UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

RODRIGO TAKASHI KURODA

UMA FERRAMENTA PARA PREDIÇÃO DE MUDANÇAS CONJUNTAS BASEADAS EM INFORMAÇÕES DE REPOSITÓRIOS DE SOFTWARE

DISSERTAÇÃO

CORNÉLIO PROCÓPIO

2017

RODRIGO TAKASHI KURODA

UMA FERRAMENTA PARA PREDIÇÃO DE MUDANÇAS CONJUNTAS BASEADAS EM INFORMAÇÕES DE REPOSITÓRIOS DE SOFTWARE

Dissertação apresentada ao Programa de Pós-graduação em Informática da Universidade Tecnológica Federal do Paraná – UTFPR como requisito parcial para a obtenção do título de "Mestre em Informática".

Orientador: Prof. Dr. Reginaldo Ré

CORNÉLIO PROCÓPIO

2017

Dados Internacionais de Catalogação na Publicação

K96 Kuroda, Rodrigo Takashi

Uma ferramenta para predição de mudanças conjuntas baseadas em informações de repositórios de software / Rodrigo Takashi Kuroda.
130 f. : il. color. ; 31 cm

Orientador: Reginaldo Ré.
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. Cornélio Procópio, 2017.
Bibliografia: p. 101-110.

1. Adivinhação. 2. Software - Manutenção. 3. Classificação. 4. Informática – Dissertações. I. Ré, Reginaldo, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD (22. ed.) 004

Biblioteca da UTFPR - Câmpus Cornélio Procópio

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Programa de Pós-Graduação em Informática

Título da Dissertação Nº 35:

"UMA FERRAMENTA PARA PREDIÇÃO DE MUDANÇAS CONJUNTAS BASEADAS EM INFORMAÇÕES DE REPOSITÓRIOS DE SOFTWARE".

por

RODRIGO TAKASHI KURODA

Orientador: **Prof. Dr. Reginaldo Ré**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM INFORMÁTICA – Área de Concentração: Computação Aplicada, pelo Programa de Pós-Graduação em Informática – PPGI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Cornélio Procópio, às 14h do dia 01 de agosto de 2017. O trabalho foi _____ pela Banca Examinadora, composta pelos professores:

_____
Prof. Dr. Willian Massami Watanabe
(Presidente – UTFPR-CP)

_____
Profa. Dra. Thelma Elita Colanzi
(UEM-PR)

_____
Prof. Dr. Igor Scaliante Wiese
(UTFPR-CM)

Visto da coordenação: _____
André Takeshi Endo
Coordenador do Programa de Pós-Graduação em Informática
UTFPR Câmpus Cornélio Procópio

A Folha de Aprovação assinada encontra-se na Coordenação do Programa.

AGRADECIMENTOS

Primeiramente, deixo o meu mais sincero agradecimento à minha família, em especial aos meus pais pelo apoio e carinho durante toda a minha vida. Durante essa jornada, enfrentei dificuldades pelas quais poderia não ter superado sem o imenso suporte que eles me deram. Sem vocês, com certeza teria sido muito mais difícil ou ficaria impossibilitado de completar mais essa sonhada jornada. Agradeço também aos meus irmãos, que fizeram parte e, de certa forma, também me apoiaram nessa jornada. Agradeço aos membros do Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná Campus Cornélio Procópio, principalmente aos professores, os quais foram essenciais nessa minha jornada. Graças ao nobre trabalho de ensinar e ao conhecimento de vocês, pude aprender muito durante essa jornada e que, de certa forma, agregou mais valor a esse trabalho. Deixo minha admiração por todos, pois o trabalho de vocês é de grande inspiração. Meu agradecimento especial ao meu orientador, professor Dr. Reginaldo Ré, por ter dado essa oportunidade como orientando. Esse trabalho é resultado do seu trabalho e dedicação, pelo qual pude aprender muito. Agradeço também ao professor Dr. Igor Wiese, com quem eu tive a grande privilégio de aprender, colaborar e pesquisar. Acredito que grande parte desse trabalho se deve à essa oportunidade. Agradeço ao professor Dr. Igor Steinmacher, que também colaborou com esse trabalho. Agradeço também aos demais autores de estudos realizados ao longo dessa jornada pela oportunidade em trabalhar em colaboração com essas contribuições para comunidade científica. Sou grato também aos meus amigos e colegas, alguns dos quais me ouviram muito falar dessa minha jornada chamada mestrado, mas que sempre me apoiaram e incentivaram. Deixo o agradecimento especial também para aqueles que me ajudaram e contribuíram de qualquer forma com esse trabalho, seja direta ou indiretamente. A todos, deixo registrado meu agradecimento. Obrigado!

RESUMO

KURODA, Rodrigo T.. **UMA FERRAMENTA PARA PREDIÇÃO DE MUDANÇAS CONJUNTAS BASEADAS EM INFORMAÇÕES DE REPOSITÓRIOS DE SOFTWARE** . 131 f. Dissertação – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2017.

A manutenção é uma fase do ciclo de vida do software reconhecida por demandar uma grande quantidade de esforço em comparação às outras, como o desenvolvimento. As tarefas da manutenção envolvem a modificação do software, mais especificamente os artefatos que o compõe. O fato de modificar um determinado artefato pode afetar outras partes do software, cujo fenômeno é conhecido como impacto de mudança. Técnicas e ferramentas para apoiar a análise de impacto de mudança geralmente são baseadas em tipos de acoplamentos e têm sido propostas por diversos trabalhos na literatura, como as Regras de Associação e Aprendizado de Máquina usando Classificação. No entanto, essas técnicas não foram avaliadas do ponto de vista prático. Motivado pela falta de uma avaliação dessa perspectiva, esse trabalho realizou um experimento com uma ferramenta que implementa tais técnicas. Para tanto, foi desenvolvida uma ferramenta para automatizar a execução dessas técnicas para realizar predição de mudanças conjuntas de artefatos e apresentá-las ao desenvolvedor, além de coletar o *feedback* dos desenvolvedores na ferramenta. Com a ferramenta desenvolvida, foi realizada uma prova de conceito com o uso da ferramenta em tarefas de manutenção (defeitos) do projeto de software livre da Apache denominado CXF, por colaboradores novatos representados pelos alunos do curso de Ciência da Computação. Apesar de depender do desempenho das técnicas de predição de mudanças conjuntas, os resultados mostraram evidências que a ferramenta pode apoiar colaboradores novatos e, também, diminuir o esforço para realizar uma tarefa de manutenção de software, comparando quando nenhuma técnica é usada.

Palavras-chave: Ferramenta de predição, mudanças conjuntas, acoplamento de mudança, informações contextuais, classificação, regras de associação, manutenção de software, colaboradores novatos

ABSTRACT

KURODA, Rodrigo T.. **A TOOL FOR PREDICTING JOINT CHANGES BASED ON INFORMATION FROM SOFTWARE REPOSITORIES** . 131 f. Dissertação – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2017.

Maintenance is a stage of the software life cycle recognized by demanding a lot of effort in comparison to others, such as development. The maintenance tasks involve software modification, specifically the artifacts that compose it. Modifying a particular artifact can affect other parts of the software. This phenomenon is known as change impact. Techniques and tools to support the change impact analysis are usually based on types of couplings and have been proposed by several studies in the literature, such as the Association Rules and Machine Learning using Classification. However, these techniques have not been evaluated from the practical point of view. Therefore, a tool was developed to automate the execution of these techniques to predict joint changes of artifacts and present them to the developer, as well as collecting developers' feedback on the tool. With the developed tool, a proof of concept was performed using the tool in maintenance tasks (bugs) of the Apache free software project called CXF, by novice collaborators represented by the students of the Computer Science course. Although it depends on the performance of joint change prediction techniques, the results showed evidence that the tool can support novice collaborators and also decrease the effort to perform a software maintenance task, comparing when no technique is used.

Keywords: Prediction tool, joint changes, change coupling, contextual information, classification, association rules, software maintenance, novice collaborators

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A manutenção de software tem sido reconhecida como uma importante etapa do desenvolvimento de software (BENNETT; RAJLICH, 2000; MENS et al., 2005; GODFREY; GERMAN, 2008; DURDIK et al., 2012; RAJLICH, 2014). Isso é causado principalmente pelo fato de que as tarefas associadas a essa etapa consomem muitos recursos, especialmente em função da intensa participação dos desenvolvedores para modificar e adaptar o software às necessidades dos usuários (SCHNEIDEWIND, 1987; PRESSMAN, 2005; IEEE, 2006).

Durante as tarefas de manutenção de software, os artefatos podem mudar por diferentes razões, tais como o desenvolvimento de uma nova funcionalidade, a correção de defeitos e a refatoração do código (CANFORA et al., 2014). A correta execução dessas atividades, advindas de solicitações de mudança[1] que tem por objetivo a evolução do software, envolve o esforço dos desenvolvedores para identificar quais artefatos devem ser modificados conjuntamente a fim de manter a integridade do software. Essa atividade é conhecida como Análise do Impacto de Mudança (AIM), que utiliza um conjunto de técnicas para determinar os efeitos de uma mudança em outras partes do software (BOHNER; ARNOLD, 1996; GETHERS et al., 2012).

Nesse sentido, a AIM é importante, do ponto de vista gerencial, para os gerentes estimarem custo e esforço necessários para realizar mudanças e, do ponto de vista técnico, para os desenvolvedores identificarem os impactos e propagarem as mudanças no software (ALMASRI et al., 2016). Portanto, identificar e predizer artefatos que mudam conjuntamente durante a realização de tarefas pode auxiliar os desenvolvedores a completarem a mudança sem inserir novos defeitos, ou completar a mudança de forma mais rápida (HASSAN; HOLT, 2004).

Na última década, pesquisadores têm proposto técnicas para apoiar os desenvolvedores na AIM, predizendo artefatos que são propensos a mudar conjuntamente a

[1] A solicitação de mudança (do inglês *Modification/Change Request*) é um termo genérico utilizado para identificar modificações propostas para um software que está recebendo manutenção (IEEE, 2006).

fim de completar uma solicitação de mudança (ORSO et al., 2004; GETHERS et al., 2012; KAGDI et al., 2013; HASSAN; HOLT, 2004; ZIMMERMANN et al., 2005; WIESE et al., 2015c). Normalmente, as técnicas de AIM utilizam a frequência de mudança conjunta no passado, denominado de acoplamento de mudança (BALL et al., 1997; YING et al., 2004; ZIMMERMANN et al., 2005). Acoplamentos de mudança têm sido identificados por meio de técnicas que avaliam alguns tipos de acoplamento entre os artefatos, tais como o dinâmico (ORSO et al., 2004), o estático (BRIAND et al., 1999b) e o semântico (GETHERS et al., 2012; KAGDI et al., 2013).

Apesar da quantidade de técnicas propostas e dos resultados obtidos, não existem avaliações destas técnicas diretamente e qualitativamente com desenvolvedores, o que pode dificultar a adoção das técnicas na prática, ou ainda indicar a não adoção dessas técnicas. Dentre as possíveis razões, encontram-se a baixa acurácia na predição que gera possíveis falsos alertas aos desenvolvedores (CANFORA et al., 2010; LI et al., 2013; SUN et al., 2015) e a necessidade de configuração de parâmetros para o uso de algum tipo de acoplamento, já que é difícil adotar valores de referências para os tipos de acoplamentos em projetos diferentes (ZIMMERMANN et al., 2005).

Recentemente, o grupo de pesquisa, do qual o autor desse trabalho também faz parte, propôs a predição de mudanças conjuntas com a utilização de informações contextuais obtidas a partir das solicitações de mudanças, da comunicação entre os colaboradores, e das mudanças dos próprios artefatos (WIESE et al., 2015c, 2017). Estudos indicaram que essas informações podem conter padrões de propensão de mudança conjunta entre artefatos (WIESE et al., 2014, 2015a, 2015c, 2017). Nesses estudos realizados, foram encontradas evidências de que o uso de algoritmos de aprendizagem de máquina, usando as informações contextuais extraídas dos repositórios de software, podem reduzir os falsos alertas das predições de mudanças conjuntas, sem a necessidade de configurar parâmetros para determinar se um acoplamento será utilizado ou não.

Nesse sentido, é interessante ter uma ferramenta que utilize as técnicas de predição de mudanças baseado em informações de repositórios de software, pois estudos mostraram uma redução do número de falsos alertas, utilizando aprendizado de máquina (WIESE et al., 2015c, 2017), e maior número de acertos na predição, utilizando a ferramenta ROSE (SUN et al., 2015). Hipotetiza-se que pode haver uma aceitação positiva da ferramenta com essas técnicas no cenário prático, uma vez que essas técnicas reduzem a quantidade de predições incorretas.

Embora os estudos tenham avaliado as técnicas ou ferramentas de forma empí-

rica, como Zimmermann et al. (2005), Sun et al. (2015) e Wiese et al. (2015c), nenhum deles realizou uma avaliação com potenciais usuários em um cenário prático. Dessa maneira, nenhum desses estudos capturou aspectos qualitativos relacionados a intenção do desenvolvedor em realizar a mudança. O experimento empírico com os usuários é importante para validar a ferramenta ou técnica e verificar sua aceitação. Com o *feedback* dos usuários, pode-se identificar pontos positivos e negativos da ferramenta e, assim, futuramente melhorar para facilitar a sua adoção em cenários práticos.

Para avaliar a aceitação, é necessário que a ferramenta seja aplicada em um cenário prático com potenciais usuários da ferramenta. Uma das aplicações da ferramenta com técnicas de predição de mudanças conjuntas seria na manutenção de software livre. Em projetos de software livre, um dos maiores desafios é prover suporte aos colaboradores novatos, pois eles enfrentam diversas dificuldades em colaborar (STEINMACHER, 2015). Essa ferramenta apoiaria os colaboradores na análise de impacto na manutenção do software livre, apresentando possíveis artefatos que mudariam para completar tarefas de manutenção de software. Hipotetiza-se que uma ferramenta de predição de mudanças conjuntas possa ajudar colaboradores novatos a contribuir com comunidades de software livre e aumentar a eficiência na manutenção do software.

Motivado pelo fato de não existir uma ferramenta que implemente as técnicas com menores número de falsas predições, esse trabalho propõe uma ferramenta que utilize informações de repositórios de software e a técnica de Regras de Associação e Classificação para predição de mudanças conjuntas. Para validar a ferramenta, foi conduzido um experimento no qual a ferramenta foi utilizada para predizer mudanças conjuntas em solicitações de mudanças de projetos de software livre para que colaboradores novatos pudessem contribuir.

1.1 OBJETIVO GERAL E ESPECÍFICO

O objetivo deste trabalho é desenvolver e propor uma ferramenta que utilize informações contextuais coletadas das tarefas, da comunicação e das mudanças para predizer de mudanças conjuntas entre artefatos, que foi proposta por Wiese et al. (2015c), além da técnica predição de mudanças conjuntas com acoplamento de mudança, proposta por Zimmermann et al. (2005). Para tanto, os seguintes objetivos específicos devem ser cumpridos:

- Automatizar a extração de dados do sistema de controle de versão e do sistema

gerenciador de tarefas dos projetos de software livre;

- Automatizar a predição de mudanças conjuntas baseadas em informações contextuais proposta por Wiese et al. (2015c, 2017);
- Automatizar a predição de mudanças conjuntas baseadas na frequência de mudanças conjuntas do histórico do projeto, com base em Zimmermann et al. (2005);
- Avaliar a aceitação da ferramenta por colaboradores novatos;
- Avaliar a mudança na autoeficácia dos colaboradores novatos antes e após o uso da ferramenta e das técnicas aplicadas;
- Avaliar o desempenho dos colaboradores nas tarefas com e sem o uso de técnicas de predição de mudanças conjuntas na ferramenta e das técnicas aplicadas.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está estruturado da seguinte forma: no Capítulo 2 são apresentados os trabalhos e conceitos que fundamentam esse estudo. No Capítulo 3 são apresentados os detalhes da ferramenta, expondo a metodologia que é seguida pela ferramenta. No Capítulo 4, são apresentados a configuração e o resultado do experimento empírico realizado com a ferramenta em um cenário prático de manutenção de software com usuários novatos, bem como a discussão dos resultados e as limitações desse estudo. A conclusão, as contribuições e os trabalhos futuros são apresentados no Capítulo 5. Por fim, estão listadas as referências bibliográficas que fundamentam este trabalho, além de materiais gerados nesse trabalho, encontrados no Apêndice.

2 REVISÃO DA LITERATURA

2.1 CONSIDERAÇÕES INICIAIS

Durante a fase de manutenção do software, ocorrem diversas modificações nos artefatos do software que podem afetar outros e diferentes artefatos do software. Na literatura, esse efeito é conhecido como impacto de mudança, abordado na Seção 2.2. Esse efeito acontece devido aos acoplamentos existentes entre as partes do software, detalhados na Seção 2.3. A fim de minimizar os impactos desse fenômeno, técnicas e ferramentas para análise de impacto de mudança têm sido propostas, sendo que as principais são apresentadas na Seção 2.4 e na Seção 2.5. Essas técnicas são avaliadas conforme seus desempenhos com cálculos baseados na taxa de acertos e erros, também apresentados na Seção 2.4.

2.2 ANÁLISE DE IMPACTO DE MUDANÇAS

Realizar uma mudança no software, mesmo que simples, pode não ser uma tarefa fácil, uma vez que pode ocasionar efeitos colaterais e/ou efeito cascata em artefatos, módulos e sistemas (BOHNER; ARNOLD, 1996; LI et al., 2013; KAGDI et al., 2013). Dessa forma, é importante que sejam identificados os artefatos, módulos e sistemas que são afetados pela mudança proposta, ou ainda que sejam identificadas as possíveis consequências de determinada mudança. Essa atividade é denominada Análise de Impacto de Mudança (AIM) (BOHNER; ARNOLD, 1996). Um dos objetivos diretos da AIM é detectar a propagação de mudanças. A propagação de mudança tem por objetivo garantir a consistência entre as entidades interdependentes, evitando que o software entre em um estado errôneo devido aos defeitos criados por uma mudança incompleta (HASSAN; HOLT, 2004; LI et al., 2013).

A Figura 1 mostra uma visão geral do processo de AIM (BOHNER, 2002; DE LUCIA et al., 2008; LI et al., 2013). O processo inicia com a análise da solicitação

de mudança e o conjunto de artefato para identificar o conjunto inicial de elementos que poderiam ser afetados pela mudança, denominado conjunto de impacto inicial (CII) (BOHNER, 2002; DE LUCIA et al., 2008). Em seguida, por meio da técnica de análise de impacto de mudança, são estimados outros elementos que provavelmente são afetados pelos elementos no conjunto de mudança (LI et al., 2013). O conjunto resultante é chamado de conjunto de impacto estimado (CIE). Ao completar determinada solicitação de mudança, todos os artefatos que foram modificados compõem o conjunto de impacto real (CIR). Na prática, o CIR nem sempre é único, uma vez que uma mudança pode ser implementada de muitas maneiras (BOHNER, 2002).

A AIM no software é um processo iterativo, como observado na Figura 1. Durante ou após uma mudança, alguns elementos impactados que não estão no CIE podem ser descobertos (elementos subestimados), que fazem parte do conjunto de impacto falso negativo (CIFN) (LI et al., 2013). Por outro lado, o CIE geralmente inclui alguns elementos que não precisam ser modificados, cujos elementos formam o conjunto de impacto falso positivo (CIFP). Removendo o CIFP da união entre o CIE com CIFN deve resultar no CIR ($(CIE \cup CIFN) \notin CIFP = CIR$) (LI et al., 2013). Neste sentido, o objetivo comum das técnicas de AIM é estimar os elementos mais próximos daqueles realmente impactados pela mudança inicial (CIE mais próximo do CIR) (LI et al., 2013).

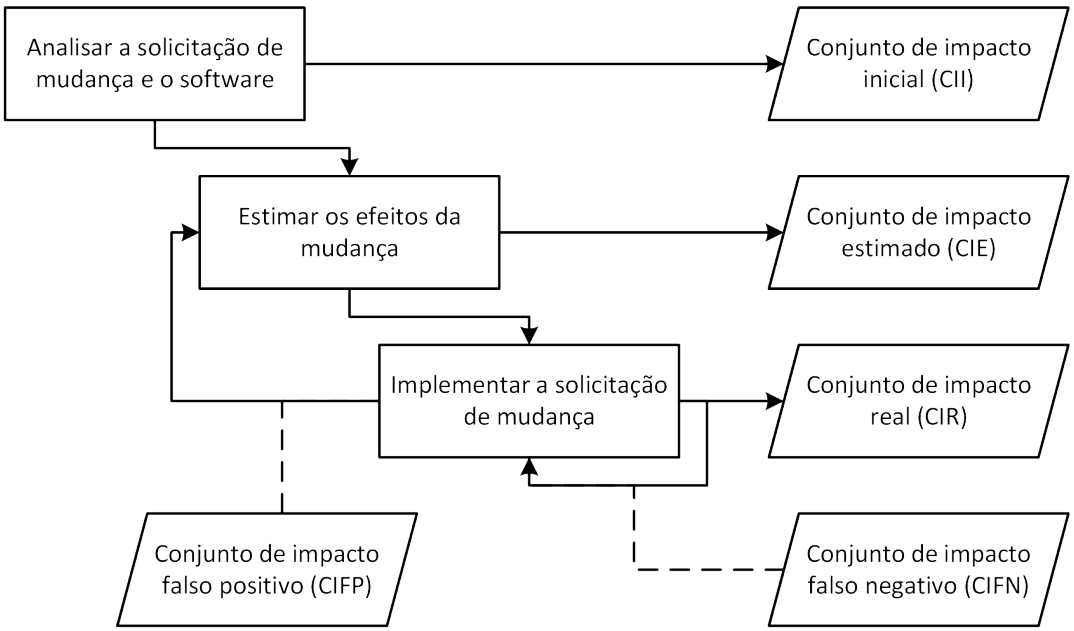


Figura 1: Processo de análise de impacto de mudança.
Fonte: Adaptado de Bohner (2002).

