtida aplicando a função exponencial no coeficiente de regressão de uma variável dependente. Valores maiores do que 1 indicam relação positiva entre a variável dependente e independente. O relacionamento negativo é denotado por valores da razão de chance entre 0 e 1.

## III. Resultados

### A. Análise preliminar: Correlação entre os preditores

Para evitar os problemas de multicolinearidade e sobreajustamento do modelo foi realizada a análise de correlação entre os preditores.

A correlação de Spearman (*rho*) é uma medida não-paramétrica de dependência estatística entre duas variáveis. O valor retornado pelo teste pode variar entre +1 (correlação positiva) e -1 (correlação negativa).

A Fig. 2 sumariza a correlação entre pares de nossos 20 preditores candidatos em um correlograma. Para cada par de variável, nós reportamos a força da correlação como um quadrado com escala gradiente de cores. Nós usamos cor vermelha para correlações negativas e cor azul para correlações positivas. Quanto mais escuro, mais positiva ou negativa é a correlação.

Além do valor da correlação o correlograma apresenta o *valor de p* do teste realizado. Os seguintes *valores de p* são apresentados: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. Essa visualização nos permite identificar relações que precisam de mais atenção durante a construção dos modelos de regressão logística.

Nós observamos correlações positivas entre *codeChurnAVG* e as medidas de *codechurn* de cada arquivo (*codeChurnFile* e *codeChurnFile2*). Essa correlação foi esperada uma vez que *codechurnAVG* foi computada usando as medidas individuais de cada arquivo. O mesmo aconteceu com a relação positiva encontrada entre a quantidade de desenvolvedores que comentaram (*commenters*), número de comentários (*comments*), tamanho da rede (*size*), *ties* e diâmetro (*diameter*). Da mesma forma, essa correlação positiva era esperada uma vez que elas representam medidas da propriedade topológica das redes sociais de comunicação.

Considerando as correlações negativas, nós observamos que *codechurnAvg* é negativamente correlacionada com a maior parte das métricas. Nós também ressaltamos a correlação negativa do número de acoplamentos de mudança detectados na versão anterior (*updates*) com a maior parte das métricas. Observamos também que as medidas de propriedade da topologia de redes sociais (*diameter, size, and ties*), e a quantidade de comentários (*commenters*) e da quantidade de desenvolvedores que comentaram (*commenters*) são fortemente correlacionados.

A adição de métricas redundantes é justamente poder explorar quais das métricas possibilitam a melhor construção do modelo, uma vez que métricas de centralidade diferentes, podem por exemplo, capturar aspectos diferentes dos grupos de acoplamento de mudança forte e fraco. É importante mencionar que o uso de correlação só permite observar o padrão de relação entre duas variáveis e não determina causa-efeito. Entretanto, como dito anteriormente, esse teste é importante para construir modelos com menor chance de sobreajustamento e problemas relacionados a multicolinearidade.

### B. Quais métricas históricas e sociais podem auxiliar a predição de acoplamentos de mudança?

Para investigar se as métricas históricas e as métricas de rede social obtidas da comunicação dos desenvolvedores podem auxiliar a predição de acoplamentos de mudanças fortes e fracos, nós executamos a análise de regressão logística como descrito na Seção III.D.

Para cada modelo, foi executado o método AIC e são reportados a razão de chance de cada coeficiente de regressão do preditor selecionado pelo modelo como relevante para explicar os acoplamentos de mudança. Além da razão de chance nós reportamos o ajuste do modelo ($\chi 2$), e o percentual de variação dos dados que é explicado por cada modelo (Dev. Expl). Por fim, nós reportamos a Curva *Receiver Operating Characteristic* (ROC).

A Curva ROC é uma representação gráfica que ilustra a performance do classificador binário para diferentes limiares de sensitividade e a taxa de falso positivos. A Área Sob a Curva ROC (AUROC, sigla do inglês *Area Under the ROC Curve*) indica se é igual à probabilidade de que um classificador irá classificar um exemplo positivo escolhido aleatoriamente maior do que um valor negativo escolhido aleatoriamente. Os valores de AUROC podem variar de 0 e 1. Valores próximos a 0.5 indicam um classificador randômico, sendo desejáveis pelo menos valores acima de 0.75. Essas

medidas foram escolhidas porque podem prover informação sobre a aplicabilidade da técnica em contextos práticos.