Dessa forma, pesquisadores têm proposto diferentes técnicas de predição de

mudanças, principalmente para apoiar os desenvolvedores enquanto eles modificam os artefatos (ORSO et al., 2004; GETHERS et al., 2012; KAGDI et al., 2013; HASSAN; HOLT, 2004; ZIMMERMANN et al., 2005; WIESE et al., 2015c). Algumas técnicas de AIM são baseadas na análise da rastreabilidade (AIM baseadas em rastreabilidade), enquanto outras se concentram nos relacionamentos de dependência (AIM baseada em dependência) (KAGDI et al., 2013; LI et al., 2013). As técnicas baseadas na rastreabilidade consistem em examinar as dependências entre entidades de diferentes níveis de abstração, tais como documento de requisitos, documento de projeto, código-fonte e casos de teste (DE LUCIA et al., 2008; LI et al., 2013). Por outro lado, as técnicas de AIM baseadas em relacionamentos de dependência (acoplamento) desempenham a AIM em artefatos no mesmo nível de abstração, por exemplo, somente entre artefatos de código-fonte (LI et al., 2013).

A AIM no nível de requisitos e projeto apoia-se em modelos abstratos de alto nível, como os diagramas de UML (do inglês *Unified Modeling Language*) e mapas de casos de uso. Entretanto, as técnicas baseadas em dependência analisam os artefatos, em especial os códigos-fonte, geralmente considerados os mais importantes na AIM (DE LUCIA et al., 2008). Algumas técnicas baseadas em dependência que podem ser encontradas na literatura são o fatiamento (WEISER, 1981; GALLAGHER; LYLE, 1991; TONELLA, 2003), grafos de dependências (CHEN; RAJICH, 2001; BADRI et al., 2005; HATTORI et al., 2008; SUN et al., 2010), análise de dependência oculta (*hidden dependency*) (RAJLICH, 1997; YU; RAJLICH, 2001), análise dinâmica (LAW; ROTHERMEL, 2003; ORSO et al., 2004; REN et al., 2004), recuperação de informações (RI) (POSHYVANYK et al., 2009; GETHERS et al., 2012; KAGDI et al., 2013) e mineração de repositórios de software (MRS) (ZIMMERMANN et al., 2005; KAGDI et al., 2013; WIESE et al., 2014, 2015c).

As métricas de acoplamento também foram exploradas para AIM em software orientados a objetos (BRIAND et al., 1999b; POSHYVANYK et al., 2009). Por exemplo, Briand et al. (1999b) utilizaram as métricas de acoplamentos de linguagens orientadas a objetos levantados por outro estudo de Briand et al. (1999a), e encontraram evidências de que as métricas de acoplamento podem ajudar na identificação de classes propensas a mudar quando outra classe muda. Arisholm et al. (2004) utilizaram métricas de acoplamento dinâmico (abordadas na Subseção 2.3.1) para AIM e encontraram evidências que as métricas de acoplamentos são indicadores significantes de propensão a mudança, complementares às métricas de acoplamento estático (abordadas na Subseção 2.3.1).

Conforme abordado, os estudos evidenciaram que os acoplamentos nos softwares são indicadores de propensão a mudanças. Portanto, as técnicas de AIM geralmente são baseadas em um ou mais tipos de acoplamentos.

2.3 ACOPLAMENTOS EM SOFTWARE

O software é composto por diversos artefatos que colaboram entre si para executar corretamente uma tarefa especificada. Deste ponto de vista, a colaboração leva a existir um certo grau de interdependência entre os artefatos do software, denominado acoplamento (WAND; WEBER, 1990; IEEE, 2010) e conhecido também como dependência (LARMAN, 2004).

O conceito de dependência de software foi definido por Parnas (1979) da seguinte forma: um programa *A* depende de um programa *B* se a execução correta de *B* é necessária para completar a tarefa descrita nas especificações do programa *A*. Mais tarde, Lakos (1996) definiu o conceito de dependência de software adicionando a ideia do relacionamento entre dois componentes de software, que ficou conhecida como a definição de dependência estrutural sintática. Por exemplo, se o componente *A* não pode ser compilado sem o componente *B*, então o componente *A* depende "fisicamente" do componente *B*.

Na UML (*Unified Modeling Language*), a relação de dependência significa que um elemento "cliente" é semântica ou estruturalmente dependente da definição do elemento "fornecedor". Larman (2004) definiu a dependência como um artefato de software "cliente" que possui conhecimento do artefato "fornecedor" e, portanto, uma mudança no "fornecedor" pode afetar o artefato "cliente". De tal maneira, a modificação em um determinado artefato pode comprometer a integridade de outros devido a certos tipos de acoplamento entre eles.

Assim, quando uma modificação é realizada em um determinado artefato, pode ser necessária a propagação dessa mudança aos outros artefatos para manter a integridade do software. Para identificar e predizer os artefatos impactados pela mudança, as técnicas de AIM geralmente baseiam-se no conceito de acoplamento de software. A seguir são apresentados alguns dos principais tipos de acoplamentos usados por essas técnicas.

2.3.1 Acoplamento estrutural

O acoplamento estrutural, ou acoplamento sintático, indica uma relação explícita entre dois artefatos de código-fonte. Por exemplo, em linguagens de programação orientadas a objetos, se uma classe A tem uma propriedade do tipo B, então A está acoplado estruturalmente com B, conforme ilustra o diagrama de classes da UML na Figura 2. Assim, uma análise estática do código-fonte pode identificar esse tipo de acoplamento.

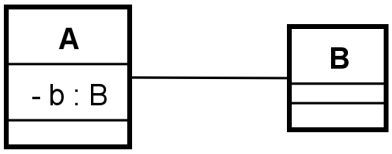


Figura 2: Exemplo de acoplamento estrutural entre as classes A e B.
Fonte: Autoria própria.

Entretanto, em linguagens de programação orientadas a objetos existem certos tipos de acoplamentos que a análise estática não consegue identificar devido, por exemplo, ao polimorfismo, à herança e à vinculação dinâmica (do inglês *dynamic binding*) (ARISHOLM et al., 2004; HASSOUN et al., 2004). Esse tipo de acoplamento é identificado com a execução do código-fonte, conhecido como acoplamento dinâmico. A execução do código-fonte permite coletar, por exemplo, informação de rastreio de execução, informação de cobertura e informação de relação de execução que podem ser usadas para identificar o acoplamento dinâmico (LAW; ROTHERMEL, 2003; ARISHOLM et al., 2004; SUN et al., 2010).

Considere o exemplo apresentado na Figura 3, que apresenta um diagrama de sequência e um diagrama de classe da UML, respectivamente. Neste exemplo, existem duas classes, A e B, que herdam das respectivas classes A' e B', de maneira que A' implementa o método mA' e B' implementa o método mB'. Se a e b são instâncias de A e B, respectivamente, e o objeto a envia uma mensagem mB' para o objeto b, a mensagem pode ter sido enviada por meio do método mA', implementado na classe A', e processado pelo método alvo mB' implementado pela classe B'. Devido à herança, a e b são objetos de classes diferentes de onde estão implementados os métodos mA' e mB'. Embora o código possa mostrar claramente o objeto do tipo A invoca, a partir do método mA', o método mB' de um objeto do tipo B, é impreciso assumir o acoplamento no nível de classe entre A e B (ARISHOLM et al., 2004). Nesse exemplo, existem dois

tipos de acoplamentos: 1) acoplamento em nível de objetos entre as classes A e B e 2) acoplamento em nível de classe entre A' e B' (ARISHOLM et al., 2004).

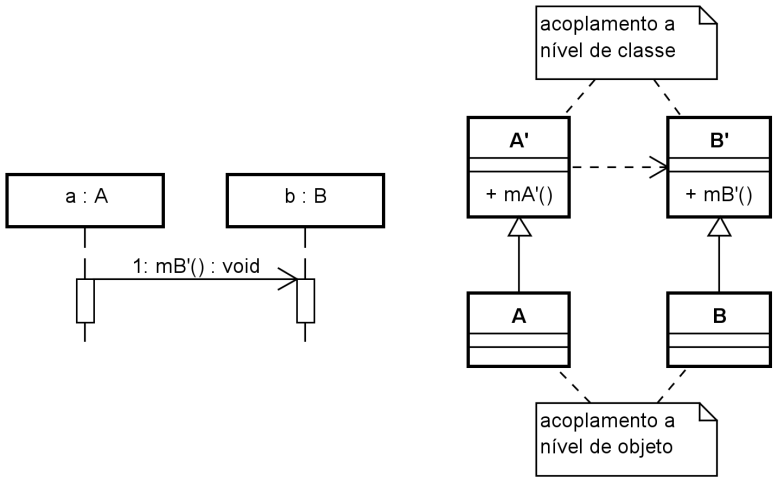


Figura 3: Exemplo das diferentes formas de acoplamento dinâmico entre as classes A, A', B e B'.

Fonte: Adaptado de Arisholm et al. (2004).

2.3.2 Acoplamento semântico

Outro fator que pode caracterizar um acoplamento entre artefatos são os conceitos contidos nos artefatos. Esse tipo de acoplamento, denominado acoplamento semântico ou acoplamento conceitual, indica uma relação semântica e, portanto, implícita entre dois ou mais artefatos. Essa relação implícita indica o grau em que os artefatos do software estão relacionados de acordo com os conceitos do domínio do software.

Para identificar o acoplamento semântico, técnicas de Recuperação de Informações (RI) podem ser utilizadas para analisar textualmente os artefatos de código-fonte (POSHYVANYK; MARCUS, 2006; GETHERS et al., 2012; KAGDI et al., 2013). O acoplamento semântico é identificado entre os artefatos de código-fonte, levando em consideração os identificadores como o nome da classe e o nome da propriedade, além dos comentários (POSHYVANYK; MARCUS, 2006). Basicamente, a similaridade entre dois documentos é calculada como o cosseno entre dois vetores (GETHERS et al., 2012; KAGDI et al., 2013), onde cada vetor representa os identificadores e comentários de uma determinada classe ou método, por exemplo. A partir dos vetores, a similaridade é calculada.

Na literatura, a técnica mais frequentemente usada é a Indexação Semântica

Latente (do inglês *Latent Semantic Indexing* ou LSI) (POSHYVANYK; MARCUS, 2006; KAGDI et al., 2013). O resultado do cálculo da similaridade pode assumir valores que variam de 0 (zero) a 1 (um), onde o valor 1 (um) significa que ambos os métodos ou classes são iguais e 0 (zero) indica que não existe semelhança semântica.

2.3.3 Acoplamento de mudança

O acoplamento de mudança é um padrão observado no qual os artefatos mudam frequentemente juntos. Desse modo, as mudanças conjuntas que são mais frequentes representam uma conexão evolucionária ou histórica entre esses artefatos. Logo, se frequentemente artefatos mudaram em conjunto no histórico de um projeto, pode-se inferir a existência de um acoplamento de mudança. O acoplamento de mudança, também conhecido como acoplamento lógico ou acoplamento evolucionário (GALL et al., 1998; ZIMMERMANN et al., 2005; D'AMBROS et al., 2009; OLIVA et al., 2011; KAGDI et al., 2013) é inferido a partir da análise do histórico das mudanças dos artefatos capturados do sistema de controle de versão. O acoplamento de mudança é importante pois, apesar de acoplamentos estruturais e semânticos explicarem certas mudanças conjuntas, existem mudanças conjuntas que não são capturadas por meio desses acoplamentos (OLIVA et al., 2011; OLIVA; GEROSA, 2015).

Para identificar tal acoplamento, são utilizadas técnicas de mineração de repositórios de software, como as Regras de Associação (YING et al., 2004; ZIMMERMANN et al., 2005). Por exemplo, considere um cenário com os artefatos *A* e *B*, no qual *A* foi modificado nas solicitações de mudança #3, #10, #19, #42, #92 e #314, e *B* nas solicitações de mudanças #3, #19, #42 e #314. A intersecção entre as solicitações de mudanças em que o artefato *A* e o artefato *B* mudaram representa as mudanças conjuntas de *A* e *B*. A Figura 4 ilustra esse cenário fictício de acoplamento de mudança entre os artefatos *A* e *B* de um software.

Duas medidas são usadas para inferir o acoplamento de mudança. Uma delas, conhecida como suporte (do inglês *support*), mede a frequência de mudanças, como o número de *commits*[1] ou o número de solicitações de mudanças em que os artefatos mudaram conjuntamente. Deste modo, a medida de suporte indica quão evidente, recorrente, uma mudança conjunta é no histórico do projeto. Levando em consideração o exemplo da Figura 4, o acoplamento de mudança entre *A* e *B* tem um valor de suporte

[1] *Commit* significa integrar as modificações realizadas pelo desenvolvedor ao repositório do sistema de controle de versão (IEEE, 2010).

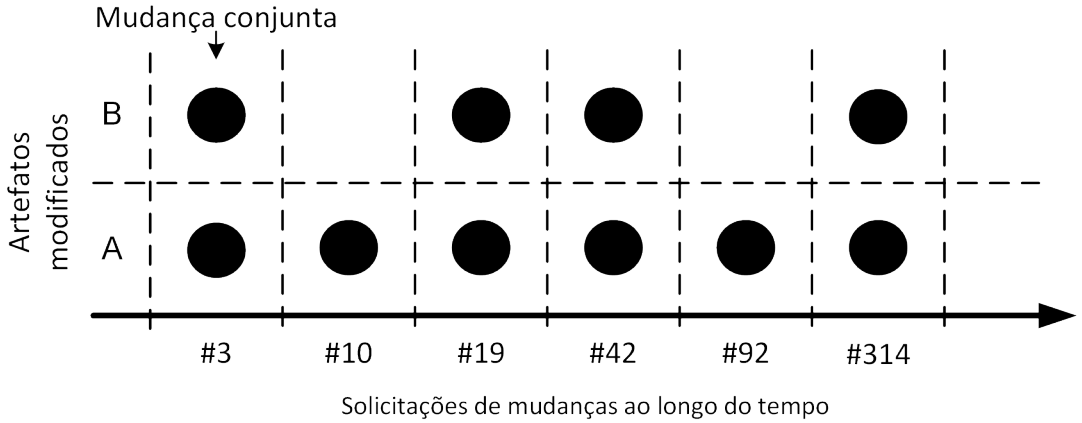


Figura 4: Exemplo de ocorrências de mudança conjunta entre os artefatos A e B em diferentes solicitações de mudança.

Fonte: Autoria própria.

igual a 4 (quatro), pois aconteceram nas solicitações de mudanças #3, #19, #42 e #314.

A outra medida é denominada confiança (do inglês *confidence*), utilizada para medir a força da regra (ZIMMERMANN et al., 2005; WIESE et al., 2015c). Seu valor é resultado da fração de *commits* contendo A que também contém B (D'AMBROS et al., 2009). No exemplo da Figura 4, de um total de 6 (seis) *commits*, 4 (quatro) são mudanças conjuntas, ou seja, *commits* contendo A e B. O valor resultante da confiança nesse exemplo é de $4 \div 6 = 0,67$. Uma regra de associação interessante tem altos valores de suporte e confiança.

Frequentemente, pesquisadores têm utilizado Regras de Associação para obter as medidas de suporte e confiança (ZIMMERMANN et al., 2005; WIESE et al., 2015c; KAGDI et al., 2013; GETHERS et al., 2012). Uma regra de associação é uma implicação na forma $I \Rightarrow J$, onde I e J são dois conjuntos disjuntos de artefatos $\mathscr{I}$ (AGRAWAL et al., 1993; ZIMMERMANN et al., 2005). Uma regra interessante $I \Rightarrow J$ significa que quando I é modificado, J é propenso a mudar também. Assim, J é uma mudança conjunta de I. Isso significa que transações que contêm I são prováveis de conter J. No escopo deste estudo, a regra $I \Rightarrow J$ significa que J mudou conjuntamente com I, nos quais I e J são conjuntos unitários de artefatos a, em que $I = a_i$ e $J = a_j$ e $a_i \neq a_j$ (WIESE et al., 2015c).

2.4 MODELOS DE PREDIÇÃO DE MUDANÇAS CONJUNTAS

A fim de apoiar os desenvolvedores na atividade de AIM e minimizar o impacto na qualidade do software causado pela mudança conjunta, torna-se importante criar mecanismos para que as mudanças conjuntas possam ser detectadas antecipadamente e as mudanças sejam propagadas. Para isso, uma das principais formas que vem sendo estudada é a utilização de técnicas de mineração de repositórios de software e modelos de propagação/predição criados por meio de algoritmos de aprendizado de máquina.

Os modelos de predição são usados para descobrir fatos desconhecidos ou eventos futuros utilizando dados históricos, e podem ser construídos com técnicas de aprendizagem de máquina (GHOTRA et al., 2015). Em Engenharia de Software, os modelos de predição têm sido utilizados para compreender ou predizer fenômenos por meio de métricas de software. Por exemplo, dados dos repositórios de software foram estudados para predizer defeitos (D'AMBROS et al., 2010; BIÇER, S.; BENER, A. B.; ÇAĞLAYAN, B., 2011; GIGER et al., 2012; RAHMAN; DEVANBU, 2013) e propagação de mudanças (HASSAN; HOLT, 2004; ZIMMERMANN et al., 2005; ZHOU et al., 2008; CANFORA et al., 2010; WIESE et al., 2015c). As métricas extraídas a partir dos dados obtidos pela mineração de repositórios de software podem ser utilizadas como variáveis preditoras para construção de modelos de predição.

A predição de mudanças conjuntas parte do princípio de que, se os artefatos mudaram em conjunto no passado, então eles são propensos a mudarem no futuro. Baseado nessa ideia, pesquisadores têm construído modelos de predição para predizer as mudanças conjuntas (YING et al., 2004; ZIMMERMANN et al., 2005; WIESE et al., 2015c). Deste modo, diversas técnicas foram propostas para determinar os artefatos mais propensos a serem modificados conjuntamente, das quais se destacam a mineração de repositórios de software aplicando técnicas de identificação da propagação de mudanças e análise conceitual de artefatos (GALL et al., 1998; YING et al., 2004; ZIMMERMANN et al., 2005; CANFORA et al., 2010), análise estática do código-fonte (BRIAND et al., 1999b) e análise híbridas entre as anteriores (GETHERS et al., 2012; KAGDI et al., 2013).

Inicialmente, métricas baseadas em acoplamentos estruturais foram utilizadas para construção de modelos probabilísticos a fim de identificar as mudanças conjuntas (BRIAND et al., 1999b). No entanto, foi observado que essas métricas são insuficientes para identificar todas as mudanças conjuntas. Assim, a partir dessa limitação, pesquisa-

dores exploraram outros tipos de acoplamentos, como o semântico e o dinâmico, para identificar mudanças conjuntas em software desenvolvido com linguagens orientadas a objetos. Geralmente, essas pesquisas visam a melhoria da predição de mudanças, especialmente da precisão e da sensibilidade que são métricas de desempenho dos modelos, abordadas mais detalhadamente na Subseção 2.4.2.

Hassan e Holt (2004) propuseram heurísticas para a propagação de mudanças em entidades que compõem o software, como funções ou variáveis. As heurísticas são caracterizadas por dois aspectos principais: (1) a sua fonte de dados, que é usada pelo algoritmo da heurística para sugerir mudanças em outras entidades, e (2) a técnica de poda (do inglês *prune*), que define o algoritmo utilizado pela heurística para reduzir o conjunto de mudanças sugeridas. De acordo com Hassan e Holt (2004), existem diversas fontes de dados que podem ser usadas para predição de mudanças em entidades, cujo objetivo é reduzir o número de artefatos sugeridos que não precisam mudar, ao mesmo tempo em que assegure que todos os artefatos que precisam mudar sejam preditos. Ou seja, o objetivo das heurísticas de fontes de dados é aumentar a precisão e sensibilidade da técnica. Algumas possíveis fontes de dados e suas heurísticas apresentadas por Hassan e Holt (2004) são:

- Dados da entidade: propõe que o processo de propagação de mudança é dependente de uma entidade particular modificada. Por exemplo, uma mudança pode propagar a outras entidades interdependentes à entidade modificada devido aos tipos de relacionamentos como o acoplamento lógico, cuja fonte de dados é o histórico de mudança conjunta; acoplamento estrutural, cuja fonte de dados é a estrutura do código-fonte; e a localização das entidades, que diz respeito a entidades do mesmo artefato em que está definido a entidade que mudou, cuja fonte de dados é definida como *layout* do código-fonte.