Devido a distribuição assimétrica dos nossos preditores foi aplicada a transformação por meio da técnica de Box-Cox. Essa técnica sugere eventuais transformações-potência (recíproca, raiz quadrada, quadrática, cúbica etc) nas variáveis baseadas em um valor de *lambda* [21].

A TABELA III apresenta os resultados dos modelos de predição para os acoplamentos de mudança fortes e fracos identificados na Release 3.2 utilizando as métricas calculadas na Release 3.1 como preditores.

Os resultados indicam que cinco métricas foram selecionadas para o modelo final após a execução do método AIC. Todas as métricas tiveram efeito positivo com a ocorrência dos acoplamentos de mudança fortes.

Figura 2. Correlação de Spearman para cada par de preditor.

Nosso primeiro modelo apresentou duas métricas históricas e outras três métricas obtidas dos aspectos de comunicação. O número de mudanças anterior e o *codeChurnAVG* foram as métricas históricas selecionadas. Já para as métricas de comunicação, as métricas selecionadas foram *egoDensity*, *Density* e *efficiencyAvg*.

É importante mencionar que duas métricas relativas a estrutura da rede (densidade e *efficiency*) foram selecionadas para essa versão. Esse resultado sugere que a estrutura da rede é mais importante que as métricas de papéis de um desenvolvedor, que pode ser obtido com as métricas de centralidade. Todos os preditores selecionados foram significativamente relevantes para $p < 0.05$ (95%). Pode-se observar que 29% do conjunto de dados foi explicado e ajustado às métricas. Apesar do valor baixo, em problemas de engenharia de software é muito difícil encontrar modelos de regressão logística com valores muito superiores a este.

TABELA III. MODELO RESULTADO DA APLICAÇÃO DO AIC PARA PREDIZER ACOPLAMENTOS DE MUDANÇA FORTES E FRACOS PARA VERSÃO 3.2.

| Variáveis Independentes | Razão de Chance | Significância Estatística |
|---|---|---|
| egoDensityAvg | 87.34 | $p < 0.05$ |
| log(efficiencyAvg) | 27721.94 | $p < 0.05$ |
| Density | 4760171.00 | $p < 0.05$ |
| log(codeChurnAVG) | 1517658.00 | $p < 0.05$ |
| Updates | 717256.20 | $p < 0.05$ |
| **Ajuste do Modelo ($\chi 2$)** | **256.93 (p = 0.524) df=259** | |
| **Dev.Expl** | **29.80%** | |
| **AUCROC** | **0.787** | |

df = graus de liberdade

Um indicativo importante é que o valor de ajuste do modelo ($\chi 2$) após a aplicação do teste qui-quadrado apresentou p = 0.524. Neste teste, o valor de p retornado é maior que 0.05, indicando que o modelo tem bom ajuste não pode ser rejeitado. Por fim, o valor de AUROC obtido (0.787) foi superior a 0.75 recomendado, logo podemos concluir que o modelo poderia ser utilizado com relativa precisão para prever acoplamentos de mudanças fortes e fracos.

A TABELA IV mostra os resultados de predição para os acoplamentos de mudança fortes e fracos identificados na Release 4.0. O segundo modelo gerado obteve valores melhores de AUROC (0.90), de ajuste nos dados ($\chi 2 = 70.099$, p=0.978) e de explicação dos dados (50.42%) com todos os preditores selecionados com significância estatística.

Após a execução do método AIC o modelo selecionado conta com seis preditores. Nesse modelo as métricas de comunicação foram mais selecionadas do que as métricas históricas. É importante ressaltar, que uma métrica de cada tipo foi selecionada em relação as métricas de comunicação. Por exemplo *dgrAVG* é uma métrica de centralidade,

*egoTiesAvg* é uma métrica da rede ego, *efvSize* e *hierarchyAVG* são métricas de buracos estruturais. Além dessas métricas, o número de comentários e número de co-mudanças anteriores (*updates*) foram considerados preditores relevantes. Esse foi o único modelo que não foi necessário realizar transformações nos preditores para ajustar as variáveis independentes a variável dependente.