- Dados do desenvolvedor: essa heurística supõe que um mesmo desenvolvedor frequente ou recentemente propaga as mudanças para outras entidades, baseado no fato de que ele adquire experiência em certas áreas do código-fonte e, portanto, é propenso a modificar as mesmas entidades.

- Dados do processo: parte do princípio que a propagação de mudanças depende do processo empregado no desenvolvimento.

- Dados de nomes semelhantes: assume que as mudanças propagam a outras entidades que possuem o nome semelhante ao da entidade que mudou.

- Dados aleatórios: assume que a propagação de mudança é um processo aleatório que é caótico e imprevisível.

Hassan e Holt (2004) também sugeriram heurísticas para as técnicas de poda. Essas técnicas servem para reduzir o número de predições de entidades. As heurísticas propostas pelos autores para as técnicas de poda são:

- Frequência: as técnicas baseadas na frequência resultam em entidades mais frequentemente relacionadas sobre um limiar.
- Recência: diz respeito a entidades que foram relacionadas em um passado recente, suportado pela intuição de que os desenvolvedores tendem a focar em alguma funcionalidade durante um período de tempo específico.
- Frequência e recência: utilizar as duas técnicas de poda apresentadas anteriores de forma híbrida, resultando em entidades que frequentemente estão relacionadas em um passado recente.
- Aleatório: técnicas que usam algum limiar para selecionar as sugestões, por exemplo, a quantidade, que podem ser utilizadas quando não há dados a respeito da frequência e recência.
- Sem poda: não usar uma técnica de poda, retornando os resultados sem remover alguma sugestão.

Dentre as heurísticas propostas, Hassan e Holt (2004) estudaram empiricamente as heurísticas de fonte de dados do desenvolvedor e aquelas baseadas na entidade, que são o histórico de mudanças conjuntas, a estrutura do código e o *layout* do código. Para o estudo empírico, foram selecionados os softwares livres NetBSD, FreeBSD, OpenBSD Postgres e GCC, desenvolvidos na linguagem de programação C, em que foram analisados 40 anos de histórico do sistema de controle de versão dos respectivos softwares. Os resultados mostraram que as heurísticas combinadas alcançaram uma sensibilidade média de 51% e uma precisão média de 49% e, dessa forma, podem ser utilizadas para auxiliar desenvolvedores durante o processo de propagação de mudança.

As regras de associação foram aplicadas por Zimmermann et al. (2005) para identificar padrões de modificações em artefatos analisando o histórico do sistema de

controle de versão. Para tanto, utilizaram o algoritmo denominado *Apriori*, que usa os valores de suporte e confiança como limiares para extrair regras de associação. Para avaliar essa técnica, os autores a aplicaram em oito projetos de software livre: Eclipse, GCC, GIMP, JBoss, jEdit, KOffice, Postgres e Python. Os resultados apresentaram na média uma precisão de 29% e sensibilidade 33%. As regras de associação com o algoritmo *Apriori* também foram utilizadas por Canfora et al. (2010). Entretanto, diferentemente de Zimmermann et al. (2005), os autores propõem uma abordagem híbrida usando a análise de séries temporais multivariadas e regras de associação. Para tanto, aplicaram particularmente a técnica denominada teste de causalidade de *Granger* bivariada (GRANGER, 1969; CANFORA et al., 2010). Considere como exemplo um par de artefatos a e b. Com base no histórico de mudanças de a e b, uma equação explica a evolução do artefato a e, então, um teste estatístico mostra se o histórico de mudanças de a é útil para predizer mudanças de b. O estudo empírico da técnica realizado nos projetos Mylyn, FreeBSD (i386), Rhino e Squid, desenvolvidos na linguagem C e Java, mostraram que a causalidade de *Granger* complementa as predições resultantes da técnica de regras de associação com o algoritmo *Apriori*. A combinação das duas técnicas alcançou uma média da medida-F de 25%, sendo que o valor máximo alcançado foi de 30% no projeto Rhino.

Outra abordagem híbrida, ou seja, combinando mais de uma técnica, foi proposta por Gethers et al. (2012). O estudo consistiu em combinar técnicas baseadas em acoplamentos de mudança, acoplamento dinâmico e acoplamento semântico para análise de impacto de mudança. Uma avaliação empírica foi feita aplicando a técnica nos projetos de software livre ArgoUML, jEdit, muCommander e JabRef. Nesses projetos, a combinação dos três tipos de acoplamentos apresentou uma melhora de até 17% na precisão e até 41% na sensibilidade quando comparada ao uso apenas do acoplamento semântico. O desempenho máximo alcançado foi de 18% de precisão e 75% de sensibilidade no projeto jEdit.

Seguindo na linha de abordagens híbridas, Kagdi et al. (2013) combinaram o acoplamento de mudança com o acoplamento semântico. A abordagem apresentada pelos autores foi avaliada empiricamente nos softwares de software livre Apache httpd, ArgoUML, iBatis, KOffice e jEdit. De forma geral, a avaliação mostrou uma melhora de até 21% na medida-F, alcançando até 34% de medida-F.

Outros trabalhos exploraram a predição das mudanças conjuntas utilizando técnicas de aprendizado de máquina. Zhou et al. (2008) utilizaram uma técnica de classi-

ficação denominada Redes Bayesianas para predição de acoplamentos lógicos com base nas seguintes características: nível de dependência estática do código-fonte, frequência da mudança conjunta, nível de significância da mudança, idade da mudança e o autor da mudança. Um experimento foi realizado aplicando essa técnica em 5 versões diferentes dos softwares ArgoUML e Azureus2. Os resultados mostraram que o uso dessa abordagem é promissor, apresentando valores de sensibilidade entre 60% e 80% e valores de precisão entre 40% e 60%.

Wiese et al. (2015c) propõe o uso de informações de repositórios de software, ou informações contextuais, com algoritmo de aprendizagem de máquina para predição de mudanças conjuntas. As informações contextuais estão relacionadas à fatores humanos e processuais, diferente de outros trabalhos que usam, por exemplo, informações centradas na linguagem de programação ou artefatos. As dimensões utilizadas por este trabalho foram divididas em três contextos:

- **Contexto de Solicitação de Mudança**: informações da solicitação de mudança, das quais foram usadas o tipo da solicitação de mudança, responsável e relator.
- **Contexto de Comunicação**: informações referentes à comunicação entre desenvolvedores e outros colaboradores na solicitação de mudança, das quais foram usadas o número de comentários, o número de comentadores, o número de palavras e número de desenvolvedores que comentaram.
- **Contexto de Mudança**: informações da mudança dos artefatos, das quais foram usadas o autor da modificação (*commit*), o número de linhas de código adicionadas, o número de linhas de código excluídas e a soma do número de linhas adicionadas e excluídas.

Os estudos de Wiese et al. (2015c) mostraram que é possível reduzir o número de predições incorretas com a técnica proposta. Para tanto, foi utilizada uma técnica de classificação (apresentada na Subseção 2.4.1) denominada Floresta Randômica (do inglês *Random Forest*) (BREIMAN, 2001). A avaliação empírica da técnica proposta nos projetos de software livre Cassandra, Camel, Hadoop e Lucene, mostrou que a técnica proposta alcançou 57% (WIESE et al., 2015c) de sensibilidade e uma média de 72% de precisão. Além disso, foi observado que o contexto de solicitação de mudança foi o mais relevante para a predição de mudanças conjuntas. Quando combinada com o contexto de comunicação ou contexto de mudança, houve uma melhora na acurácia do modelo de predição. Quando os três contextos foram combinados, não houve uma melhora

significativa no desempenho do modelo. Entretanto, o uso de dois contextos em vez de três pode reduzir o esforço necessário para predizer as mudanças conjuntas utilizando a técnica proposta.

2.4.1 Aprendizado de Máquina com Classificação

A classificação é uma técnica de aprendizagem de máquina supervisionada, onde modelos (ou classificadores) são construídos para predizer classes (ou categorias) a partir de um conjunto de dados rotulados (ou características). O processo de classificação possui duas fases. Inicialmente, é feito o treino do modelo, ou seja, o algoritmo constrói um classificador por meio da análise do conjunto de dados de treino. Após isso, o classificador criado pode ser testado a fim de avaliar sua performance. Por fim, pode-se realizar a classificação, ou seja, o classificador rotula uma classe quando se fornece um conjunto de dados. A Figura 5 mostra uma visão geral desse processo de predição com a classificação.

O conjunto de dados de treino é composto por tuplas rotuladas. Uma tupla X, também denominada instância, é representada por um vetor, $X = (x_1, x_2, \ldots, x_n)$, com n medidas que descrevem valores de n atributos, respectivamente $(A_1, A_2, \ldots, A_n)$. Presume-se que cada tupla X do conjunto de treino pertence a uma classe pré-definida, determinada por um atributo denominado rótulo de classe, ou simplesmente classe. Essa classe é categórica (ou nominal), cujo valor é discreto e não ordenado (HAN et al., 2012).

O conjunto de regras aprendidas pelo algoritmo compõe o classificador que, dada uma instância como entrada, resulta em uma classificação dessa instância. Com o classificador construído, pode-se avaliar o desempenho desse classificador aplicando um conjunto de dados para teste. O conjunto de teste geralmente é uma parte menor do conjunto de dados disponível, e a outra parte maior é utilizada para o treino. No entanto, ao realizar o teste do classificador, a classe é utilizada para fins comparativos do previsto com o observado para avaliar o seu desempenho, conforme abordado na Subseção 2.4.2. Por fim, o classificador pode ser colocado em prática para predizer artefatos com base nos dados relacionados ao artefato que está sendo mudado.

Existem inúmeros algoritmos de classificação que foram usados em estudos de Engenharia de Software, como Máquina de Vetores de Suporte (do inglês *Support Vector Machine*), Naïve Bayes (BIÇER, S.; BENER, A. B.; ÇAĞLAYAN, B., 2011) e Regressão Logística (BIRD et al., 2009). Dentre eles, existe a Floresta Randômica (do inglês *Ran-*

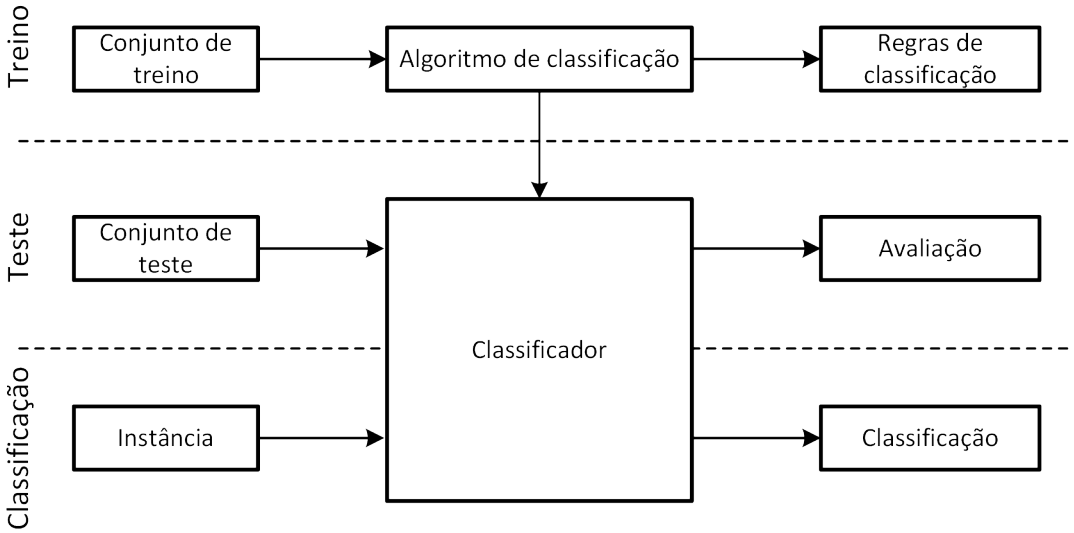


Figura 5: Visão geral do processo de classificação: treino, teste e classificação.
Fonte: Autoria própria.

dom Forest), utilizada recentemente por Wiese et al. (2015c). Esse algoritmo constrói um grande número de árvores de decisão para treinamento com o uso de um subconjunto de métricas, garantindo uma baixa correlação entre as métricas e, dessa forma, evitando o sobreajuste (do inglês *overfitting*) (BREIMAN, 2001). O sobreajuste ocorre quando um modelo apresenta uma alta acurácia para o conjunto de treino, mas quando aplicado sobre o conjunto de teste, a acurácia é muito baixa. Em outras palavras, é quando o modelo construído apresenta baixa capacidade de generalização (HAN et al., 2012).

A Floresta Randômica, utilizada nesse trabalho, é baseada em árvores de decisão. A árvore de decisão é estrutura de árvore que se assemelha a um fluxograma, onde cada nó interno denota um teste em um atributo, cada galho representa a saída do teste, e cada folha possui um rótulo de classe (HAN et al., 2012). O processo de classificação em uma árvore de decisão consiste em, a partir da raiz dessa árvore, chegar até as folhas dado um objeto a ser classificado.

Um exemplo de árvore de decisão está ilustrado na Figura 6. Nesse exemplo, pretende-se decidir em jogar golfe ou não, levando em consideração três atributos (fatores): tempo, umidade e vento. Para o atributo tempo, os valores possíveis são *ensolarado*, *nublado* e *chuvoso*. Para o atributo umidade, os dois valores possíveis são *alto* e *normal*. Por fim, para o atributo vento, os valores possíveis são *verdadeiro* e $falso$. As folhas da árvore de decisão são as classes, e os valores possíveis são *sim* ou *não*.

Um exemplo de cenário para decisão seria um dia ensolarado e com umidade alta, cuja classificação conforme a Figura 6 seria *sim*, jogar golfe.

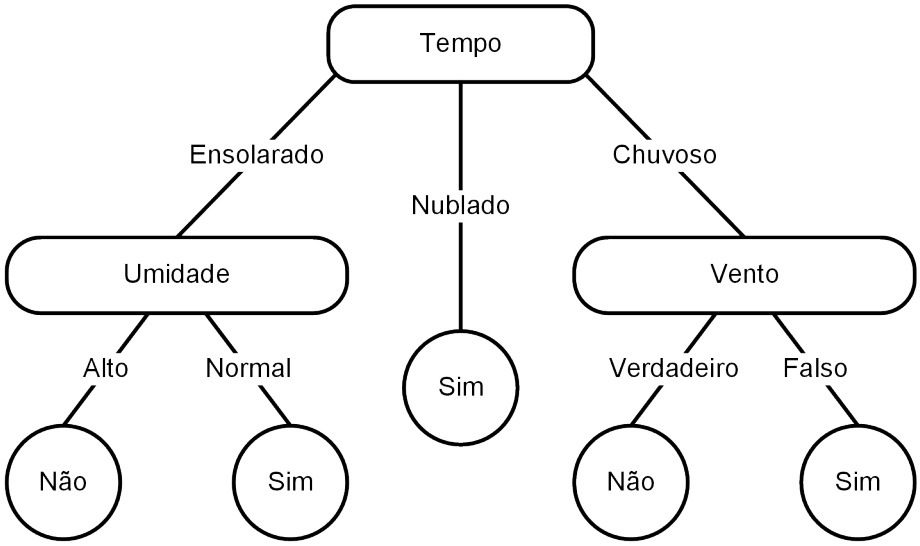


Figura 6: Exemplo de árvore de decisão.
Fonte: Adaptado de Quinlan (1986).

2.4.2 Avaliação dos Modelos de Predição

Para avaliar o poder de predição dos modelos, geralmente são utilizados uma parte do conjunto de dados disponível, sendo que a outra parte dos dados são utilizados para o treino do modelo de predição. Aplicando esses dados de teste ao classificador, pode-se comparar o resultado do modelo com base na instância de entrada do conjunto de teste (previsto) com a classificação correta daquela instância testada (observado) e categorizar em uma matriz de confusão, ilustrado no Quadro 1.

As categorizações possíveis na matriz de confusão são: Verdadeiro Positivo (VP), Verdadeiro Negativo (VN), Falso Positivo (FP) e Falso Negativo (FN). Quanto ao significado de cada categoria, o VP é o número de instâncias positivas preditas corretamente, o VN é o número de instâncias negativas preditas corretamente, o FP é o número de instâncias negativas preditas incorretamente e o FN é o número de instâncias positivas preditas incorretamente.

Para avaliar o modelo de predição, podem ser calculadas as medidas precisão, sensibilidade e medida-F, e geralmente são reportadas em estudos na literatura (POWERS, 2007; AVAZPOUR et al., 2014):

Quadro 1: Matriz de confusão.

	Observado Verdadeiro	**Observado Falso**
Previsto Verdadeiro	VP	FP
Previsto Falso	FN	VN

Fonte: Autoria própria.

- **Precisão (do inglês *Precision*)**: proporção de instâncias classificadas corretamente, calculada pela Equação 1. Os valores resultantes da equação variam entre 0 (zero) e 1 (um), onde 1 (um) significa a precisão perfeita.

$$Precisão = \frac{VP}{VP + FP} \tag{1}$$

- **Sensibilidade (do inglês *Recall*)**: proporção de instâncias verdadeiras classificadas incorretamente, calculada pela Equação 2, cujo resultado varia de 0 (zero) a 1 (um), onde o valor de 1 (um) significa a ausência de falso negativo.

$$Sensibilidade = \frac{VP}{VP + FN} \tag{2}$$

- **Medida-F (do inglês *F-measure* ou *F-score*)**: média harmônica das medidas da precisão e sensibilidade, calculada pela Equação 3, resultando em valores entre 0 (zero) e 1 (um), onde 1 (um) indica que todas as instâncias foram classificadas corretamente, ou seja, é o desempenho perfeito de um classificador.

$$Medida - F = 2 \times \frac{Sensibilidade \times Precisão}{Sensibilidade + Precisão} \tag{3}$$

Outra forma de avaliar o modelo preditivo é com a **Área sob a Curva ROC**, (do inglês *Area Under an ROC Curve* ou *AUC*). A Curva ROC, ou Curva de **Característica de Operação do Receptor** (do inglês *Receiver Operating Characteristic* ou *ROC*) é um gráfico onde são plotados a taxa de FP, dada pela Equação 4, no eixo x e a taxa de VP, dada pela Equação 5, no eixo y de cada limiar utilizado. A Figura 7 mostra um exemplo de um gráfico com a curva ROC. A Área sob a Curva ROC é a área abaixo da linha plotada, cujo resultado varia entre 0 (zero) e 1 (um), onde valores altos representam modelos de predição melhores. Um modelo de predição randômico tem o valor de UAC de $0,5$, representado pela linha diagonal no gráfico da Figura 7.

$$Taxa\ de\ FP = \frac{FP}{FP + VN} \tag{4}$$

$$Taxa\ de\ VP = \frac{VP}{VP+FP} \tag{5}$$

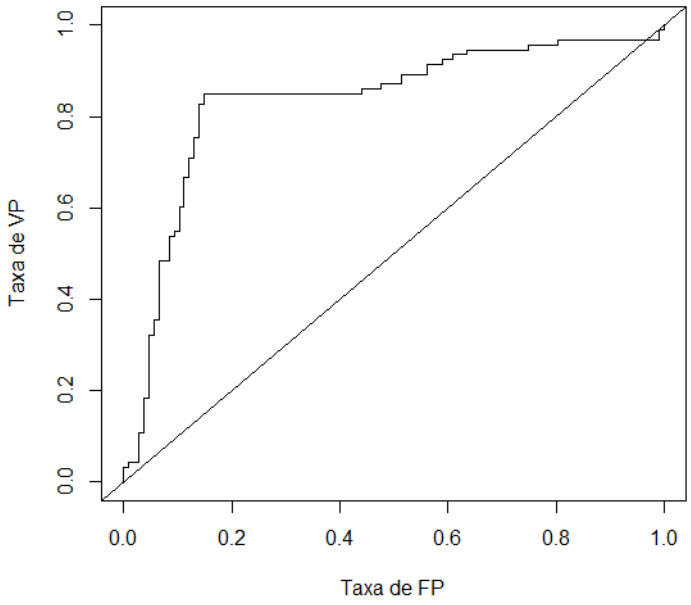


Figura 7: Exemplo da Curva ROC.

Fonte: Autoria própria.

Os estudos que exploram abordagens com modelos de predição sempre buscam modelos robustos, que apresentam altos valores de sensibilidade e precisão. Atingir valores altos de sensibilidade e precisão é um desafio para os estudos, uma vez que existe um *tradeoff* entre a sensibilidade e a precisão, pois aumentando um, diminui o outro e vice-versa (AVAZPOUR et al., 2014). No entanto, essas medidas não consideram todos os valores negativos (VN e FN), podendo resultar em um viés caso não reportadas com cuidado (POWERS, 2007). Para que este viés não ocorra, pode-se reportar, por exemplo, o Coeficiente de Correlação de Matthews, obtido pela Equação 6.

- **Coeficiente de Correlação de Matthews (do inglês *Matthews Correlation Coefficient* ou *MCC*)**: é a correlação entre classificações observadas e as preditas.

$$MCC = \frac{(VP \times VN) - (FP \times FN)}{\sqrt{(VP+FP) \times (VP+FN) \times (VN+FP) \times (VN+FN)}} \tag{6}$$

Outro ponto importante do processo de classificação é quanto ao esforço necessário para realizá-lo. Do ponto de vista computacional, quanto mais características

existirem em uma instância, maior o esforço computacional necessário para realizar a classificação e capturar as características que compõem cada instância. Uma técnica que tem por objetivo diminuir esse esforço é a seleção de atributos, também conhecido como seleção de características (*feature selection*).