A TABELA V apresenta os resultados para a Release 4.1. Os resultados se assemelham da versão 3.2. Os valores das métricas de AUROC (0.78) indicam um modelo consistente com a aplicação prática, entretanto, comparado com a release 4.0 houve uma redução na explicação que ficou em torno de 20.32%. Novamente, todos os preditores foram estatisticamente significantes para $p < 0.05$ e o ajuste do modelo rejeitou a hipótese nula indicando ajuste do modelo aos dados (p = 0.290).

TABELA IV. MODELO RESULTADO DA APLICAÇÃO DO AIC PARA PREDIZER ACOPLAMENTOS DE MUDANÇA FORTES E FRACOS PARA VERSÃO 4.0.

| Variáveis Independentes | Razão de Chance | Significância Estatística |
|---|---|---|
| dgrAvg | 2.16 | p < 0.05 |
| egoTiesAvg | 0.92 | p < 0.05 |
| efvSizeAvg | 0.61 | p < 0.05 |
| hierarchyAvg | 190.48 | p < 0.05 |
| comments | 1.19 | p < 0.05 |
| Updates | 0.23 | p < 0.05 |
| **Ajuste do Modelo ($\chi 2$)** | **70.099 (p= 0.978) df=96** | |
| **Dev.Expl** | **50.42%** | |
| **AUROCC** | **0.900** | |

df = graus de liberdade

Após a execução do AIC o modelo resultante selecionou seis variáveis. Duas métricas históricas (*updates* e *codeChurn2*), 3 métricas de rede ego (*egoBtwAvg*, *egoSizeAvg*, *egoDensityAvg*) e uma métrica de buracos estruturais (*efficiencyAvg*). Todas as métricas com exceção de egoDensity tiveram relacionamento positivo com a variável dependente.

TABELA V. MODELO RESULTADO DA APLICAÇÃO DO AIC PARA PREDIZER ACOPLAMENTOS DE MUDANÇA FORTES E FRACOS PARA VERSÃO 4.1.

| Variáveis Independentes | Razão de Chance | Significância Estatística |
|---|---|---|
| sqrt(egoBtwAvg) | 1441911.00 | p < 0.05 |
| sqrt(egoSizeAvg) | 64899.35 | p < 0.05 |
| egoDensityAvg | 0.0000431426 | p < 0.05 |
| log(efficiencyAvg) | 6128.67 | p < 0.05 |
| log(codeChurnFile2) | 1876044.00 | p < 0.05 |
| log(updates) | 104842.80 | p < 0.05 |
| **Ajuste do Modelo ($\chi 2$)** | **170.40 (p= 0.290) df=161** | |
| **Dev.Expl** | **20.32%** | |
| **AUROCC** | **0.790** | |

df = graus de liberdade

Comparando os resultados dos três modelos, nós observamos que a variável independente número de mudanças da Release anterior (*updates*) foi a única métrica selecionada em todas as Releases, o que mostra evidências que pares de arquivos que são mais modificados em uma versão tendem a ocorrer novamente na próxima Release. Outras duas métricas históricas foram selecionadas no modelo das versões 3.1 e 4.0.

Nesses modelos, nós verificamos que as métricas relacionadas a adição e remoção das linhas de código que formam o acoplamento de mudança foram selecionados. É importante mencionar que nessas duas versões, houve menos comunicação entre os desenvolvedores durante a realização das tarefas como mostra a Figura 3. Na versão com maior intensidade (release 4.0), com exceção da métrica *updates*, todas as métricas selecionadas têm relação com aspectos de comunicação.

### C. Resumo das contribuições.

Considerando os resultados, nós encontramos evidências que corroboram com nosso primeiro estudo [8]. Nesse estudo nós comparamos algumas métricas de buracos estruturais das redes sociais de comunicação dos desenvolvedores e métricas de processo. Nesse primeiro experimento, não foram encontradas diferenças estatísticas entre os dois conjuntos de métricas históricas e de comunicação para prever a ocorrência de dependência de mudança fortes e fracos.

Neste estudo, entretanto, nós estendemos o conjunto de métricas, especialmente as métricas sociais relacionadas a comunicação dos desenvolvedores. Pôde-se observar que essas novas métricas foram selecionadas como preditores relevantes. Os principais resultados obtidos foram:

- Diferentes métricas obtidas da comunicação dos desenvolvedores podem ser úteis para prever acoplamentos de mudança. Entretanto, a dinâmica e intensidade da comunicação durante a evolução de software pode ser modificada e consequentemente influenciar os resultados. Por exemplo, mais e novos desenvolvedores podem contribuir de diferentes formas no projeto, e isso pode influenciar se as métricas de comunicação podem ser bons preditores ou não, uma vez que cada métrica capturara aspectos diferentes das interações entre os desenvolvedores.
- Considerando os tipos de métricas de comunicação, foi possível identificar que as métricas da rede ego (*ego network*) e buracos estruturais (*structural holes*) foram as métricas mais frequentemente selecionadas. Isso mostra que os papéis dos desenvolvedores (métricas de centralidade) são menos relevantes para prever a ocorrência os acoplamentos de mudança.
  - Inicialmente, nós utilizamos 20 métricas nesse estudo, entretanto em torno de 1/3 das métricas foram suficientes após a execução do método AIC para prever a ocorrência dos acoplamentos de mudança.
  - A métrica mais selecionada foi a quantidade de modificações anteriores que os artefatos tiveram. O número de comentários e a quantidade de linhas adicionadas e removidas *(codeChurn*) também foram selecionados em duas releases.

## IV. Trabalhos Relacionados

### A. Rede social de comunicação entre desenvolvedores

Para entender o impacto das estruturas de comunicação, trabalhos anteriores têm utilizado análises baseada em grafos. De fato, esse tipo de análise tem sido utilizado para estudar

diferentes fenômenos da engenharia de software, e tem se mostrado útil para auxiliar a predição de defeitos [7], [13–16], [22], [23], estimar à severidade dos defeitos [22], triagem de defeitos [24] e vulnerabilidade de um artefato [25].

Especificamente, no que diz respeito ao uso da rede social de comunicação, alguns trabalhos são destacados. Wolf et al. [14] reportou resultados indicando que a rede social de comunicação assume um importante papel na qualidade do software para predizer quando a construção do software é propensa a falhar. Eles construíram modelos de predição para o projeto IBM Jazz atingindo taxas de sensitividade (*recall*) entre 55% e 75% e taxas de precisão entre 50% e 70%.

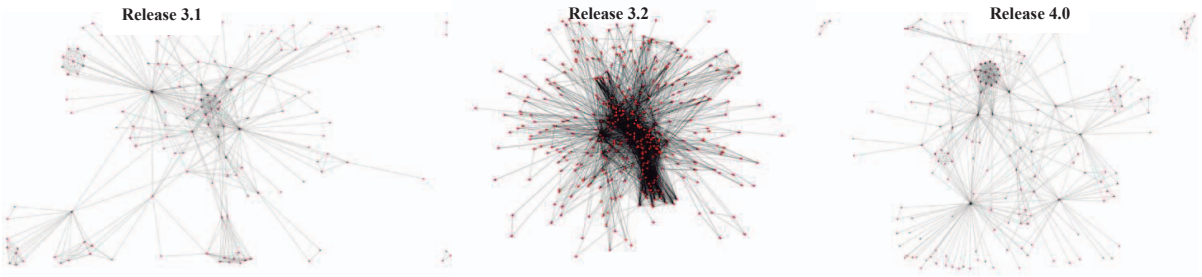


Figura 3. Representação Gráfica da rede social dos desenvolvedores que comentaram nas tarefas de cada release que formaram os acoplamentos de mudança.

Biçer et al. [26] indicou que métricas obtidas da rede social de comunicação dos desenvolvedores dos projetos Drupal e IBM Jazz permitem a redução considerável de taxas de falsos positivos nos modelos de predição de defeitos.