A seleção de características consiste em selecionar um subconjunto de características mais relevantes para o classificador (BLUM; LANGLEY, 1997), o que diminui a complexidade quando há muitas características e também ajuda pesquisadores a entender os dados utilizados na construção do classificador. Por relevância, entende-se que, usando um subconjunto de características x pertencente ao conjunto X ($x \subset X$), resulta em uma acurácia do classificador melhor que a acurácia alcançada usando todo o conjunto de características X. Além disso, a seleção de características pode evitar o superajuste (do inglês *overfitting*) do classificador (KOHAVI; JOHN, 1997; GUYON et al., 2003).

2.5 FERRAMENTAS DE PREDIÇÃO DE MUDANÇAS CONJUNTAS

Muitas técnicas foram propostas para predizer mudanças e apoiar a propagação de mudanças, mas poucas foram colocadas em prática por meio de ferramentas. No contexto de predição de mudanças, as principais ferramentas propostas foram o ROSE[2] (ZIMMERMANN et al., 2005), o JRipples[3] (BUCKNER et al., 2005) e o ImpactMiner[4] (DIT et al., 2014).

A ferramenta ROSE, proposta por Zimmermann et al. (2005), é um *plugin* para o ambiente de desenvolvimento integrado (IDE) denominado Eclipse. O objetivo do ROSE é de sugerir aos desenvolvedores modificações de artefatos a partir de outros que estão sendo modificados. Para tanto, são extraídas regras de associação com base no histórico do sistema de controle de versão, utilizando limiares sobre suporte e confiança com o algoritmo denominado *Apriori*.

Outras ferramentas encontradas servem de apoio à análise de impacto de mudança. Uma delas é o JRipples, uma ferramenta que visa a facilitar e sistematizar as mudanças incrementais em um software, especificamente desenvolvidos na linguagem Java (BUCKNER et al., 2005). Essa ferramenta, assim como a ferramenta ROSE, é um *plug-in* para o Eclipse que utiliza grafo de dependências estruturais criadas a par-

[2]Disponível em: `https://www.st.cs.uni-saarland.de/softevo/erose/`
[3]Disponível em: `http://jripples.sourceforge.net/`
[4]Disponível em: `http://www.cs.wm.edu/semeru/ImpactMiner/`

tir de uma análise estática do código-fonte para levantar os artefatos, e até mesmo os métodos, que são propensos a mudarem juntos. Com isso, a ferramenta fornece ao desenvolvedor uma orientação metodológica predizendo os artefatos a serem visitados e possibilitando marcar cada artefato como impactado ou visitado (não impactado). Apesar do estudo não avaliar empiricamente a ferramenta JRipples, os autores reconhecem que a sensibilidade é importante para a qualidade da análise de impacto de mudança.

Outra ferramenta que é um *pug-in* para o Eclipse é o Chianti (REN et al., 2004), que visa demonstrar os testes impactados por alguma mudança no software desenvolvido em Java. Essa ferramenta analisa duas versões de um software e extrai um conjunto de mudanças atômicas comparando duas versões desse software. Para tanto, são utilizados dados do sistema de controle de versão. O resultado do experimento empírico, realizado com histórico de um ano do sistema de controle de versão do software denominado Daikon, mostrou que, após uma mudança no software, o conjunto de testes resultante representava, na média, 52% de todos os possíveis testes. Além disso, para cada teste afetado, o número de mudanças foi pequeno, atingindo um valor médio de 3,95%.

O ImpactMiner é uma ferramenta que prediz um conjunto de artefatos a serem alterados dada uma solicitação de mudança auxiliando, assim, desenvolvedores a realizar uma análise de impacto de mudança dessa solicitação (DIT et al., 2014). Para tanto, é utilizada uma técnica que combina diferentes tipos de técnicas para extração de dados: recuperação de informações em dados textuais, análise dinâmica em dados de execução e mineração de repositórios de software em dados de mudanças (GETHERS et al., 2012), resultando em três diferentes tipos de informações.

No experimento empírico realizado pelos autores da técnica utilizada pelo ImpactMiner, os resultados mostraram que a combinação dessas três fontes de informações melhorou em até 17% de precisão e 41% de sensibilidade quando comparada com a utilização de apenas uma das fontes de informações, especificamente aquela extraída pela técnica de recuperação de informações (GETHERS et al., 2012). O melhor caso para precisão foi de 18%, mas com a sensibilidade de 23%. Em contraste, o melhor caso para sensibilidade foi de 75%, mas a precisão foi de 7%. Esse resultado mostra, além do *trade-off* entre precisão e sensibilidade, valores baixos para ambas as medidas, o que talvez pode comprometer o uso da ferramenta por apresentar muitos falsos negativos.

Sun et al. (2015) comparou três ferramentas de predição de mudanças conjuntas. Os autores compararam a ferramenta (1) ROSE, que mede o acoplamento de

mudança com base na frequência de mudanças conjuntas dos artefatos no histórico do projeto (ZIMMERMANN et al., 2005); a ferramenta (2) Columbus, que mede o acoplamento estrutural com base na análise da arquitetura do software (FERENC et al., 2002); e a ferramenta (3) IRC2M, que mede o acoplamento semântico com base no grau de similaridade dos identificadores e comentários do código-fonte (POSHYVANYK et al., 2009). Os resultados indicaram que o acoplamento por mudança e conceitual capturam melhor as mudanças conjuntas entre dois artefatos. Porém, o objetivo do estudo foi replicar os acoplamentos em uma base comum de projetos, já que esses acoplamentos não haviam sido comparados nos mesmos projetos e era difícil determinar qual técnica obtinha melhor desempenho.

2.6 CONSIDERAÇÕES FINAIS

A análise de impacto é importante para a manutenção do software, uma vez que identifica partes do software que são afetadas por uma mudança baseada em acoplamentos em software como o acoplamento estrutural, semântico e/ou de mudança.

Para apoiar a análise de impacto, técnicas e ferramentas que exploram os diversos tipos de acoplamentos foram propostas e avaliadas. Entretanto, um dos desafios dessas técnicas de predição de mudanças conjuntas são o desempenho, ou seja, predições feitas de forma correta. Nesse sentido, a técnica de aprendizado de máquina com classificação utilizando informações contextuais mostrou que pode reduzir a quantidade de falsos alertas (WIESE et al., 2015c), o que pode facilitar a aceitação e, consequentemente, seu uso futuramente pelos desenvolvedores. Apesar ter sido avaliada empiricamente, até então, essa técnica não havia sido implementada em uma ferramenta.

Embora ferramentas que utilizem técnicas de predição de mudanças conjuntas possam ser encontradas na literatura, nenhum dos estudos encontrados na literatura avaliaram a(s) ferramenta(s) de uma perspectiva prática com potenciais usuários. Um dos motivos disso também é o fato das técnicas apresentarem um baixo desempenho, ou seja, um número alto de falsos-positivo, o que reforça a motivação para desenvolver uma ferramenta com a técnica de classificação para predição de mudanças conjuntas.

3 RECOMINER: FERRAMENTA DE PREDIÇÃO DE MUDANÇAS CONJUNTAS

3.1 CONSIDERAÇÕES INICIAIS

Nesse capítulo, é apresentada a ferramenta desenvolvida bem como sua motivação e, principalmente, sua metodologia. A ferramenta desenvolvida, nomeada Recominer, automatiza técnicas de Regras de Associação e Classificação para executar a predição de mudanças conjuntas em um projeto de software, apresentada na Seção 3.2. Dessa forma, a ferramenta apresenta as predições de mudanças conjuntas (artefatos de código-fonte) e suas probabilidades de mudanças resultantes de cada técnica em particular ao desenvolvedor. Com base nisso, o desenvolvedor pode fazer uma análise de impacto de mudança baseado nessas predições, minimizando o esforço necessário para realizar tarefas de manutenção de software. Além disso, a fim de realizar um experimento, a ferramenta permitiu que os desenvolvedores indicassem os artefatos e opinassem sobre a tarefa. Neste capítulo também é descrita a arquitetura e as técnicas utilizadas, detalhadas na Seção 3.3. Por fim, finalizando este capítulo, são apresentadas as limitações da ferramenta.

3.2 PROJETO

O objetivo desse trabalho consistiu em automatizar a predição de mudanças conjuntas entre dois artefatos utilizando a frequência de mudanças durante a evolução dos artefatos (Regras de Associação) propostas por Zimmermann et al. (2005) e o uso das informações contextuais coletadas das solicitações de mudanças, da comunicação e de cada mudança (Classificação) propostas por Wiese et al. (2015c).

Para atingir esse objetivo, uma ferramenta foi desenvolvida para que coletasse as informações dos repositórios de software necessárias para predição de mudanças conjuntas. Para as Regras de Associação, são necessárias informações do sistema de controle de versão. Já para a Classificação, além das informações do sistema de controle

de versão, são necessárias as informações do sistema de gerenciamento de tarefas (*issues tracker*). Com base nessas informações, a ferramenta é capaz de executar as Regras de Associação e Classificação com base nessas informações contextuais devidamente processadas, seja extraindo métricas, filtrando informações, entre outros. Ao final desse processo, o resultado são os artefatos propensos a mudarem em uma tarefa, ou seja, as possíveis mudanças conjuntas, dado um artefato que já tenha sido modificado.

O desenvolvimento de uma ferramenta para esse contexto foi motivado pelo fato de não ter sido encontrada nenhuma outra ferramenta capaz de fazer predições de forma automática com base em métricas extraídas de repositórios de software. Embora existam ferramentas que compartilhem o mesmo propósito de predizer mudanças conjuntas, como o ROSE (ZIMMERMANN et al., 2005) e o ImpactMiner (DIT et al., 2014), elas não utilizam métricas extraídas de repositórios de software aplicando algoritmos de aprendizagem. Além disso, diferentemente de outras ferramentas relacionadas, a ferramenta proposta captura o *feedback* dos desenvolvedores sobre as mudanças preditas, o que pode contribuir para a melhoria do estado da arte no que diz respeito a técnicas de predição de mudanças de artefatos. O *feedback* pode fornecer dados quantitativos e qualitativos em relação à ferramenta e às técnicas implementadas na ferramenta. Esses dados podem fornecer resultados para melhoria da ferramenta e das técnicas, além de reforçar que seu uso pode apoiar desenvolvedores na manutenção do software. A tabela Tabela 1 mostra uma visão geral da diferença da ferramenta proposta, Recominer, com outras ferramentas descritas no Capítulo 2. Essa tabela destaca as diferenças em relação às técnicas utilizadas, os dados utilizados e coletados, além da granularidade da predição.

Tabela 1: Comparação das ferramentas de apoio à análise de impacto de mudança.

		ROSE	Chianti	JRipples	ImpactMiner	Columbus	IRC2M	Recominer
Técnicas utilizadas	Acoplamento estrutural		x	x	x	x		
	Regras de Associação	x			x			x
	Recuperação de Informações				x		x	
	Classificação							x
	Abordagem híbrida				x			
Dados utilizados	Dados contidos no código-fonte	x	x	x		x	x	
	Dados do histórico de mudanças	x	x		x			x
	Dados das solicitações de mudanças							x
Dados coletados	Coleta de feedback							x
Granularidade da predição	Atributos e Métodos	x	x		x			
	Arquivos	x	x	x	x	x	x	x

Fonte: Autoria própria.

3.3 METODOLOGIA DA FERRAMENTA

A ferramenta Recominer[1] foi desenvolvida com o apoio de ferramentas de mineração de dados e de análise estatística. Para mineração de dados dos repositórios de software, especificamente o sistema de controle de versão, como o Git e Subversion, e o sistema de gerenciamento de solicitações de mudanças, como o Jira e o Bugzilla, foram adotadas duas ferramentas de código aberto, denominados CVSAnalY e Bicho, respectivamente. Ambas foram desenvolvidas na linguagem Python e coletam as informações dos repositórios e armazenam em um banco de dados relacional, como o MySQL. Para realizar a predição de mudanças conjuntas com Classificação, foi utilizada a ferramenta R que é amplamente utilizada pela comunidade científica (R DEVELOPMENT CORE TEAM, 2008). Especificamente, foi utilizado o algoritmo Floresta Randômica (do inglês *Random Forest*), disponível no pacote `randomForest`[2]. Para auxiliar nos cálculos de métricas de grafos, explicados adiante na ßsub:calculometricas, foi utilizada a biblioteca denominada JUNG[3] (*Java Universal Network/Graph Framework*), que fornece recursos para modelagem, análise e visualização de dados que podem ser representadas em grafo ou rede.

Como essas ferramentas não possuem integração e são de diferentes linguagens, é necessário que sejam integradas a fim de automatizar a técnica de predição de mudanças conjuntas. Para esse fim, foi utilizada a linguagem Java[4] na versão 8.

Tecnicamente, as ferramentas de extração de dados, o Bicho e o CVSAnalY, são independentes e não fornecem uma forma de integrá-las, seja em forma de Interface de Programação de Aplicações (do inglês *Application Programming Interface* ou API) ou bibliotecas. Por causa disso, foi necessário fazer com que a ferramenta Recominer disparasse o comando para o sistema operacional executar o Bicho e o CVSAnalY. Em relação à ferramenta R, apesar de fornecer outras formas de integrar com a linguagem Java, optou-se por executá-lo via comando do sistema operacional também, simplificando o desenvolvimento da ferramenta, já que o mesmo foi feito para o Bicho e o CVSAnalY.

Para que os dados do Bicho e do CVSAnalY fossem disponibilizados para Recominer, utilizou-se o mesmo banco de dados relacional, o MySQL. Já para integração com o R, que é responsável pela execução do algoritmo de Classificação, foi necessário tra-

[1]Disponível em: https://github.com/rodrigokuroda/recominer
[2]Disponível em: https://cran.r-project.org/web/packages/randomForest/
[3]Disponível em: http://jung.sourceforge.net/
[4]Disponível em: https://www.java.com/

balhar com arquivos CSV (*Comma-separated values*), ou seja, a ferramenta Recominer cria os arquivos CSV que serão consumidos pelo R e, após o processamento, é criado um CSV com o resultado gerado pelo R e lido pela ferramenta Recominer. Toda a integração foi feita utilizando o processamento em lote (*batch*), que pode ser programado para executar periodicamente. O processamento em lote também foi utilizado para fazer todo o processamento dos dados necessários para a predição de mudanças conjuntas e executar a predição em si. O responsável por toda essa integração é o módulo principal da ferramenta Recominer.

A interface web é um outro módulo da ferramenta Recominer, ficando independente do módulo principal. No lado do cliente (*client-side*) foi utilizado o framework JavaScript denominado AngularJS[5] na versão 1 com o *framework* de componentes de interface denominado AngularJS Material[6]. No lado do servidor (*server-side*) foi utilizada a linguagem Java na versão 8 para prover os serviços web REST que fornecerão os dados que serão consumidos pelo AngularJS e fazer a comunicação com o banco de dados.

A visão geral da arquitetura para integração das ferramentas citadas é apresentada na Figura 8. É interessante notar que essa arquitetura permite que outras ferramentas sejam integradas de forma que utilizem os dados coletados e processados, pois dados podem ser disponibilizados por meio de serviços web utilizando REST, ou Transferência de Estado Representacional (do inglês *Representational State Transfer*), o que pode facilitar a extensão e integração com futuras ferramentas que venham a fazer uso das informações geradas do Recominer.

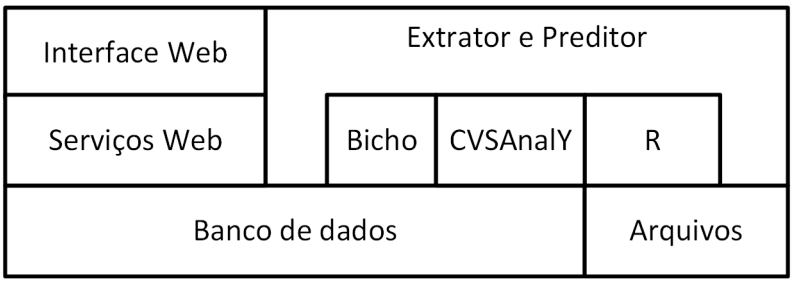


Figura 8: Visão geral da arquitetura da ferramenta Recominer.
Fonte: Autoria própria.

A Figura 9 mostra a interface Web da ferramenta exibida no navegador, na qual o usuário interage e visualiza as predições dadas pelas técnicas. Ao acessar o projeto,

[5]Disponível em: https://angularjs.org/
[6]Disponível em: https://material.angularjs.org/

primeiro é exibido a (1) lista de projetos disponíveis no banco de dados e que foram processados pela ferramenta, ou seja, que tenha predições por alguma técnica. Ao selecionar um projeto, então é exibida a (3) lista de tarefas disponíveis e com predições. Quando é (2) selecionada uma tarefa, é exibida uma (5) lista de predições daquela tarefa, bem como a probabilidade calculada pela técnica. Esses artefatos listados são preditos com base no artefato inicial indicado no (5) cabeçalho da lista de artefatos preditos. Uma (4) descrição da tarefa no experimento é exibida, contendo o seguinte texto: "Dado que o artefato a/b/c foi modificado para corrigir a tarefa T do projeto P, os artefatos que provavelmente mudariam para completar essa tarefa são apresentados abaixo. Quais desses artefatos preditos você mudaria para completar a tarefa T, dada a descrição?". Para visualizar a descrição da tarefa do projeto, é necessário clicar no botão (6) "Descrição", localizada acima da lista de artefatos preditos. O *feedback* pode ser escrito no (7) campo de texto, que possui uma breve descrição: "Justifique a escolha dos arquivos que você mudaria ou não para completar a tarefa T". Esse *feedback*, bem como as indicações dos artefatos que o usuário mudaria para completar a tarefa, podem ser enviados clicando no (7) botão "Enviar", localizado ao lado do campo de *feedback* (7) no final da lista de artefatos preditos.

A técnica de Wiese et al. (2015c) utiliza o aprendizado de máquina, especificamente a classificação com o algoritmo Floresta Randômica. Essa técnica cria modelos capazes de predizer mudanças conjuntas a partir de métricas classificadas, em que, no contexto desse trabalho, a classe significa se a mudança conjunta aconteceu ou não com determinadas métricas. Neste trabalho, as métricas foram extraídas a partir de informações das solicitações de mudanças e dos *commits*, denominadas métricas contextuais. Para complementar a técnica, a ferramenta possui um recurso para capturar informações sobre as predições por meio do *feedback* dos desenvolvedores. As predições de mudanças conjuntas são feitas a cada novo *commit* que esteja associado a uma solicitação de mudança em aberto (não concluída). Para fornecer um parâmetro de comparação e funcionar como complemento, a técnica de Zimmermann et al. (2005) também foi adotada. Essa técnica pode ser aplicada utilizando apenas o histórico de mudanças dos artefatos, pois é baseada na técnica de mineração de dados denominada Regras de Associação. É importante enfatizar que as duas técnicas não são concomitantes, ou seja, não formam uma abordagem híbrida e são executadas independentemente uma da outra. Assim, a ferramenta apresenta as predições de cada técnica individualmente.

Em suma, a ferramenta analisa periodicamente cada artefato modificado de um

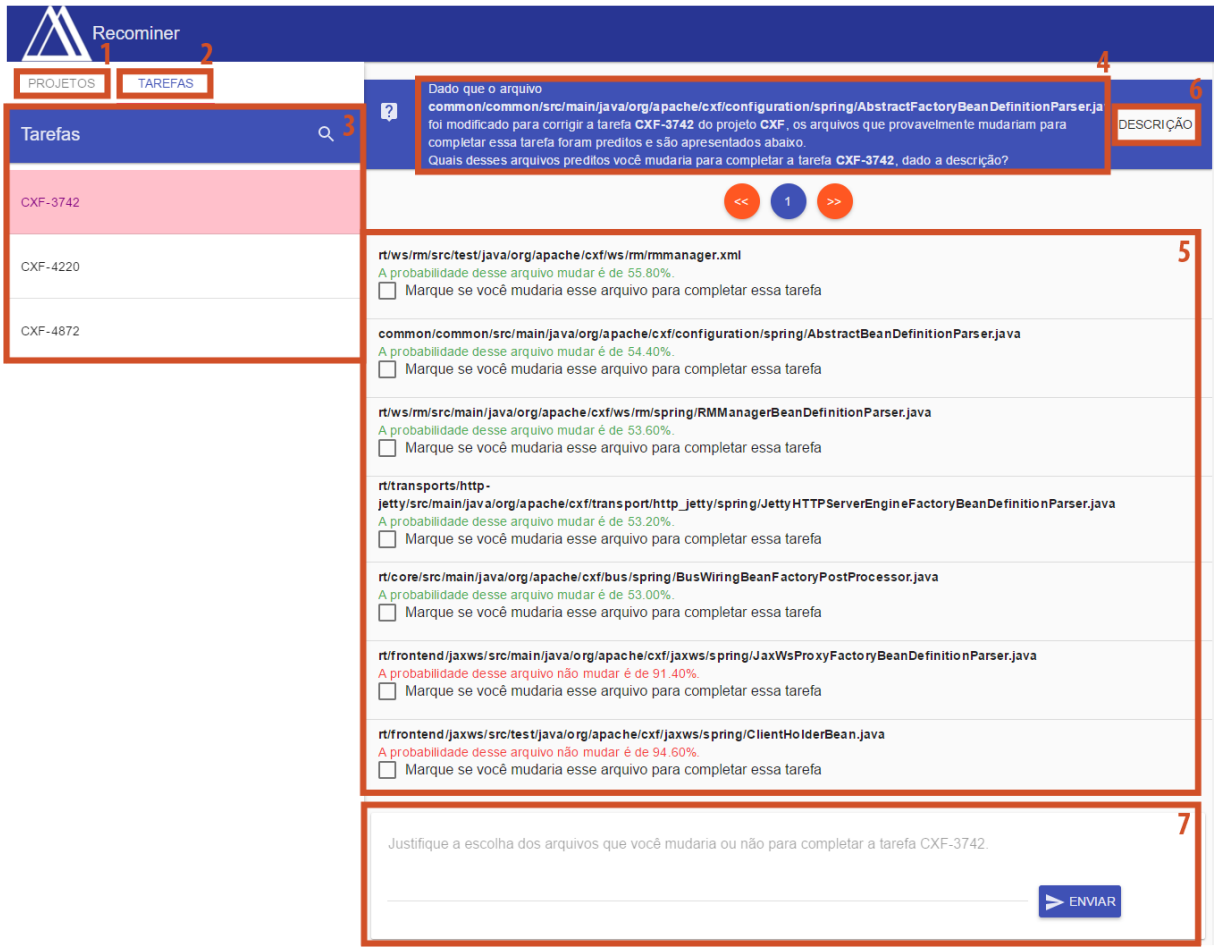


Figura 9: Interface Web da ferramenta Recominer.