Nenhum desses trabalhos utilizou informações de comunicação ou redes sociais para identificar a ocorrência do acoplamento de mudança em projetos de software. Nosso trabalho utiliza subconjunto de métricas similares aos utilizados por Wolf et al. [14] e Biçer et al. [26], entretanto nosso foco é diferente.

### *B. Acoplamento de Mudanças*

Alguns estudos em acoplamento de mudança focaram na sua relação com defeitos em projetos de software livre. Por exemplo, Kouroshfar [27] usou modelos estatísticos para investigar o impacto da dispersão de co-mudanças na qualidade do software. Os resultados revelaram que co-mudanças que incluem arquivos de diferentes subsistemas resultam em mais defeitos do que co-mudanças que não envolvem subsistemas diferentes.

D'Ambros et al. [28] executou estudo em três sistemas de software (ArgoUML, Eclipse JDT e Mylyn) e encontrou uma correlação maior entre acoplamento de mudanças e defeitos do que com as métricas de complexidade de código e defeitos.

Zimmermman et al. [11] construiu um *plug-in* que coleta informações sobre as mudanças do código fonte do sistema de controle de versão possibilitando ao desenvolvedor saber com que outros artefatos um dado artefato foi modificado no passado. Os autores reportaram taxas de precisão em torno de 30% e recomendaram que projetos com continua evolução a ferramenta deveria considerar a recomendação em nível de arquivo ao invés de métodos.

Zhou et al. [29] propôs um modelo de predição para identificar quais arquivos eram propensos a ter acoplamento com outro arquivo ou não. Nesse trabalho, os autores extraíram métricas da dependência estática do código fonte, a frequência de co-mudança, a idade da mudança, o autor da mudança e o tipo da requisição de mudança. Os autores reportaram taxas de precisão em torno de 60% e sensitividade em torno de 40%.

Outros pesquisadores [30], [31] mostraram que acoplamentos de mudanças podem ser identificadas usando regras de associação e séries temporais de Granger.

Neste trabalho nós utilizamos as regras de associação, seguindo a abordagem de Zimmermman [11] para identificar os acoplamentos de mudanças. Entretanto, baseado no limiar de terceiro quartil nós categorizamos os acoplamentos de mudanças em fortes e fracos, hipotetizando que os acoplamentos de mudanças fortes – que mais aconteceram no passado – são mais relevantes e devem ser priorizados para tarefas de revisão de código, refatoração e testes. Dessa forma, nosso trabalho difere dos demais porque tenta prever se um acoplamento de mudança novo tem características históricas e de comunicação que permitam a identificação de qual grupo esse acoplamento deva pertencer.

## V. AMEAÇAS À VALIDADE

Algumas ameaças a validade podem afetar os resultados desse estudo. A primeira ameaça é relativa a generalização. Durante nossa análise, nós apresentamos um único estudo de cados envolvendo três releases do Ruby on Rails. Dessa forma, o escopo do nosso resultado se limita ao projeto Ruby on Rais. Projetos de características semelhantes, como por exemplo, de organização e contribuição podem apresentar resultados semelhantes. Entretanto, não podemos afirmar que nossos resultados podem ser generalizados para outros projetos e domínios. Em contrapartida, a escolha de um único projeto nos permite controlar e entender melhor os dados analisados.

No que diz respeito a identificação dos acoplamentos de mudança, nós percebemos diferenças entre o primeiro trabalho e o segundo trabalho. Quando usamos um período temporal, um conjunto de acoplamentos de mudanças foi identificado. Nesse estudo agrupamos os commits e tarefas por releases, e o

resultado da execução da técnica de geração de dependências de mudanças apresentou resultados um pouco diferentes, mas não resultou em modelos com resultados muito diferentes dos apresentados no nosso estudo anterior [8].

Outra ameaça identificada na identificação dos acoplamentos de mudanças é relacionado ao "tangle commits" [32]. Esse termo é relacionado a interação do Sistema de controle de versões e os desenvolvedores, uma vez que os desenvolvedores têm hábitos diferentes para realizar commits dos arquivos modificados no sistema de controle de versão, que por vezes, pode adicionar arquivos não relacionados ou "fora do contexto" da tarefa que se está resolvendo. No nosso estudo, essa ameaça é limitada, já que nós agrupamos os commits realizados por *pull request* que foram aceitos e integrados dentro do projeto. Esse processo normalmente é inspecionado por desenvolvedores que fazem parte do núcleo que mantêm o projeto.

Por último, a seleção de métricas pode trazer resultados diferentes. Nosso conjunto de métricas pode não ser o mais completo possível. Entretanto, nós também procuramos selecionar métricas que capturassem diferentes aspectos da comunicação dos desenvolvedores, como por exemplo, métricas de ego network, de centralidade e buracos estruturais. Para diminuir essa ameaça nós selecionamos as métrica cuidadosamente, procurando encontrar métricas que já foram utilizadas em outros trabalhos que envolvessem modelos de predição conforme identificamos em uma revisão sistemática anterior [33].

## VI. Conclusões e Trabalhos Futuros

Nesse estudo, apoiados pela Lei de Conway, nós avaliamos o uso de métricas históricas e da rede de comunicação dos desenvolvedores para predizer a ocorrência de acoplamentos de mudança fortes (mais recorrentes) e fracas (menos recorrentes) no projeto Ruby on Rails.

Os resultados indicaram que um subcojnjunto de métricas históricas e de comunicação são estatisticamente significantes para as três versões do projeto. Na versão com maior interação social – rede social com maior troca de mensagens entre os desenvolvedores – uma maior parte de preditores relacionados a comunicação foram selecionados, especialmente as métricas relacionadas a estrutura da rede (*ego network* e *structural hole*), ao invés de métricas relacionadas ao papel dos desenvolvedores (métricas de centralidade). Em releases com menos interação social, as métricas históricas apareceram em maior quantidade. A métrica relativa a quantidade de co-mudanças anteriores foi a única métrica que foi relevante nas três versões estudadas.

Esse estudo mostrou evidências que a combinação de métricas históricas e de comunicação são úteis para identificar artefatos que tendem a ser modificados conjuntamente, e com maior frequência em tarefas de manutenção e evolução do software.

Em trabalhos futuros, nós queremos investigar novos projetos e métricas analisando mais profundamente a relação das métricas obtidas durante o desenvolvimento de software e a sua capacidade de prever artefatos que são modificados conjuntamente, uma vez que o acoplamento de mudança é relacionado com a ocorrência de defeitos em releases futuras.

## Agradecimentos

Nós gostaríamos de agradecer a Fundação Araucária que financia o Dinter UTFPR-IME/USP, o NAPSOL, a FAPESP, o NAWEB e o CNPQ (461101/2014-9 – Projeto Universal) por todo o suporte financeiro. Marco Aurério Gerosa recebe bolsa individual. Igor Wiese recebeu bolsa do Ciência sem Fronteiras CAPES (BEX 2039-13-3). Gustavo O. recebeu bolsa do Ciência sem fronteiras CAPES (250071/2013-4).

## Referências

[1] G. Canfora, L. Cerulo, M. Cimitile, and M. D. Penta, "How changes affect software entropy: an empirical study," *Empirical Software Engineering*, vol. 19, no. 1, pp. 1–38, 2014.

[2] T. Ball, J.-M. K. Adam, A. P. Harvey, and P. Siy, "If Your Version Control System Could Talk...," in *ICSE Workshop on Process Modeling and Empirical Studies of Software Engineering*, 1997.

[3] M. D'Ambros, M. Lanza, and M. Lungu, "Visualizing Co-Change Information with the Evolution Radar," *Software Engineering, IEEE Transactions on*, vol. 35, no. 5, pp. 720–735, 2009.

[4] G. A. Oliva, F. W. Santana, M. A. Gerosa, and C. R. B. de Souza, "Towards a classification of logical dependencies origins: a case study," in *EVOL/IWPSE*, 2011, pp. 31–40.

[5] G. A. Oliva and M. A. Gerosa, "On the Interplay between Structural and Logical Dependencies in Open-Source Software," in *SBES*, 2011, pp. 144–153.

[6] M. M. Geipel and F. Schweitzer, "The Link between Dependency and Cochange: Empirical Evidence," *IEEE Trans. Software Eng.*, vol. 38, no. 6, pp. 1432–1444, 2012.