Fonte: Autoria própria.

novo *commit* com o objetivo de predizer as mudanças conjuntas que são propensas a ocorrerem com o artefato recém-modificado. Para que isso seja possível, é necessário primeiramente coletar os dados de ambos os repositórios e relacioná-los. Em seguida, ao identificar um *commit*, todos os artefatos modificados são analisados para identificar possíveis mudanças conjuntas e gerar um conjunto de métricas rotuladas para cada mudança conjunta. Com as métricas rotuladas, é construído o modelo de predição, que classifica as métricas baseado no novo *commit*, resultando nas predições de mudanças conjuntas. Em paralelo, as regras de associação serão aplicadas com base no histórico de mudança do artefato. Essas predições podem ser apresentadas e utilizadas para apoiar os colaboradores do projeto de software em tarefas de manutenção. A Figura 10 apresenta, de forma geral, todo o processo que a ferramenta executará de forma automática. Todo esse processo foi dividido em subprocessos, os quais são explicados

sequencialmente com mais detalhes nas subseções seguintes.

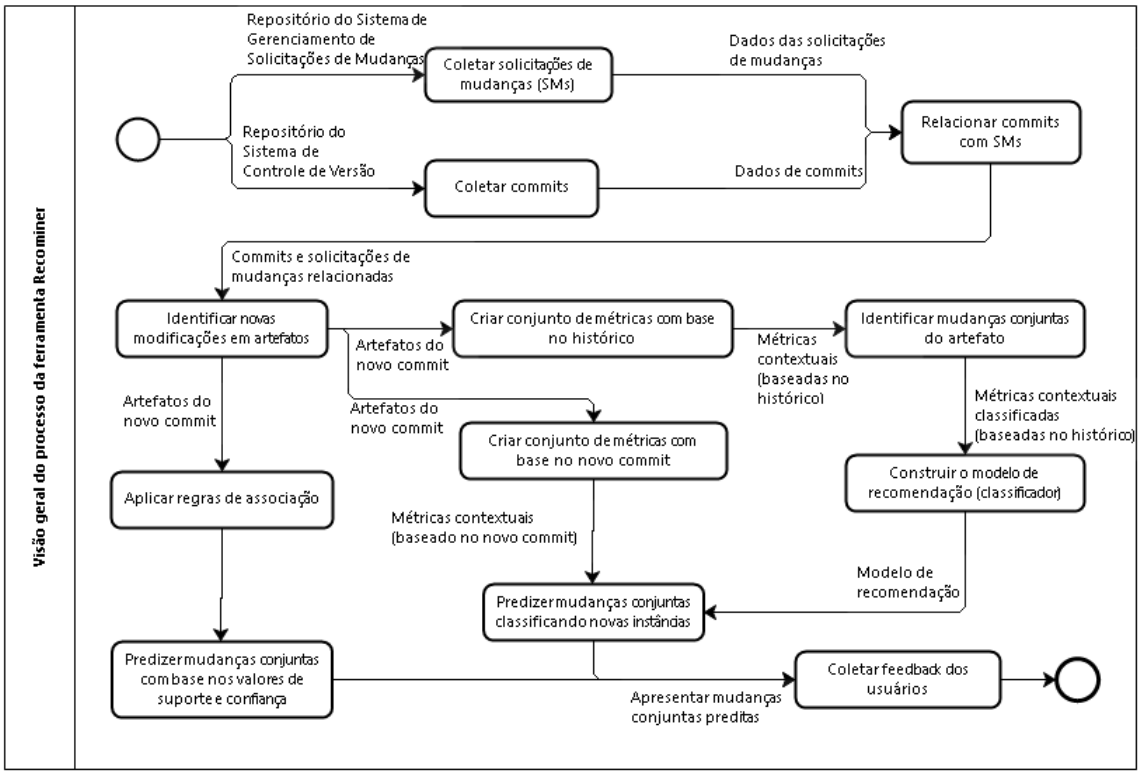


Figura 10: Visão geral do processo da ferramenta.

Fonte: Autoria própria.

3.3.1 Coletar solicitações de mudanças

O processo de desenvolvimento e manutenção de software é composto por diversas solicitações de mudanças (SMs, ou *issues* em inglês). Uma solicitação de mudança pode ser o relato de algum colaborador sobre um defeito ou um novo requisito que precisa ser atendido, e geralmente são gerenciados pelo sistema de gerenciamento de solicitações de mudanças, ou *issue tracker* em inglês.

O sistema de gerenciamento de solicitações de mudanças mantém informações sobre a solicitação de mudança, como o tipo de solicitação, o solicitante/relator, o responsável pela solicitação, a situação da solicitação de mudança, a data da solicitação e a data de correção, além de fornecer mecanismos de comunicação assíncrona entre os colaboradores, incluindo os próprios desenvolvedores. Para coletar esses dados, foi utilizado o software denominado Bicho[7], que armazena os dados obtidos em um banco

[7]Disponível em: https://github.com/MetricsGrimoire/Bicho

de dados relacional, como o MySQL.

3.3.2 Coletar commits

Para concluir uma solicitação de mudança, o desenvolvedor pode precisar modificar diversos artefatos por meio de um ou mais *commits*. Os *commits* dizem respeito ao envio e à frequência de modificações realizadas para o SCV, onde as modificações de cada artefato são versionadas.

Um *commit* possui alguns dados como a data do envio, o autor, as linhas adicionadas e removidas de cada um dos artefatos de código-fonte e uma mensagem escrita pelo autor. Esses dados são coletados com o software denominado CVSAnalY[8], que, assim como o Bicho, armazena os dados em um banco de dados relacional.

3.3.3 Relacionar *commits* com solicitações de mudanças

Como as duas fontes de informações não são relacionadas, é necessário associar as solicitações de mudanças com os respectivos *commits*. Isso é necessário, pois a técnica de Classificação necessita de métricas calculadas a partir dos dados das solicitações de mudanças, que por sua vez devem estar associadas aos *commits*.

Em projetos de software livre, geralmente os *commits* possuem uma mensagem contendo um padrão que o associa a uma solicitação de mudança. Esse padrão depende do sistema de gerenciamento de solicitações de mudanças adotado pelo projeto e pode variar de projeto para projeto. Alguns exemplos de identificação da solicitação de mudança na mensagem do *commit* são "`PROJETO-42`" e "`#314`". Por meio dessa identificação na mensagem, o *commit* é relacionado à respectiva solicitação de mudança. É importante ressaltar que existem casos em que um *commit* está relacionado com mais de uma solicitação de mudança. Entretanto, não existe garantia que o desenvolvedor inseriu uma identificação da solicitação de mudança na mensagem e se a identificação é correta.

3.3.4 Identificar novas modificações em artefatos

Com os *commits* associados a solicitações de mudanças, são identificadas novas mudanças em artefatos. Como um conjunto de artefatos são modificados em um único

[8]Disponível em: https://github.com/MetricsGrimoire/CVSAnalY

commit, primeiro é preciso identificar os novos *commits* realizados. São considerados novos aqueles *commits* que estão associados à uma solicitação de mudança que não está concluída. O novo *commit* é usado como base para predição, ou seja, a partir dele é criado o conjunto de métricas que são aplicadas ao modelo de predição.

Na ferramenta, cada artefato desse novo *commit* recebe suas predições individualmente e, portanto, é identificado e analisado individualmente. Artefatos que não sejam textuais, como imagens e artefatos binários, não são considerados, pois a técnica de Classificação também conta com métricas calculadas a partir do texto contido no artefato, como o número de linhas modificados do artefato de código-fonte é calculado.

3.3.5 Criar conjunto de métricas com base no histórico

Quando um artefato é modificado em um novo *commit*, a ferramenta deverá gerar um conjunto de métricas com base em cada modificação do artefato no passado. O conjunto de métricas é um dos elementos que torna possível predizer mudanças conjuntas com a técnica de Classificação, uma vez que é utilizada para criar um modelo de predição.

As métricas adotadas nesse trabalho podem ser divididas nas seguintes dimensões (WIESE et al., 2014, 2015c): Contexto das Solicitações de Mudanças, Contexto da Comunicação e Contexto da Mudança. Essas métricas são conhecidas como informações contextuais (WIESE et al., 2017). Embora existam inúmeras métricas que podem ser capturadas, é importante ressaltar que as métricas selecionadas apresentaram resultados satisfatórios em estudos preliminares (WIESE et al., 2014, 2015c). A seguir, são apresentadas cada uma dessas dimensões, bem como a sua justificativa em usar cada uma das métricas que compõe cada contexto.

3.3.5.1 Métricas do contexto da solicitação de mudança

As mudanças conjuntas podem ser necessárias para resolver determinados tipos de solicitação de mudanças. Nesse sentido, certas mudanças conjuntas são mais propensas de ocorrer em correções de defeitos e outras na implementação de novos requisitos. Além disso, um desenvolvedor pode trabalhar em solicitações de mudança de partes similares do software, o que o torna mais propenso a modificar artefatos e submeter contribuições para um mesmo conjunto de artefatos. O mesmo pode ocorrer com um colaborador que relata a solicitação de mudança, pois ele pode ser propenso a rela-

tar problemas para um mesmo componente ou que afetem a mesma parte do software, uma vez que seus interesses podem ser para requisitos específicos do software. Assim, os relatos de um usuário específico podem ser relacionados a um mesmo conjunto de artefatos. Essas informações são fatores que podem levar a determinadas mudanças conjuntas ocorrerem (WIESE et al., 2015c). As métricas que capturam esses fatores estão apresentadas no Quadro 2.

Quadro 2: Métricas no contexto da solicitação de mudança.

Métrica	**Tipo**	**Definição**
Tipo da solicitação de mudança	Textual	O Tipo da solicitação de mudança indicada quando a solicitação de mudança foi descrita. Alguns exemplos são: Defeito, Melhoria, Nova Tarefa, Documentação e Infraestrutura.
Responsável	Textual	Nome do Desenvolvedor indicado como responsável pela resolução da solicitação de mudança.
Relator	Textual	Nome do usuário ou desenvolvedor que relatou a solicitação de mudança.

Fonte: Wiese et al. (2015c)

3.3.5.2 Métricas do contexto de comunicação

Para completar uma solicitação de mudança, geralmente ocorrem discussões sobre ela. Essa comunicação envolve aspectos sobre uma determinada solicitação de mudança que podem levar a uma mudança conjunta a acontecer. Por exemplo, uma discussão pode ter determinada quantidade de palavras, comentários ou participantes porque é mais difícil de entender a solicitação de mudança. Isso pode significar um impacto em diversos artefatos, levando a mudança conjunta acontecer. Nesse sentido, as métricas do contexto de comunicação são importantes para predição de mudanças conjuntas, pois nessas condições certas mudanças conjuntas podem ser mais propensas de ocorrerem. O Quadro 3 apresenta as métricas do contexto de comunicação que são utilizadas nesse trabalho, mencionadas anteriormente. Essas métricas já foram utilizadas em nossos estudos anteriores (WIESE et al., 2015c, 2015a, 2015b).

Para calcular as métricas de comunicação, é levado em consideração apenas os comentários submetidos antes da data do *commit* que está sendo analisado e que está associado a solicitação de mudança.

Outros trabalhos estudaram o impacto da estrutura de comunicação utilizando a análise de redes (MENEELY et al., 2008; WOLF et al., 2009). A rede é representada por um grafo, onde os vértices do grafo são chamados de nó e as arestas são chamadas de

Quadro 3: Métricas do contexto de comunicação.

Métrica	Tipo	Definição
Número de comentários	Numérico	Número total de comentários feitos na solicitação de mudança.
Número de comentadores	Numérico	Número de usuários e desenvolvedores distintos que comentaram na solicitação de mudança.
Número de palavras	Numérico	Número total de palavras contidas na descrição e comentários realizados na solicitação de mudança.
Número de desenvolvedores que comentaram	Numérico	Número total de desenvolvedores distintos que previamente modificaram o artefato e também escreveram comentários na solicitação de mudança.

Fonte: Wiese et al. (2015c)

conexões. O grafo pode representar diversas estruturas, como a colaboração (MENEELY et al., 2008), a comunicação (WOLF et al., 2009; BIÇER, S.; BENER, A. B.; ÇAĞLAYAN, B., 2011; PANICHELLA et al., 2014) e relacionamentos sociotécnicos, baseadas em redes de dependências dos componentes de software em conjunto com redes de contribuição (BIRD et al., 2009).

A representação da comunicação em forma de grafo possibilita calcular outras métricas. Nesse trabalho, o grafo é construído composto por vértices, representado pelos autores das mensagens, e arestas, representadas pelas mensagens. É levada em consideração a ordem das mensagens para ligar os autores uns aos outros, pois o autor é ligado por uma aresta direcional a cada um dos outros autores que enviaram alguma mensagem anteriormente. A partir do grafo, são calculadas as suas propriedades, das quais incluem o tamanho, os laços, o diâmetro e a densidade. Essas métricas são apresentadas no Quadro 4. Dentre as métricas, o diâmetro e a densidade foram abordadas em nosso estudo anterior (WIESE et al., 2015a).

Por exemplo, suponha que, em uma solicitação de mudança, o desenvolvedor D_1 enviou uma dúvida. Algum tempo depois, o desenvolvedor D_2 respondeu a dúvida de D_1. Nesse caso, teríamos um grafo de dois vértices (D_1 e D_2) ligados por uma aresta direcional que aponta para D_1 ($D_1 \leftarrow D_2$). Caso outro desenvolvedor D_3 envie, depois, uma outra mensagem nessa solicitação de mudança, então seria adicionado o vértice D_3 no grafo, com mais duas arestas direcionais, apontando para D_1 e D_2 ($D_1 \leftarrow D_3 \rightarrow D_2$).

Além das propriedades do grafo, as métricas de centralidade podem ser calcu-

Quadro 4: Métricas de propriedades do grafo de comunicação.

Métrica	Tipo	Definição
Tamanho	Numérico	Quantidade de nós do grafo.
Laços	Numérico	Quantidade de arestas do grafo.
Diâmetro	Numérico	Tamanho máximo do caminho mais curto entre um par de nós do grafo.
Densidade	Numérico	Porcentagem de arestas do grafo em relação a quantidade máxima possível de arestas no grafo.

Fonte: Autoria própria.

ladas. As métricas de centralidade quantificam o quão próximo um nó está em relação a todos os outros no grafo (MENEELY et al., 2008), capturando, assim, a importância do nó. Nesse sentido, no contexto desse trabalho, essas métricas capturam o papel dos colaboradores na comunicação por meio de métricas de centralidade como a intermediação (do inglês *betweenness*) e proximidade (do inglês *closeness*) (MENEELY et al., 2008). Essas duas métricas são utilizadas neste trabalho e apresentadas no Quadro 5. É importante ressaltar que, como cada nó do grafo tem sua métrica de centralidade, é calculada a mediana de cada métrica de centralidade, uma vez que cada instância do conjunto de métricas representa uma única solicitação de mudança. É importante ressaltar que outros estudos já mostraram a importância das métricas de centralidade em outros problemas da Engenharia de Software (MENEELY et al., 2008; BIRD et al., 2009; WOLF et al., 2009; BIÇER, S.; BENER, A. B.; ÇAĞLAYAN, B., 2011; BETTENBURG; HASSAN, 2013; PANICHELLA et al., 2014).

Quadro 5: Métricas do papel do colaborador na comunicação (centralidade).

Métrica	Tipo	Definição
Centralidade de grau	Numérico	Número de vértices conectado ao nó.
Centralidade de intermediação	Numérico	Número de vezes que um nó do grafo serve como ponte ao longo de um caminho mais curto.
Centralidade de proximidade	Numérico	Indica o quão próximo um desenvolvedor está em relação aos outros. Pode ser considerado como uma medida de quanto tempo a informação leva para propagar em um grafo.

Fonte: Autoria própria.

Outras métricas interessantes são os buracos estruturais (do inglês *structural holes*) (BURT, 1995). Essas métricas denotam lacunas (falta de conexões) entre nós do grafo, indicando que os colaboradores de ambos os lados da lacuna têm acesso a

diferentes fluxos de informações. Em um estudo preliminar (WIESE et al., 2014), as métricas de buracos estruturais apresentaram evidências de que podem complementar a predição de propagação de mudanças quando utilizadas em classificadores. Dentre as métricas de buracos estruturais, foram utilizadas a restrição, a hierarquia, o tamanho efetivo e a eficiência, apresentadas no Quadro 6. Assim como as métricas de centralidade, é calculada a mediana de cada métrica de buracos estruturais, conforme feito em estudos anteriores (WIESE et al., 2015a).

Quadro 6: Métricas de buracos estruturais (*structural holes*) na comunicação.

Métrica	**Tipo**	**Definição**
Restrição (*Constraint*)	Numérico	Mede a ausência de buracos entre os nós vizinhos.
Hierarquia (*Hierarchy*)	Numérico	Mede a concentração de restrição em um nó.
Tamanho efetivo (*Effective size*)	Numérico	Indica a quantidade de nós vizinhos não redundantes de um nó.
Eficiência (*Efficiency*)	Numérico	Tamanho efetivo normalizado pela quantidade de vizinhos, indicando a proporção de nós vizinhos não redundantes.

Fonte: Adaptado de Wiese et al. (2014).

3.3.5.3 Métricas do contexto de mudança

Outros fatores que levam a mudança conjunta dos artefatos são as próprias mudanças que ocorreram. Os desenvolvedores podem contribuir para partes específicas do software, podendo modificar sempre os mesmos artefatos. Esse indicador pode ser útil para identificar quando dois artefatos são modificados conjuntamente se o mesmo conjunto de desenvolvedores modificam os mesmos artefatos. Além disso, os números de linhas adicionadas e excluídas do artefato podem indicar padrões de propensão de mudança conjunta. Alguns artefatos podem mudar mais linhas ou excluir mais linhas de código quando são modificados com alguns artefatos e apresentar um padrão diferente quando a quantidade de mudanças conjuntas é menor. Essas métricas foram utilizadas nesse trabalho e são apresentadas no Quadro 7.

3.3.6 Identificar mudanças conjuntas do artefato

A partir de um conjunto de métricas, é criado um modelo de predição para cada mudança conjunta que já ocorreu com o artefato em análise. Isso é feito pois, para a

Quadro 7: Métricas do contexto de mudanças.

Métrica	Tipo	Definição
Autor da mudança	Textual	Nome do desenvolvedor que modificou/realizou o *commit* do artefato.
Número de linhas de código inseridas	Numérico	Soma do número de linhas adicionadas/modificadas em um *commit*.
Número de linhas de código excluídas	Numérico	Soma do número de linhas excluídas em um *commit*.
Code churn	Numérico	Soma do número de linhas adicionadas e excluídas em um *commit*.

Fonte: Adaptado de Wiese et al. (2015c).

técnica de Classificação, cada mudança conjunta é analisada individualmente, buscando descobrir se a mudança conjunta é propensa a acontecer ou não.

Para que seja possível construir um modelo de predição, além do conjunto de métricas que compõe cada instância, é necessário que cada instância do conjunto esteja classificada (ou rotulado). Neste trabalho, a classificação é um valor binário (zero ou um) que indica se o artefato mudou ou não em conjunto com o artefato modificado no novo *commit*. Dessa forma, o algoritmo de classificação consegue saber que, com determinados valores de métricas, um artefato é propenso a mudar em conjunto ou não. O conjunto de instâncias classificadas é conhecido como treino.

As mudanças conjuntas foram encontradas utilizando as regras de associação, técnica proposta por Zimmermann et al. (2005). Dado que i e j são artefatos distintos, a regra de associação $i \Rightarrow j$ significa que a mudança no artefato i leva j também a mudar. No contexto desse trabalho, cada artefato j que foi modificado para concluir a solicitação de mudança é considerado uma mudança conjunta do artefato em análise i, considerando que i e j foram modificados em uma mesma solicitação de mudança. Entretanto, para um artefato ser considerado, este deve estar em *commits* associados à uma solicitação de mudança concluída. Além disso, esse *commit* deve ter sido realizado no período entre a criação da solicitação de mudanças até o momento da sua conclusão. A fim de evitar operações que envolvam a reestruturação de diretórios ou modificações de ramos (*branches*), *commits* com mais de vinte (20) artefatos não foram considerados.

Para classificar a mudança conjunta, é verificado se no *commit* ocorreu uma mudança nos dois artefatos. Considere o exemplo, apresentado na Tabela 2, de treino para predição de mudanças conjuntas do artefato A. Supondo que B e C já foram modificados com A, então são gerados dois treinos, uma para cada mudança conjunta.

Tabela 2: Exemplo de conjunto de dados de treino para o artefato A

Treino de mudança conjunta do artefato A com B				
Mudança conjunta	**Solicitação de mudança**	***Commit***	**Métricas contextuais**	**Classe**
AB	#10	#21	Métricas contextuais	1
AB	#19	#32	Métricas contextuais	0
AB	#92	#109	Métricas contextuais	1
AB	#96	#115	Métricas contextuais	0
AB	...	...	...	...
Treino da mudança conjunta do artefato A com C				
Mudança conjunta	**Solicitação de mudança**	***Commit***	**Métricas Contextuais**	**Classe**
AC	#10	#21	Métricas contextuais	0
AC	#19	#32	Métricas contextuais	1
AC	#92	#109	Métricas contextuais	1
AC	#96	#115	Métricas contextuais	0
AC	...	...	...	...

Nesse exemplo, o artefato *A* foi modificado nos *commits* #10, #19, #92 e #96, que estão respectivamente associados às solicitações de mudanças #21, #32, #109 e #115. No entanto, o artefato *A* mudou com *B* apenas nos *commits* #10 e #92, enquanto o artefato *A* mudou com *C* nos *commits* #19 e #92. No *commit* #96, não houve mudança conjunta de *A* com *B* e com *C*. É importante notar que a diferença está na classificação de cada conjunto de métricas, pois as métricas continuam as mesmas para os dois treinos, uma vez que as métricas são baseadas no artefato *A*.

Para exemplificar as métricas contextuais no conjunto de treino ou conjunto de teste da Tabela 2, um conjunto de métricas calculadas a partir de uma tarefa é apresentada na tabela Tabela 3. A tarefa escolhida foi uma tarefa de um projeto de código aberto denominado Apache CXF, que é explicado e usado adiante no Capítulo 4. As métricas foram calculadas a partir da tarefa CXF-4002[9] desse projeto.

3.3.7 Construir modelo de predição (classificador)

O conjunto de métricas classificadas é essencial para construir o modelo de predição. Nesse trabalho, os classificadores são construídos pela ferramenta com a finalidade de predizer as mudanças conjuntas a partir do conjunto de métricas classificadas. Para cada artefato modificado em um novo *commit* é construído um conjunto de mode-

[9]Disponível em: https://issues.apache.org/jira/browse/CXF-4002

Tabela 3: Exemplo de métricas contextuais.

Tipo da tarefa	**Prioridade da tarefa***	**Responsável pela tarefa***	**Autor da tarefa***
Bug	Major	dkulp	longbeach
Idade da tarefa	**Palavras na descrição**	**Palavras nos comentários**	**Comentadores**
122	444	924	8
Comentários	**Desenvolvedores que comentaram**	**Intermediação****	**Proximidade****
10	2	0	0,6905
Grau**	**Eficiência****	**Tamanho Efetivo****	**Restrição****
7	0,1429	0	0
Hierarquia**	**Tamanho**	**Arestas**	**Densidade**
0,0963	8	28	0,5
Diâmetro	**Autor da mudança***	**Linhas adicionadas**	**Linhas removidas**
1	J. Daniel Kulp	0	1
Code Churn	**Idade do arquivo**		
1	141		

* Métricas qualitativas que são geradas pela ferramenta, mas não foram usadas devido às limitações do algoritmo de classificação, cujo problema é descrito com mais detalhes adiante no Capítulo 4.
** Para métricas de centralidade da rede e buracos estruturais, foi calculada a mediana dos valores, calculados para cada nó do grafo.

Fonte: Autoria própria.

los de predição, de forma que cada mudança conjunta ocorrida do artefato em análise tenha um modelo.

Para construir um modelo de predição, é utilizada a técnica denominada Floresta Randômica (do inglês *Random Forest*). Essa técnica foi utilizada em um estudo preliminar (WIESE et al., 2015c) e apresentou resultados satisfatórios. A Floresta Randômica é um algoritmo de classificação para realizar as predições, e nesse trabalho é utilizada para a predição de mudanças conjuntas entre artefatos. O processo de classificação é feito com apoio da ferramenta denominada R[10], um software livre de estatística. Mais especificamente, é utilizado o pacote `randomForest`[11] que fornece funções para classificação com o algoritmo Floresta Randômica, em conjunto com o pacote Caret[12] que fornece funções padronizadas para predição.

[10]Disponível em: `https://www.r-project.org/`
[11]Disponível em: https://cran.r-project.org/web/packages/randomForest/
[12]Disponível em: http://caret.r-forge.r-project.org

3.3.8 Criar conjunto de métricas com base no novo *commit*

Com o modelo de predição construído, é possível classificar novos conjuntos de métricas, ou seja, instâncias sem rótulos. No contexto desse trabalho, classificar significa dizer se mudanças conjuntas (entre dois artefatos) são propensas a acontecer ou não. Dessa forma, por meio de métricas de uma mudança recente de determinado artefato, pode-se inferir se outros artefatos também são propensos ou não a mudarem.

Para tanto, é gerado o conjunto de teste composto por apenas uma instância, que representa o novo *commit* mais a tarefa associada. A instância, como explicado um conjunto de métricas é criado para cada artefato modificado no novo *commit*. Esse *commit* deve estar associado a uma solicitação de mudança que não está concluída. As métricas seguem o mesmo processo que é realizado para criar o conjunto de métricas para treino, com exceção da classificação dessas métricas.

3.3.9 Aplicar regras de associação

Além da técnica de Wiese et al. (2015c), a técnica de Zimmermann et al. (2005) é automatizada para que faça a predição das mudanças conjuntas. A técnica de Zimmermann et al. (2005) consiste em aplicar as regras de associação para encontrar mudanças entre os artefatos e, então, apresentar como propensas a mudarem as n mais frequentes com base nos valores mínimos de suporte e confiança, conhecido como *Apriori* (ZIMMERMANN et al., 2005). Para tanto, são aplicadas consultas sobre cada conjunto de artefatos modificados para concluir uma solicitação de mudança onde o artefato em análise mudou.

Para técnica de Zimmermann et al. (2005), a consultas para predição realizada é denominada navegação (ZIMMERMANN et al., 2005). Considere um conjunto de artefatos Δ associados a uma solicitação de mudança da qual o artefato em análise pertence. A navegação é uma consulta que, para cada artefato $i \in \Delta$, busca um conjunto de artefatos $J = \Delta - i$ que mudaram com i.

3.3.10 Predição de mudanças conjuntas

A predição de mudança, na técnica de Wiese et al. (2015c), é o resultado da classificação de novas métricas observadas aplicados ao modelo de classificação. Assim, quando um artefato muda, é possível inferir se outro deveria ser modificado também.

De maneira informal, seria como perguntar ao modelo de predição se o artefato deve mudar ou não por causa de outro artefato. A pergunta é representada pelas métricas e a resposta seria representado pelos artefatos que devem mudar em conjunto. Na técnica de Zimmermann et al. (2005), as predições são o resultado de consultas sobre os conjuntos de artefatos. Os resultados são ordenados para apresentação na ferramenta por valores de suporte e confiança para predições da técnica de Regras de Associação, e probabilidade para predições da técnica de Classificação.

A probabilidade de o artefato mudar ou não, exibida na ferramenta, foi com base em cada técnica. Para a técnica de Regras de Associação, foi considerado como artefato provável de mudar aqueles que tivessem os maiores 10 valores de suporte e confiança. Já para a técnica de Classificação, a probabilidade foi obtida da saída da técnica de Classificação, especificamente do algoritmo Floresta Randômica. Nesse trabalho, considera-se que um artefato deveria mudar caso a probabilidade resultante do algoritmo seja maior que 50%. Onde não foi utilizada a técnica, não foi mostrada nenhuma informação de que o artefato devia ou não mudar e sua probabilidade.

Assim, ao final desse processo, obtém-se em um conjunto de artefatos que, de acordo com o modelo de predição, são propensos a mudar com determinado artefato, especificamente daquele *commit* que está associado a uma solicitação de mudança. O conjunto de artefatos propensos a mudarem é denominado como predição de mudanças conjuntas.

3.3.11 Coletar *feedback*

Por fim, as predições de mudanças conjuntas resultantes da técnica são disponibilizadas para os desenvolvedores em uma página da *Web*. É possível que os desenvolvedores opinem sobre as predições. Esse *feedback* foi coletado por meio de um campo de texto disponibilizado na página, onde os desenvolvedores poderão descrever sua opinião sobre as predições de mudanças conjuntas, em especial as falso-positivas.

Para os desenvolvedores, as predições de mudanças conjuntas dadas pela ferramenta podem ajudá-los a realizarem uma mudança no software, alertando-os sobre possíveis impactos em outras partes do software devido à mudança realizada. Isso é particularmente interessante aos desenvolvedores iniciantes no projeto, por exemplo, em projetos de código aberto, onde geralmente eles não possuem um conhecimento suficiente para completar uma solicitação de mudança. Outro benefício que as predições de mudanças conjuntas podem trazer é evitar a introdução de defeitos devido a uma mu-

dança incompleta. No entanto, caberá ao desenvolvedor decidir se os artefatos preditos foram realmente afetados por uma mudança e, portanto, se devem ser modificados.

O mecanismo de *feedback* será utilizado em trabalhos futuros, podendo ajudar a entender melhor as predições de mudanças conjuntas, em especial as falso-positivas. Com base nessas informações, poderão ser levantados os principais pontos da técnica que podem ser ajustados para obter um resultado melhor para predição de mudanças conjuntas, ou seja, com menos predições falso-positivas. Além disso, esse processo de melhoria da técnica pode ser realizado de forma contínua.

3.4 DIFICULDADES DE IMPLEMENTAÇÃO DA FERRAMENTA RECOMINER

Durante o desenvolvimento da ferramenta Recominer, diversos desafios foram enfrentados. Inicialmente, o desafio foi a integração entre as diferentes ferramentas desenvolvidos em diferentes linguagens. Tais ferramentas não fornecem ou não são um *framework* e nem fornecem um meio de integração simples com outras linguagens. Além disso, o projeto e arquitetura de uma ferramenta que integre a outras ferramentas de forma que seja extensível e manutenível com conceitos da orientação a objetos é um desafio de todo software de integração. A extensibilidade e manutenibilidade são algumas das características buscadas constantemente em softwares.

Dentre outros desafios, pode-se destacar o tratamento dos dados a serem usados pelas técnicas e o cálculo das métricas de diferentes contextos. Esses dois processos citados, em especial, demandam mais recurso computacional a medida em que a quantidade de dados a serem processadas aumenta. Devido à limitação de recurso computacional existente, muitos desses processos (algoritmos), bem como o armazenamento dos dados no banco de dados, foram reformulados para que fossem otimizados em relação ao desempenho.

Em relação aos dados, o desafio é obter os dados do modelo de banco de dados relacional, criado pelas ferramentas de extração de dados dos repositórios de software, o Bicho e o CVSAnalY. Obter os dados relacionados e organizados da forma necessária para o cálculo de métricas e execução das técnicas de predição foi um desafio, já que foi preciso entender como os dados coletados se relacionavam. Tecnicamente, para obter e organizar os dados relacionados, consultas com SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) precisaram ser criadas. Além disso, essas consultas passaram por melhorias para otimizar, pois inicialmente houve diversos

problemas devido ao baixo desempenho dessas consultas.

Outro desafio é em relação ao teste e à validação dos dados utilizados. Isso ocorre devido à grande quantidade de dados que são processados e gerados a partir dos dados coletados dos repositórios de software. Garantir que todos as informações e dados estão corretos continua sendo um desafio.

O desenvolvimento da interface utilizada na ferramenta também foi um desafio. O objetivo é que a interface da ferramenta mostrasse as informações das tarefas e das predições de maneira simples para facilitar o uso pelos usuários, sem onerar a execução do trabalho dos usuários. Embora ainda possa ser melhorada, essa interface passou por melhorias ao longo do desenvolvimento. Isso foi possível com opiniões e sugestões de outros pesquisadores e potenciais usuários. É importante ressaltar que nenhum experimento formal foi realizado com o intuito de avaliar e melhorar a usabilidade da interface da ferramenta Recominer, devido às limitações de tempo e voluntários, além de sair do escopo desse trabalho.

3.5 LIMITAÇÕES DA FERRAMENTA

A ferramenta foi desenvolvida para solicitações de mudanças do tipo corretivo, ou seja, tarefas de manutenção de software, pois não é possível predizer artefatos novos. Além disso, a ferramenta não suporta cenários onde não há um artefato inicial. A premissa para que a ferramenta faça as predições de mudanças conjuntas é que um artefato seja modificado.

Outras limitações são mais técnicas e relacionadas às tecnologias de apoio para que a ferramenta fosse desenvolvida. Uma delas é quanto ao banco de dados para qual a ferramenta foi projetada. Para essa ferramenta, optou-se pelo banco de dados relacional MySQL. Outras ferramentas que são utilizadas são o Bicho e o CVSAnalY, utilizadas para coleta de informações dos repositórios de software, e que também estão limitadas a banco de dados relacionais e a certos repositórios de software. O Bicho suporta os seguintes sistemas de gerenciamento de solicitações de mudanças: Bugzilla, GitHub, JIRA, Sourceforge, Launchpad, Maniphest, Redmine, Gerrit, Allura, Google Code, Trac. O CVSAnaly suporta os seguintes sistema de controle de versão: CVS (*Concurrent Version System*), SVN (*Subversion*) e Git.

Essas limitações podem ser exploradas em trabalhos futuros, estendendo e aperfeiçoando a ferramenta.

3.6 CONSIDERAÇÕES FINAIS

Nesse capítulo, foi apresentado o funcionamento da ferramenta em forma de processo dividido em subprocessos de forma detalhada. Em suma, a ferramenta coleta os dados dos repositórios de software, faz tratativas com os dados, aplica a técnica de Regras de Associação e a técnica de Classificação para predizer artefatos e apresenta os artefatos propensos a mudarem em conjunto com outro. As predições podem receber um *feedback* do usuário, seja um comentário ou a escolha dos artefatos que o usuário mudaria para completar a solicitação de mudanças.

4 EXPERIMENTO CONDUZIDO COM A FERRAMENTA RECOMINER

4.1 CONSIDERAÇÕES INICIAIS

Diversas ferramentas são criadas para apoiar na realização de determinado trabalho, mas com baixa aceitação e, consequentemente, baixa adoção pelo público-alvo. A baixa aceitação pode ser causada por diversos fatores como a facilidade de uso e utilidade da ferramenta. Dessa forma, é importante avaliar a aceitação da ferramenta pelos potenciais usuários, a fim de predizer e explicar o uso da ferramenta (DAVIS, 1989). Então, a fim de avaliar a aceitação da ferramenta Recominer, foi realizada uma avaliação da aceitação com um cenário prático de manutenção de software com potenciais usuários utilizando o Modelo de Aceitação de Tecnologia (TAM), explicado com mais detalhes adiante. Além disso, foi avaliada a autoeficácia dos participantes por meio de um questionário. A autoeficácia é uma medida da capacidade percebida pelo próprio participante para realizar uma tarefa (BANDURA, 2002). Também foram coletados o tempo, além dos artefatos indicados pelo participante, os quais foram usados para medir o desempenho em relação a eficácia e eficiência em realizar uma tarefa de manutenção de software.

4.2 CONFIGURAÇÃO DO EXPERIMENTO

Para o experimento, foram selecionadas solicitações de mudanças, classificadas como correção, de um projeto de software livre, o que caracteriza um cenário de manutenção de software, para que a ferramenta pudesse predizer possíveis mudanças conjuntas para completar esse tipo de tarefa[1]. Além disso, foi necessária a participação de pessoas com o perfil de potenciais usuários da ferramenta. Um dos perfis de potenciais usuários são os novatos em projetos de código aberto, pois enfrentam diversas dificuldades em colaborar (STEINMACHER, 2015). Além disso, um dos maiores desa-

[1]Para simplificar e facilitar o entendimento, nesse Capítulo o termo “solicitação de mudanças” foi substituído por “tarefa”.

fios em projetos de software livre é prover suporte aos novatos (STEINMACHER, 2015). Nesse trabalho, os potenciais usuários da ferramenta convidados para participar do experimento foram alunos do curso de Bacharelado em Ciência da Computação (BCC) da Universidade Tecnológica Federal do Paraná (UTFPR) Campus Campo Mourão, onde também foi o local do experimento.

O objetivo dos participantes no experimento foi indicar artefatos que mudariam para corrigir defeitos reportados utilizando a ferramenta como apoio. A correção do defeito não foi obrigatória, ou seja, não fazia parte do experimento ser codificada a correção, já que a ferramenta Recominer ajuda mostrando os artefatos mais propensos a mudarem no contexto daquela tarefa, mas não partes específicas, por exemplo linhas e métodos, do artefato de código-fonte. Além disso, o tempo para realizar a tarefa não foi limitado para que cada participante pudesse realizá-la de maneira confortável, assim como quando desenvolvedores colaboram em softwares livres.

Em relação às tarefas, foram escolhidas tarefas do projeto CXF da comunidade da *Apache Software Foundations*[2]. O Apache CXF[3] é um *framework* de código aberto desenvolvimento, em sua grande parte, na linguagem Java. Seu objetivo é ajudar no desenvolvimento de Serviços Web com diversos protocolos (SOAP, XML/HTTP, RESTful HTTP ou CORBA) que funcionam sobre uma variedade de transporte (HTTP, JMS ou JBI). Esse projeto já foi utilizado em pesquisas anteriores do grupo de pesquisa, do qual o autor participou, para avaliar a metodologia de predição de mudanças conjuntas entre artefatos com Classificação (WIESE et al., 2017).

Desse projeto foram selecionadas apenas tarefas de correção já concluídas (classificadas como *Bug*), ou seja, defeitos que já tivessem sido corrigidos. Isso foi necessário para avaliar a taxa de acerto e erro dos participantes com o uso de diferentes técnicas implementadas na ferramenta Recominer. No entanto, é importante ressaltar que a ferramenta é capaz de predizer mudanças conjuntas a partir de qualquer modificação relacionada a pelo menos uma tarefa, conforme descrito na Seção 3.3.

Das tarefas concluídas, foram selecionadas aquelas mais simples, com poucos artefatos e poucas linhas alteradas, de forma que colaboradores novatos, inexperientes com o projeto, pudessem participar e contribuir com o projeto em um tempo hábil, diminuindo a interferência do cansaço, além de diminuir o risco da escolha incorreta de artefatos a serem modificados. Foi definido como tarefas simples aquelas que possuís-

[2]https://www.apache.org/
[3]http://cxf.apache.org/

sem entre três a cinco artefatos modificados, com um total de linhas modificadas entre 10 e 50. Dessa forma, de todas as tarefas que atendessem esses critérios, três tarefas foram selecionadas para o experimento: CXF-3742, CXF-4220 e CXF-4872. A seguir, segue título e descrição das tarefas apresentadas na ferramenta, com exceção da Tarefa 2, cuja descrição foi parcialmente omitida. Mais detalhes dessas tarefas são apresentados no Apêndice C.

- **Tarefa 1: CXF-3742**. **Título**: *"Too many startServer() and clientDestroyed() got invoked when spring created and java code created client/server mixed used [sic]"*. **Descrição**: *"From the testcase, there are 2 servers(one is spring created,the other is code created) and 2 clients(one is sprint created,the other is code created), when install/uninstall this bundle to osgi container, startServer() got invoked 3 times and clientDestroyed() got invoked 4 times! [sic]"*.
- **Tarefa 2: CXF-4220**. **Título**: *"After loading XSDs from links in WADL, JAX-RS get all for all resources fail with 400 [sic]"*. **Descrição**: *"We use a CustomCXFNonSpring-JaxrsServlet with one overriding method in order to use our hand written WADL. [...] [sic]"*.
- **Tarefa 3: CXF-4872**. **Título**: *"JSONProvider can not drop a root element if DOM Document is copied to JSON [sic]"*. **Descrição**: Sem descrição.

Os artefatos modificados e a quantidade de linhas adicionadas e removidas nas tarefas CXF-3742, CXF-4220 e CXF-4872 são apresentadas na Tabela 4. O artefato inicial significa que o artefato foi indicado na ferramenta para o participante como alteração inicial para que a ferramenta realizasse a predição de outros artefatos que mudariam em conjunto. Os artefatos alterados são a solução da tarefa e, assim, é esperado que os participantes os indiquem. Apesar de existir muitas formas de se completar uma tarefa de manutenção, foi considerada correta aquela solução que havia sido executada. Todas as tarefas possuíam um artefato inicial, que foi selecionado aleatoriamente e é necessário para as técnicas de predição, além de outros dois artefatos que foram alterados de fato para corrigir o defeito reportado.