[7] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software Dependencies, Work Dependencies, and Their Impact on Failures," *Software Engineering, IEEE Transactions on*, vol. 35, no. 6, pp. 864–878, 2009.

[8] I. S. Wiese, R. T. Kuroda, D. N. R. Junior, R. Ré, G. A. Oliva, and M. A. Gerosa, "Using Structural Holes Metrics from Communication Networks to Predict Change Dependencies," in *Collaboration and Technology - 20th International Conference, CRIWG 2014, Santiago, Chile, September 7-10, 2014. Proceedings*, 2014, pp. 294–310.

[9] C. M.E., "How do Committees Invent?," *Datamation*, 1968.

[10] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," *SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, 1993.

[11] T. Zimmermann, A. Zeller, P. Weissgerber, and S. Diehl, "Mining version histories to guide software changes," *IEEE Transactions on soft Engineering*, vol. 31, no. 6, pp. 429–445, 2005.

[12] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*, vol. 8. Cambridge university press, 1994.

[13] C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu, "Putting It All Together: Using Socio-technical Networks to Predict Failures," in *Proceedings of the 2009 20th International Symposium on Software Reliability Engineering*, 2009, pp. 109–119.

[14] T. Wolf, A. Schroter, D. Damian, and T. Nguyen, "Predicting build failures using social network analysis on developer communication," in *Proceedings of the 31st International Conference on Software Engineering*, 2009, pp. 1–11.

[15] S. Biçer, A. B. Bener, and B. Çaglayan, "Defect prediction using social network analysis on issue repositories," in *Proceedings of the 2011 International Conference on Software and Systems Process*, 2011, pp. 63–71.

[16] A. Meneely, L. Williams, W. Snipes, and J. Osborne, "Predicting Failures with Developer Networks and Social Network Analysis," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2008, pp. 13–23.

[17] D. E. Farrar and R. R. Glauber, "Multicollinearity in Regression Analysis: The Problem Revisited," *The Review of Economics and Statistics*, vol. 49, no. 1, pp. 92–107, 1967.

[18] H. Akaike, "A new look at the statistical model identification," *Automatic Control, IEEE Transactions on*, vol. 19, no. 6, pp. 716–723, Dec. 1974.

[19] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, vol. Wiley Series in Probability and Statistics - 3 edition. 2013.

[20] N. Bettenburg and A. E. Hassan, "Studying the impact of social interactions on software quality," *Empirical Software Engineering*, vol. 18, no. 2, pp. 375–431, 2013.

[21] G. E. P. Box and D. R. Cox, "An Analysis of Transformations," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 26, no. 2, pp. 211–252, 1964.

[22] P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos, "Graph-based analysis and prediction for software evolution," in *Software Engineering (ICSE), 2012 34th International Conference on*, 2012, pp. 419–429.

[23] N. Bettenburg and A. E. Hassan, "Studying the impact of social interactions on software quality," *Empirical Softw. Engg.*, vol. 18, no. 2, pp. 375–431, Apr. 2013.

[24] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, "Categorizing Bugs with Social Networks: A Case Study on Four Open Source Software Communities," in *Proceedings of the 2013 International Conference on Software Engineering*, 2013, pp. 1032–1041.

[25] A. Meneely and L. Williams, "Secure Open Source Collaboration: An Empirical Study of Linus' Law," in *16th ACM Conference on Computer and Communications Security*, 2009.

[26] S. Bicer, A. B. Bener, and B. Cauglayan, "Defect prediction using social network analysis on issue repositories," in *Proceedings of the 2011 International Conference on Software and Systems Process*, 2011, pp. 63–71.

[27] E. Kouroshfar, "Studying the effect of co-change dispersion on software quality," in *Proceedings of the 2013 International Conference on Software Engineering*, 2013, pp. 1450–1452.

[28] M. D'Ambros, M. Lanza, and R. Robbes, "On the Relationship Between Change Coupling and Software Defects," in *WCRE '09*, 2009, pp. 135–144.