Essas tarefas selecionadas foram utilizadas na ferramenta, que por sua vez fez a predição de mudanças conjunta com três técnicas diferentes para cada uma das tarefas. As técnicas de predição utilizadas foram aquelas implementadas na ferramenta: (a) utilizando a técnica de Regras de Associação proposta por Zimmermann et al. (2005), (b)

Tabela 4: Artefatos alterados em cada tarefa e suas respectivas quantidade de linhas alteradas.

Tarefa		Artefato	Linhas alteradas	
CXF-3742	**Artefato inicial**	common/common/ src/main/java/org/apache/cxf/ configuration/spring/ AbstractFactoryBeanDefinitionParser.java	+7	-0
	Artefatos alterados	rt/frontend/jaxws/ src/main/java/org/apache/cxf/ jaxws/spring/ JaxWsProxyFactoryBeanDefinitionParser.java	+3	-0
		rt/frontend/simple/ src/main/java/org/apache/cxf/ frontend/spring/ ClientProxyFactoryBeanDefinitionParser.java	+3	-1
CXF-4220	**Artefato inicial**	rt/frontend/jaxrs/ src/main/java/org/apache/cxf/ jaxrs/model/wadl/ WadlGenerator.java	+6	-3
	Artefatos alterados	rt/frontend/jaxrs/ src/main/java/org/apache/cxf/ jaxrs/servlet/ CXFNonSpringJaxrsServlet.java	+10	-1
		systests/jaxrs/ src/test/java/org/apache/cxf/ systest/jaxrs/ JAXRSClientServerSpringBookTest.java	+2	-1
CXF-4872	**Artefato inicial**	rt/rs/extensions/providers/ src/main/java/org/apache/cxf/ jaxrs/provider/json/ JSONProvider.java	+1	-1
	Artefatos alterados	rt/rs/extensions/providers/ src/main/java/org/apache/cxf/ jaxrs/provider/json/utils/ JSONUtils.java	+7	-8
		rt/rs/extensions/providers/ src/test/java/org/apache/cxf/ jaxrs/provider/json/ JSONProviderTest.java	+16	-1

Fonte: Autoria própria.

utilizando a técnica de Classificação proposta por Wiese et al. (2015c) e (c) sem a utilização de uma técnica, apresentando todos os artefatos para simular que a ferramenta e as técnicas não estivessem sendo utilizadas.

Para realizar as predições com as técnicas na ferramenta, foi utilizado todo o histórico de tarefas e versionamento do projeto até a data de correção da tarefa. As predições de cada tarefa e cada técnica são apresentadas com mais detalhes no Apêndice D. Em relação às métricas para a técnica de classificação, foram removidas todas variáveis qualitativas, por exemplo, o tipo da tarefa, a prioridade da tarefa, o responsável pela tarefa e o autor das mudanças. Isso foi necessário devido à limitação do algoritmo de classificação, que não consegue classificar uma nova instância caso o valor de uma variável qualitativa seja nova, inexistente no treino. Dessa forma, toda vez que houver um caso onde uma nova instância a ser classificada conter um novo valor que não existe no conjunto de treino, o classificador não consegue classificar. Para exemplificar sua limitação, suponha que exista um conjunto de treino onde a variável qualitativa V tenha os valores A, B e C. Com o modelo criado a partir do conjunto de treino citado anteriormente, não será possível classificar caso uma nova instância possua o valor de $V = D$.

Participaram do experimento 15 alunos que foram divididos aleatoriamente em 3 grupos. Esses 3 grupos receberam as mesmas 3 tarefas. No entanto, para evitar um viés no experimento, as técnicas utilizadas por cada grupo foram organizadas de forma que não se repetissem, usando a técnica do quadrado latino. O quadrado latino é uma matriz quadrada de itens que são organizados de forma que a ordem dos itens não se repita em cada linha e cada coluna. Dessa forma, para esse experimento, variou-se a técnica a ser utilizada por determinado grupo em determinada tarefa, de forma que (a) nenhuma técnica se repita numa tarefa entre os grupos, (b) a ordem das técnicas seja totalmente diferente para todos os grupos e (c) todas as técnicas sejam utilizadas por todos os grupos.

O Quadro 8 mostra como ficou configurado o experimento. Na primeira tarefa, o Grupo 1 utilizou a técnica de Regras de Associação, o Grupo 2 utilizou a técnica de Classificação e o Grupo 3 utilizou a lista de artefatos. Na segunda tarefa, o Grupo 1 utilizou a técnica de Classificação, o Grupo 2 utilizou a lista de artefatos e o Grupo 3 utilizou a técnica de Regras de Associação. Por fim, na terceira tarefa, o Grupo 1 utilizou a lista de artefatos, o Grupo 2 utilizou a técnica de Regras de Associação e o Grupo 3 utilizou a técnica de Classificação.

Antes do experimento, foi feita uma breve apresentação geral da ferramenta e foi demonstrado como a ferramenta deveria ser usada no experimento. Também foi explicado o objetivo dos participantes no experimento, bem como o objetivo do pró-

Quadro 8: Distribuição de técnicas por grupo e tarefa utilizando a técnica do quadrado latino

Tarefa	Grupo 1	Grupo 2	Grupo 3
Tarefa 1 CXF-3742 (T1)	Regras de Associação (AR)	Classificação (ML)	Lista de todos artefatos (ALL)
Tarefa 2 CXF-4220 (T2)	Classificação (ML)	Lista de todos artefatos (ALL)	Regras de Associação (AR)
Tarefa 3 CXF-4872 (T3)	Lista de todos artefatos (ALL)	Regras de Associação (AR)	Classificação (ML)

Fonte: Autoria própria.

prio experimento. É importante ressaltar que nenhum participante tinha conhecimento prévio da ferramenta nem das técnicas utilizadas para predição de mudanças conjuntas entre artefatos.

Para o experimento, foram disponibilizados o código-fonte do Apache CXF referente à tarefa, incluindo a resolução parcial da tarefa com um artefato já alterado com a correção, já que um artefato foi escolhido como partida inicial da tarefa. O código-fonte foi disponibilizado com a IDE Eclipse.

Durante o experimento, foi permitido ao participante utilizar recursos de pesquisa e tradução na Internet, com exceção da consulta no sistema de gerenciamento de tarefas do próprio projeto, já que nela contém a solução do defeito. Além disso, os participantes puderam utilizar recursos do próprio IDE e do Sistema Operacional, por exemplo, para procurar por algum artefato desejado. Orientou-se também para que a comunicação sobre as tarefas entre os participantes não fosse feita. Também optou-se por monitorar a atividade do participante por meio da gravação da tela para consulta posterior caso necessário a análise de alguma situação ocorrida quanto aos resultados.

Antes dos participantes iniciarem as tarefas, foi solicitado que respondessem um questionário para identificar características quanto ao perfil deles. O perfil dos participantes é apresentado e discutido na Seção 4.3. Também foi solicitado para os participantes responderem o questionário de autoeficácia logo antes de iniciarem a primeira tarefa e logo após o término de cada tarefa. O questionário permite medir a autoeficácia, ou seja, a capacidade percebida do próprio participante para realizar uma tarefa (BANDURA, 2002). E, por fim, foi solicitado aos participantes responderem o questionário do Modelo de Aceitação de Tecnologia (TAM, do inglês *Technology Acceptance Model*) ao fim de cada tarefa. O TAM avalia a percepção de Utilidade e Facilidade de Uso do usuário sobre uma tecnologia, dois fatores básicos que determinam a acei-

tação da tecnologia pelo usuário (LAITENBERGER; DREYER, 1998). Os questionários de autoeficácia e TAM são apresentados na Seção 4.4. Os resultados do TAM são discutidos de forma geral, na Subseção 4.5.2, e por grupo, na Subseção 4.5.3. O resultado da autoeficácia e do desempenho são discutidos na Subseção 4.5.4 e Subseção 4.5.5, respectivamente.

4.3 PERFIL DOS PARTICIPANTES

Foi aplicado um questionário com relação ao perfil dos participantes do experimento logo no início, antes deles iniciarem as atividades do experimento. O objetivo do questionário foi identificar o participante e suas experiências acadêmicas e profissionais com desenvolvimento e manutenção de software livre que pudessem influenciar de alguma forma no experimento. Essas informações podem ajudar a explicar resultados discrepantes que eventualmente pudessem ter ocorrido. As nove características do perfil obtidas dos participantes são apresentadas na Tabela 5, bem como a resposta de cada participante para que seja possível cruzar com outros resultados.

Dos participantes, apenas 2 dos 15 já tinham experiência na indústria de software, enquanto os demais participantes declararam-se sem experiência. Quanto a autoavaliação como desenvolvedor, 10 se autoavaliaram como intermediário, ou seja, um valor de 3, numa escala de 1 - iniciante a 5 - avançado. Outros 2 se consideraram em um nível entre avançado e intermediário (escala 4). Dos outros 3 restantes, 1 se avaliou como iniciante (escala 1), outro entre o iniciante e intermediário (escala 2), e outro se avaliou como um desenvolvedor avançado (5).

A maioria dos participantes estão a, no máximo, 4 semestres (2 anos) de concluir o curso de Bacharelado em Ciência da Computação (BCC), exceto por um único participante que está a mais de 4 semestres. Desses 14 participantes, 4 estavam a 1 semestre de concluir, 4 estavam a 2 semestres de concluir, 1 estava a 3 semestres de concluir e outros 5 estavam a 4 semestres. Dentro do curso de BCC, 13 participantes já cursaram alguma disciplina que discutisse sobre a contribuição em softwares livres. Além disso, dos 15 participantes, 7 já tentaram contribuir para algum projeto de software livre cujas contribuições não foram aceitas. Outros 2 participantes já tiveram entre 2 a 10 contribuições aceitas e 1 participante teve 1 contribuição aceita. Essa experiência pode ajudar os participantes, já que eles possuem certo conhecimento sobre as dificuldades de se contribuir em um projeto de software livre. Em contrapartida, 5 participantes nunca contribuíram, o que também pode ter influência na realização das

tarefas do experimento desse trabalho.

Em relação ao tipo de tarefas que os participantes já contribuíram ou tentaram, 11 participantes lidaram com tarefas de correção de defeito, 6 participantes com atualizar ou escrever uma nova documentação e 3 participantes com traduções. Outros 2 participantes declararam que nunca contribuíram. É interessante notar que nenhum participante tentou contribuir com uma implementação de uma funcionalidade inédita em algum projeto de software livre. No entanto, é importante ressaltar que no experimento desse trabalho, foram selecionadas apenas tarefas de manutenção de software, já que a ferramenta é voltada para manutenção do software.

Houve uma divergência entre as questões de contribuições aceitas e o tipo de contribuição feita. Na questão relacionada ao tipo de contribuição, 2 participantes declararam nunca ter contribuído, enquanto 5 participantes declararam nunca ter contribuído na questão de contribuições aceitas.

Todos os participantes possuem alguma experiência acadêmica ou profissional com a linguagem de programação Java. Esse perfil é importante pois pode influenciar esse experimento, já que as tarefas desse experimento consistiam em corrigir um defeito de um software desenvolvido em sua grande parte na linguagem Java. Além disso, outros 11 participantes tinham experiência com C/C++, 9 participantes com Python, 6 com Javascript, 3 com Ruby on Rails e 3 com outras linguagens.

Outro fator importante é a experiência com o ambiente integrado de desenvolvimento, ou IDE (do inglês *Integrated Development Environment*), o qual foi utilizada no experimento para apoio na realização das tarefas. Dentre todos os participantes, 13 já haviam utilizado o Eclipse, que foi utilizada para esse experimento, já que é um software livre e uma das IDE disponíveis para desenvolvimento na plataforma Java.

Tabela 5: Perfil dos participantes

Variável	Valores	Quantidade	%	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10	P11	P12	P13	P14	P15
Experiência com desenvolvimento de software para a indústria	Sim	2	13,3							X	X							
	Não	13	86,7	X	X	X	X	X	X			X	X	X	X	X	X	X
Autoavaliação como desenvolvedor	1 (iniciante)	1	6,7	X														
	2	1	6,7														X	
	3	10	66,7			X	X	X	X		X	X	X	X		X		X
	4	2	13,3							X					X			
	5 (avançado)	1	6,7		X													
Semestres para conclusão do curso	1	4	26,7					X			X			X	X			
	2	4	26,7		X				X	X			X					
	3	1	6,7															X
	4	5	33,3	X		X	X									X	X	
	5 ou mais	1	6,7									X						
Cursou alguma disciplina a respeito de software livre	Sim	13	86,7		X	X	X	X	X	X	X	X	X	X	X	X		X
	Não	2	13,3	X													X	
Contribuições aceitas em software livre	1	1	6,7										X					
	2 a 10	2	13,3		X					X								
	11 ou mais	0	0,0															
	Já tentei, mas nunca aceitaram a contribuição	7	46,7			X	X	X	X						X	X		X
	Nunca contribuí	5	33,3	X							X	X		X			X	
Tipo de contribuição, inclusive os recusados	Corrigir um defeito	10	66,7		X	X	X	X	X	X			X	X		X		X
	Implementar uma nova funcionalidade	0	0,0															
	Atualizar ou escrever nova documentação	6	40,0		X	X		X	X	X					X			
	Fazer traduções	3	20,0		X						X					X		
	Nenhum	3	20,0	X								X					X	
Experiência com linguagens de programação na academia ou indústria	Java	15	100	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	Python	9	60,0		X	X			X	X	X		X	X	X	X		
	C/C++	11	73,3	X	X		X		X	X		X	X	X	X	X	X	
	Javascript	6	40,0		X			X		X			X		X	X		
	Ruby on Rails	3	20,0		X	X	X											
	Outros	3	20,0				X	X							X			
Experiência com o Eclipse	Sim	13	86,7		X	X	X	X	X	X	X	X	X	X	X	X	X	
	Não	2	13,3	X														X
Experiência com o Apache CXF (contribuição ou utilização)	Sim	0	0															
	Não	15	100	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Fonte: Autoria própria.

É importante notar que nenhum participante declarou já ter usado ou colaborado com tal projeto de software livre. Isso significa que os participantes se enquadram no perfil de colaboradores novatos. Além disso, a contribuição ou utilização prévia do Apache CXF pelos participantes poderia ajudá-los na realização da tarefa no experimento e, assim, afetar o resultado.

4.4 AVALIAÇÃO DA ACEITAÇÃO DA FERRAMENTA

Para avaliar a aceitação e intenção de uso da ferramenta desenvolvida, foi aplicado aos participantes um questionário elaborado com base no Modelo de Aceitação de Tecnologia (TAM, do inglês *Technology Acceptance Model*), que já foi utilizado por diversos estudos para avaliar a aceitação de sistemas de informação (MARANGUNIĆ NIKOLA E GRANIĆ, 2015). O TAM é uma extensão da Teoria da Ação Racional (TRA, do inglês *Theory of Reasoned Action*) (FISHBEIN; AJZEN, 1975), utilizada para predizer o comportamento humano. O TAM foi proposto por Davis (1986) especificamente para o contexto de sistemas de informação, cujo objetivo é medir a aceitação pelos potenciais usuários de alguma tecnologia.

Dentre diversas variáveis que podem influenciar na adoção de uma ferramenta, existem duas variáveis determinantes que foram destacadas por pesquisas anteriores e são capturadas pela primeira versão do TAM (DAVIS, 1989). Nessa versão, as duas variáveis que compõem o modelo são a Utilidade Percebida (U) e a Facilidade de Uso Percebida (E) (DAVIS, 1986, 1989). A Utilidade Percebida (U) mede o grau em que o usuário acredita que a tecnologia aumenta o seu desempenho na realização de determinado trabalho e a Facilidade de Uso Percebida (E) mede o quão livre de esforço a ferramenta é (DAVIS, 1986, 1989). Apesar de existirem outras variáveis, a Utilidade Percebida (U) e a Facilidade de Uso Percebida (E) provavelmente desempenham um papel central, pois formam dois constructos distintos e são indicadas como fundamentais nas decisões de utilização de tecnologia da informação (DAVIS, 1989).

O TAM é composto por questões que capturam esses constructos, a Utilidade Percebida e a Facilidade de Uso Percebida, em relação ao uso da tecnologia. Além desses, também foi capturada a intenção do participante em adotar a ferramenta com as questões do fator de Uso no Futuro (SP), que geralmente é influenciado pela Utilidade Percebida e a Facilidade de Uso Percebida. Nesse trabalho, esses três constructos foram capturados pelas questões adaptadas do TAM com respostas na escala de Likert com 5 níveis, cujas opções são: “Discordo Totalmente”, “Discordo Parcialmente”, “Neutro”,

"Discordo Parcialmente" e "Concordo Totalmente". Tais questões são apresentadas no Quadro 9 e o questionário que foi aplicado aos participantes pode ser consultado no Apêndice A.

Quadro 9: Questões do TAM.

Utilidade Percebida (U) (do inglês *Perceived Usefulness*)	
U1	A ferramenta RECOMINER me fez encontrar mais RAPIDAMENTE os arquivos que devem ser modificados.
U2	Usar a ferramenta RECOMINER melhorou meu DESEMPENHO com o objetivo de encontrar arquivos que devem ser alterados juntos.
U3	Usar a ferramenta RECOMINER melhorou minha PRODUTIVIDADE para encontrar arquivos que mudam em conjunto.
U4	A utilização da ferramenta RECOMINER melhorou a EFETIVIDADE de encontrar arquivos que mudam em conjunto.
U5	Usar a ferramenta RECOMINER FACILITOU a documentação e gestão de casos de teste.
U6	A ferramenta RECOMINER foi ÚTIL para completar a tarefa proposta.
Facilidade de Uso (E) (do inglês *Perceived Ease of Use*)	
E1	Aprender a usar a RECOMINER foi fácil para mim.
E2	Eu acho que é fácil de obter as informações necessárias para encontrar os arquivos que precisam ser modificados para completar uma tarefa usando a RECOMINER.
E3	Minha interação com a RECOMINER é clara e compreensível.
E4	É fácil me tornar hábil (aprender a usar) a RECOMINER.
E5	É fácil lembrar como usar a RECOMINER para encontrar os artefatos que devem ser modificados em uma tarefa.
E6	Considero a RECOMINER fácil de usar.
Uso no Futuro (SP) (do inglês *Self-prediction of Future Use*)	
SP1	Supondo que a RECOMINER estivesse disponível para qualquer projeto, eu a usaria no futuro.
SP2	Eu prefiro usar a RECOMINER para encontrar os artefatos que devem ser modificados ao invés do "procurar" da IDE ou a função de *debug*.

Fonte: Autoria própria.

Foi avaliada também a influência da ferramenta sobre a confiança dos participantes na realização das tarefas que lhes foram atribuídas. Para tanto, foi aplicado um questionário de autoeficácia aos participantes, baseado em outros estudos da literatura que também o aplicaram (STEINMACHER et al., 2015; DAVIDSON et al., 2014). A autoeficácia é a medida da percepção da capacidade do usuário em realizar determinada tarefa, que pode afetar a capacidade atual do usuário em completar essa tarefa (BANDURA, 1986).

O questionário de autoeficácia aplicado era composto por 4 perguntas relacionadas a realização de tarefas de manutenção do software, apresentadas no Quadro 10.

As questões foram elaboradas com base em estudos anteriores que aplicaram a avaliação de autoeficácia no contexto de colaboradores novatos (STEINMACHER et al., 2016; DAVIDSON et al., 2014). Essas questões dizem respeito ao entendimento da tarefa (S1), a realização da tarefa (S2, S3) e a confiança na realização da tarefa (S4). O questionário de autoeficácia foi aplicado quatro vezes: antes de iniciarem a primeira tarefa (Tarefa 1) e logo depois da conclusão de cada tarefa (Tarefa 1, Tarefa 2 e Tarefa 3). Isso possibilita medir e comparar a confiança inicial do participante e como cada tarefa e a ferramenta afetou sua percepção sobre a própria eficácia. O formulário aplicado aos participantes encontra-se com mais detalhes no Apêndice B.

Quadro 10: Questões de Autoeficácia (*Self-efficacy*).

Autoeficácia (*Self-efficacy*)	
S1	Eu consigo ler uma issue (tarefa) e encontrar os artefatos a serem modificados
S2	Quando altero um arquivo, eu consigo saber se outros arquivos devem ser alterados em conjunto
S3	Eu tenho boas habilidades para escrever e mudar o código-fonte
S4	Dado que um alterei um arquivo, eu me sinto confortável elencando quais são outros potenciais arquivos que devem ser alterados para evitar problemas

Fonte: Autoria própria.

4.5 RESULTADOS

Nessa seção são apresentados os resultados desse estudo com base no experimento aplicado para avaliação da ferramenta Recominer. Para tanto, primeiramente é feita uma avaliação da confiabilidade e validade do questionário TAM aplicado nesse experimento. Após isso, as respostas dos participantes nos questionários TAM e de Autoeficácia são analisadas e discutidas, além do desempenho e tempo dos participantes nas tarefas.