[29] Y. Zhou, M. Wursch, E. Giger, H. Gall, and J. Lu, "A Bayesian Network Based Approach for Change Coupling Prediction," in *WCRE '08. 15th Working Conference on*, 2008, pp. 27–36.

[30] G. Canfora, M. Ceccarelli, L. Cerulo, and M. D. Penta, "Using multivariate time series and association rules to detect logical change coupling: An empirical study," in *26th IEEE International Conference on Software Maintenance, ICSM 2010*, 2010, pp. 1–10.

[31] M. Ceccarelli, L. Cerulo, G. Canfora, and M. D. Penta, "An eclectic approach for change impact analysis," in *Proceedings of the 32nd International Conference on Software Engineering - Volume 2, ICSE 2010*, 2010, pp. 163–166.

[32] K. Herzig and A. Zeller, "The Impact of Tangled Code Changes," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, 2013, pp. 121–130.

[33] I. S. Wiese, F. R. Côgo, R. Ré, I. Steinmacher, and M. A. Gerosa, "Social metrics included in prediction models on software engineering: a mapping study," in *The 10th International Conference on Predictive Models in Software Engineering, PROMISE '14, Torino, Italy, September 17, 2014*, 2014, pp. 72–81.


**Igor Scaliante Wiese** é doutorando pela Universidade de São Paulo (IME-USP) sob supervisão do professor Marco Gerosa. É professor do Departamento de Computação da Universidade Tecnológica Federal do Paraná - campus Campo Mourão. Cursou doutorado-sanduíche na University of California - Irvine sob a orientação do professor David Redmiles. Atua na área de Engenharia de Software, em especial os temas de mineração de repositórios de software, métricas de software e dependências de software.

**Rodrigo Takashi Kuroda** é mestrando pela Universidade Tecnológica Federal do Paraná (UTFPR) – Campus Cornélio Procópio, orientado pelo professor Reginaldo Ré. Pesquisa Engenharia de Software com foco em mineração de repositórios de software, métricas de software e dependências de software. Graduado em Tecnologia em Sistemas para Internet pela Universidade Tecnológica Federal do Paraná (UTFPR) – Campus Campo Mourão. Atua profissionalmente na Atos como Analista e Desenvolvedor de sistemas em linguagem Java.

**Reginaldo Ré** é, desde de agosto de 2007, professor e pesquisador da Universidade Tecnológica Federal do Paraná, tem mestrado (2002) e doutorado (2009) em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação. Tem experiência na área de Ciência da Computação, com ênfase em Engenharia de Software, atuando principalmente nos seguintes temas: teste de software, sistemas de informação baseados na web, frameworks OO, programação orientada a aspectos e mineração de repositórios de software.

**Rodrigo Bulhões** é professor assistente do Departamento de Estatística da Universidade Federal da Bahia (UFBA). Possui graduação em Estatística pela UFBA e mestrado em Estatística pela Universidade de São Paulo. Foi aluno de intercâmbio acadêmico da Universidade de Coimbra, Portugal. Atua na área de Probabilidade e Estatística, com ênfase em análise de equações estruturais, modelos lineares generalizados e estatística aplicada.

**Gustavo Ansaldi Oliva** é doutorando pela Universidade de São Paulo (IME-USP) sob supervisão do professor Marco Gerosa. Sua principal pesquisa é em Engenharia de Software, com foco na recuperação, análise e visualização de dependências de software. Gustavo já recebeu bolsas de estudo da HP Brasil e da Comissão Europeia. Na indústria, Gustavo trabalhou na IBM Brasil como consultor e desenvolvedor de software por mais de 3 anos. Recentemente, cursou doutorado-sanduíche na Queen's University sob a supervisão do professor Ahmed Hassan.

**Marco Aurélio Gerosa** é professor associado no Departamento de Ciência da Computação da Universidade de São Paulo. Sua pesquisa está na interseção entre Engenharia de Software e Sistemas Colaborativos, focando em engenharia de software experimental, mineração de repositórios de software, evolução de software e dimensões sociais do desenvolvimento de software. Recebe bolsa de produtividade do CNPq e coordena projetos de software livre que já receberam diversos prêmios. Para mais informações, visite http://www.ime.usp.br/~gerosa.