4.5.1 Confiabilidade e Validade do questionário TAM

A confiabilidade pode ser vista como o grau de precisão do estudo empírico. Portanto, é importante avaliar a confiabilidade e validade do instrumento de avaliação. Nesse trabalho, o instrumento de avaliação é o questionário baseado no TAM, o qual foi submetido a uma avaliação estatística. Para tanto, o alfa de Cronbach (CRONBACH, 1951) foi calculado, cujo valor considerado satisfatório na literatura é acima de $0,8$ (CARMINES; ZELLER, 1979). Embora a amostragem desse experimento seja pequena,

o valor do alfa de Cronbach foi de $0,917$ para do fator de Facilidade de Uso e $0,953$ para o fator de Utilidade, conforme mostra a Tabela 6. No entanto, o valor para o fator de Uso no Futuro foi de $0,783$, abaixo do valor satisfatório. Apesar disso, esses valores mostram que o questionário aplicado foi significativamente adequado para o experimento. O alfa de Cronbach foi calculado utilizando o SmartPLS 3[4], um software para Modelagem de Equações Estruturais (do inglês *Structural Equation Modeling*, ou SEM).

Tabela 6: Valores de Alfa de Cronbach para os fatores do TAM

Facilidade de uso (E)	Utilidade (U)	Uso futuro (SP)
0,917	0,953	0,783

Fonte: Autoria própria.

É imprescindível também validar o carregamento das questões para os fatores do modelo TAM, mostrando que elas foram adequadas para cada fator do modelo TAM, ou seja, que cada conjunto de questões formam construtos diferentes. Uma das formas que essa validação pode ser feita é realizando a análise fatorial confirmatória. Para tanto, pode-se calcular o carregamento externo, também calculado com o SmartPLS 3. Os valores da análise fatorial são apresentados na Tabela 7. Na literatura, o valor mínimo considerado aceitável é $0,7$ (LAITENBERGER; DREYER, 1998). Como pode-se observar na Tabela 7, os maiores valores para o fator Facilidade de Uso são as questões E1 a E6, com valores entre $0,755$ e $0,920$. Para fator de Utilidade, os valores vão de $0,846$ a $0,931$ com relação às questões U1 a U6. Por fim, para o fator de Uso Futuro, os valores para as questões SP1 e SP2 vão de $0,870$ até $0,909$. Com base nesse resultado, afirma-se que as questões aplicadas estão alinhadas com os fatores de Facilidade de Uso (E), Utilidade (U) e Uso no Futuro (SP) que compõe o TAM.

4.5.2 Resultado geral do TAM

A Figura 11 apresenta o resultado das questões do TAM. Como forma de comparação, as respostas de cada questão foram sumarizadas e agrupadas por técnica. Para todos os gráficos, as respostas estão representadas na escala de Likert de 5 níveis.

A primeira observação que pode-se fazer em relação a Figura 11 é a diferença na aceitação entre o uso das técnicas e sem o uso de técnicas, que pode também ser

[4]Disponível em: https://www.smartpls.com/

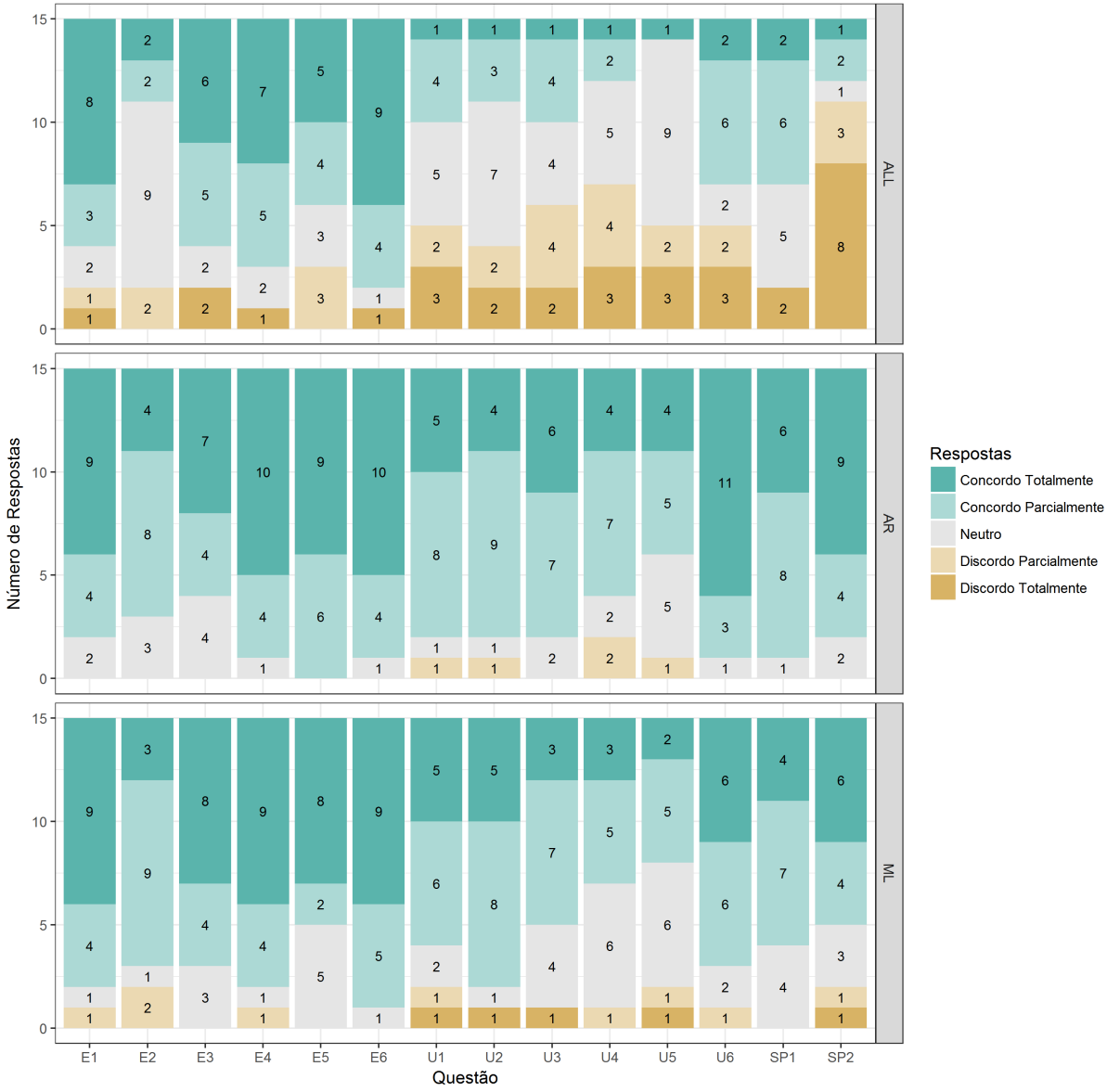
Tabela 7: Validação dos fatores do TAM com as questões.

Questão	Facilidade de Uso (E)	Utilidade (U)	Uso no Futuro (SP)
E1	**0,862**	0,680	0,373
E2	**0,755**	0,636	0,575
E3	**0,831**	0,270	0,373
E4	**0,920**	0,389	0,465
E5	**0,784**	0,360	0,417
E6	**0,870**	0,344	0,417
U1	0,492	**0,898**	0,754
U2	0,502	**0,931**	0,709
U3	0,565	**0,911**	0,707
U4	0,283	**0,878**	0,643
U5	0,351	**0,846**	0,623
U6	0,473	**0,931**	0,712
SP1	0,539	0,727	**0,909**
SP2	0,419	0,641	**0,870**

Fonte: Autoria própria.

observado nas Figuras 12, 13 e 14, apresentadas adiante na Subseção 4.5.3. Em contraste com o uso das técnicas, a maioria discordou totalmente ou parcialmente, ou ainda achou indiferente (neutro) em relação às questões de Utilidade quando não é utilizada alguma técnica (ALL). Isso denota que a ferramenta sem o uso de técnica não ajuda na tarefa de manutenção, ou ainda que pode ser pior quando a ferramenta é utilizada. Esse resultado pode ter sido refletido sobre o fator de Uso no Futuro, especificamente na questão SP2 que trata da intenção do uso da ferramenta pelo participante. Nessa questão, 8 dos 15 participantes discordaram totalmente e outros 3 discordaram parcialmente, além de uma resposta "Neutro". Isso significa que apenas 3 participantes mostraram intenção de uso no futuro da ferramenta sem o uso da técnica.

Embora tenha tido mais respostas concordando com as questões relacionadas à Facilidade de Uso da ferramenta sem o uso de alguma técnica (ALL), houve pelo menos uma discordância, total ou parcial, para cada questão de Facilidade de Uso. A discordância na questão de Facilidade de Uso era esperada, já que são muitos artefatos preditos apresentados na ferramenta. O que evidencia esse fato é o resultado da questão E2, que teve apenas 4 participantes que concordaram "que é fácil de obter as informações necessárias para encontrar os arquivos que precisam ser modificados para completar uma tarefa". Além disso, o que pode ter agravado essa situação é que a ferramenta não possuía um mecanismo de filtro para os artefatos. Apesar disso, o resultado sumarizado



ALL: Sem o uso de técnica. AR: Regras de Associação. ML: Classificação.

Figura 11: Gráfico com o resultado do TAM sumarizado dos grupos.
Fonte: Autoria própria.

do questionário TAM, no geral, mostra evidências que a ferramenta é fácil de usar.

Quanto ao fator de Utilidade, é possível notar que no geral a maioria discorda ou não encontrou diferença (“Neutro”) quando não é usada alguma técnica de predição (ALL). Esse resultado era esperado, já que a ferramenta apenas listaria todos os artefatos do software como um índice e não diminui o esforço do desenvolvedor em encontrar os artefatos que deveriam mudar para completar determinada tarefa. Isso é afirmado por um *feedback* fornecido pelo participante P08 na Tarefa 2, que diz o seguinte: *“Escolhi*

arquivos importados, mas como são muitos arquivos é difícil verificar todos [sic]". Esse *feedback* reforça que o uso de técnicas de predição de artefatos em ferramentas pode ajudar os desenvolvedores em tarefas de manutenção do software, já que diminui o número de artefatos a serem analisados.

Em casos onde os participantes usaram a ferramenta com predições realizadas por meio das técnicas de Regras de Associação (AR) e Classificação (ML), é possível observar que a maioria concordou que a ferramenta é útil e fácil de usar, ou seja, houve mais respostas positivas do que negativas quanto à Utilidade e Facilidade de Uso. Entretanto, houve casos onde não houve aceitação por parte de alguns participantes em ambas as técnicas de predição seja a Regras de Associação (AR) ou a Classificação (ML).

Observando os casos em que foi utilizado uma técnica para predição de artefatos, nota-se que, para 2 dos 15 participantes, não houve uma boa aceitação da ferramenta. Um desses casos foi utilizando a técnica de Regras de Associação com o participante P07 do Grupo 2 na Tarefa 3, que respondeu "Discordo Parcialmente" ou "Neutro" para todas as questões de Utilidade. Isso pode ter levado o participante P07 a responder "Neutro" para as questões de Uso no Futuro. O outro caso foi o participante P04 do Grupo 1 que, para a Tarefa 2 utilizando predições da técnica de Classificação, respondeu "Discordo Totalmente", "Discordo Parcialmente" ou "Neutro" para todos os fatores, Utilidade, Facilidade de Uso e Uso no Futuro.

Em relação à técnica de Regras de Associação, especificamente, é importante notar que todos os participantes responderam "Concordo Totalmente", "Concordo Parcialmente" ou "Neutro" para as questões de Uso no Futuro. Apesar disso, não houve concordância de alguns participantes para algumas das questões. Um desses casos foi o participante P07, que respondeu "Neutro" para a questão SP1 e SP2. Outro participante, P03, também respondeu "Neutro" para a questão SP2. Com exceção do participante P07 discutido anteriormente, houve uma única discordância na questão U4, respondida pelo participante P02 do Grupo 1 na Tarefa 1, utilizando a técnica de Regras de Associação. A questão U04 é relacionada a ajuda da ferramenta na efetividade em encontrar os outros artefatos que deveriam mudar para completar aquela tarefa.

Já em relação à técnica de Classificação (ML), houve uma aceitação menor em relação ao fator de Uso no Futuro se comparado com a técnica de Regras de Associação. Com a Classificação, 10 dos 15 participantes responderam "Concordo Totalmente" ou "Concordo Parcialmente", contra 13 com a técnica de Regras de Associação. Ainda para técnica de Classificação, outros 3 participantes, responderam "Neutro", e 2 responderam

"Discordo Parcialmente" e "Discordo Totalmente", contra 2 respostas "Neutro" com a técnica de Regras de Associação.

Entretanto, houve dois participantes que discordaram de algumas questões para a técnica de Classificação. Um deles foi o participante P03, na Tarefa 2 do Grupo 1, que discordou totalmente da questão U5. De maneira geral, tanto para a técnica de Classificação quanto para Regras de Associação, houve posicionamentos neutros ou discordando de que a ferramenta ajuda na documentação do software. O que pode justificar esse resultado é o objetivo da ferramenta, cujo foco é a manutenção do software e não na documentação. Além disso, pelas tarefas não serem relacionados à documentação pode ter sido um fator decisivo para essa questão. Em outras questões de Utilidade, o participante P03 respondeu "Neutro". Apesar de não concordar com nenhuma questão de Utilidade, o participante P03 concordou parcialmente com a questão SP1 e concordou totalmente com a questão SP2, indicando uma intenção de uso. O outro caso foi com o participante P06 do Grupo 2 na Tarefa 1, que discordou parcialmente com as questões U1 e E2. A questão U1 diz respeito a velocidade em encontrar os artefatos que o participante modificaria para completar a tarefa e a questão E2 diz respeito a facilidade em encontra informações para completar a tarefa. Uma justificativa dessas discordâncias seria que, na primeira tarefa, o participante não havia parâmetros para comparação, já que ele começou usando a ferramenta com predições usando a técnica de Classificação. Outro caso a se observar com a técnica de Classificação é que, embora o participante P05 não discorde com nenhuma questão de Utilidade e Facilidade e Uso, ele não mostrou uma intenção de uso no futuro, pois discordou parcialmente com a questão SP2.

A questão U5, em específico, houve poucas avaliações positivas e muitas neutras no geral. Isso se deve ao fato de que essa questão trata da documentação e caso de uso do projeto, uma vez que as tarefas utilizadas no experimento não foram relacionadas especificamente às tarefas de documentação e casos de uso.

Para aferir se as diferenças do resultado do TAM entre as técnicas foram significativas, foi aplicado o teste de Wilcoxon pareado (do inglês *Wilcoxon Signed-Rank Test*). A Tabela 8 mostra os valores-p (do inglês *p-values*) obtido do teste de Wilcoxon efetuado com apoio da ferramenta R. Os valores utilizados para esse teste foram do resultado do questionário TAM, convertido para um ordinal que varia de 1 a 5, que representa respectivamente a resposta de "Discordo Totalmente" e "Concordo Totalmente".

Considerou-se significativas os valores-p menores que $0,05$ (p-*value* $< 0,05$).

Tabela 8: Significância (*p-values*) da diferença do resultado TAM entre as técnicas.

Questão	AR x ALL	ML x ALL	ML x AR
U1	**0,0070**	0,0734	0,3951
U2	**0,0053**	**0,0423**	0,8870
U3	**0,0037**	0,0836	0,0836
U4	**0,0140**	0,0562	0,4773
U5	**0,0158**	0,2064	0,2217
U6	**0,0041**	0,1279	0,1279
E1	0,2840	0,3351	0,7728
E2	**0,0182**	0,1052	0,5201
E3	0,3471	0,1600	0,3471
E4	0,1600	0,4062	1,0000
E5	**0,0379**	0,2095	0,2095
E6	0,6093	1,0000	1,0000
SP1	**0,0106**	**0,0353**	0,0726
SP2	**0,0066**	**0,0110**	0,1870

Fonte: Autoria própria.

A comparação do resultado do TAM entre as técnicas mostrou que houve uma diferença significativa entre a técnica de Regras de Associação (AR) e sem o uso de técnica (ALL) na maioria das questões, com exceção das questões E1, E3 e E6. Já em relação a diferença entre a técnica de Classificação (ML) e sem o uso de técnica (ALL), as diferenças significativas foram nas questões U2, SP1 e SP2. É importante notar que, entre as técnicas de Regras de Associação (AR) e Classificação (ML), não houve diferenças significativas. Além disso, ressalta-se que essas duas técnicas demonstraram uma diferença significativa nas questões SP1 e SP2, que dizem respeito a intenção de uso da ferramenta no futuro pelo usuário. Essas diferenças corroboram com e reforçam as observações realizada anteriormente de que as técnicas influenciaram positivamente na intenção de uso da ferramenta, além de ajudar os desenvolvedores na manutenção do software (fator de Utilidade (U)) e ser fácil de usar (fator de Facilidade de Uso (E)).

4.5.3 Resultado por grupo do TAM

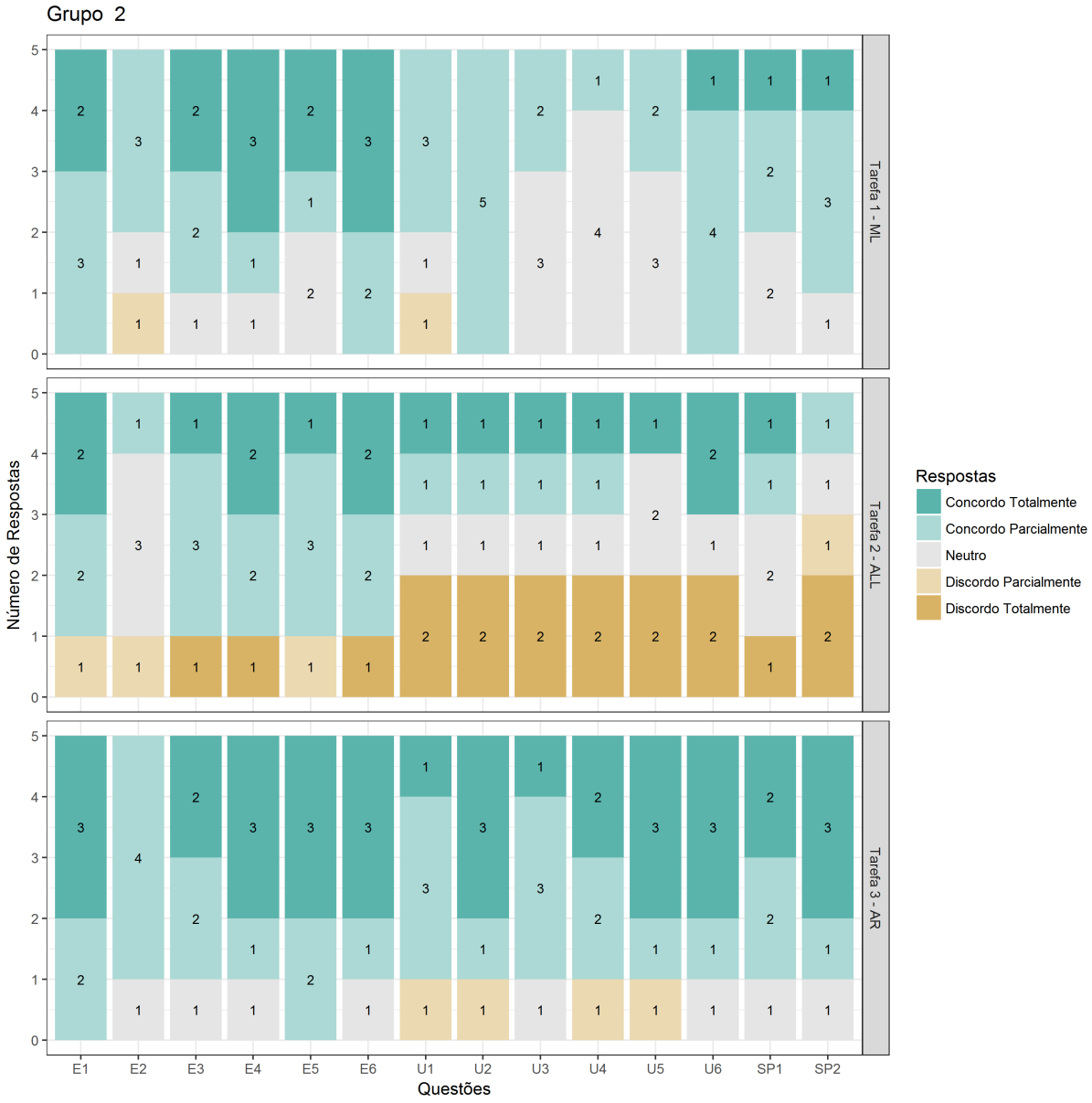
As Figuras 12, 13 e 14 representam o resultado sumarizado do Grupo 1, 2 e 3, respectivamente. Assim como na subseção anterior (Subseção 4.5.2) todos os gráficos também estão com as respostas representadas na escala de Likert de 5 níveis.

Observando as figuras dos gráficos dos resultados por Grupo, ou seja, Figuras 12, 13 e 14, é possível notar uma diferença no resultado entre os grupos. Isso pode signi-

ALL: Sem o uso de técnica. AR: Regras de Associação. ML: Classificação.

Figura 12: Gráfico com o resultado do TAM sumarizado do Grupo 1.
Fonte: Autoria própria.

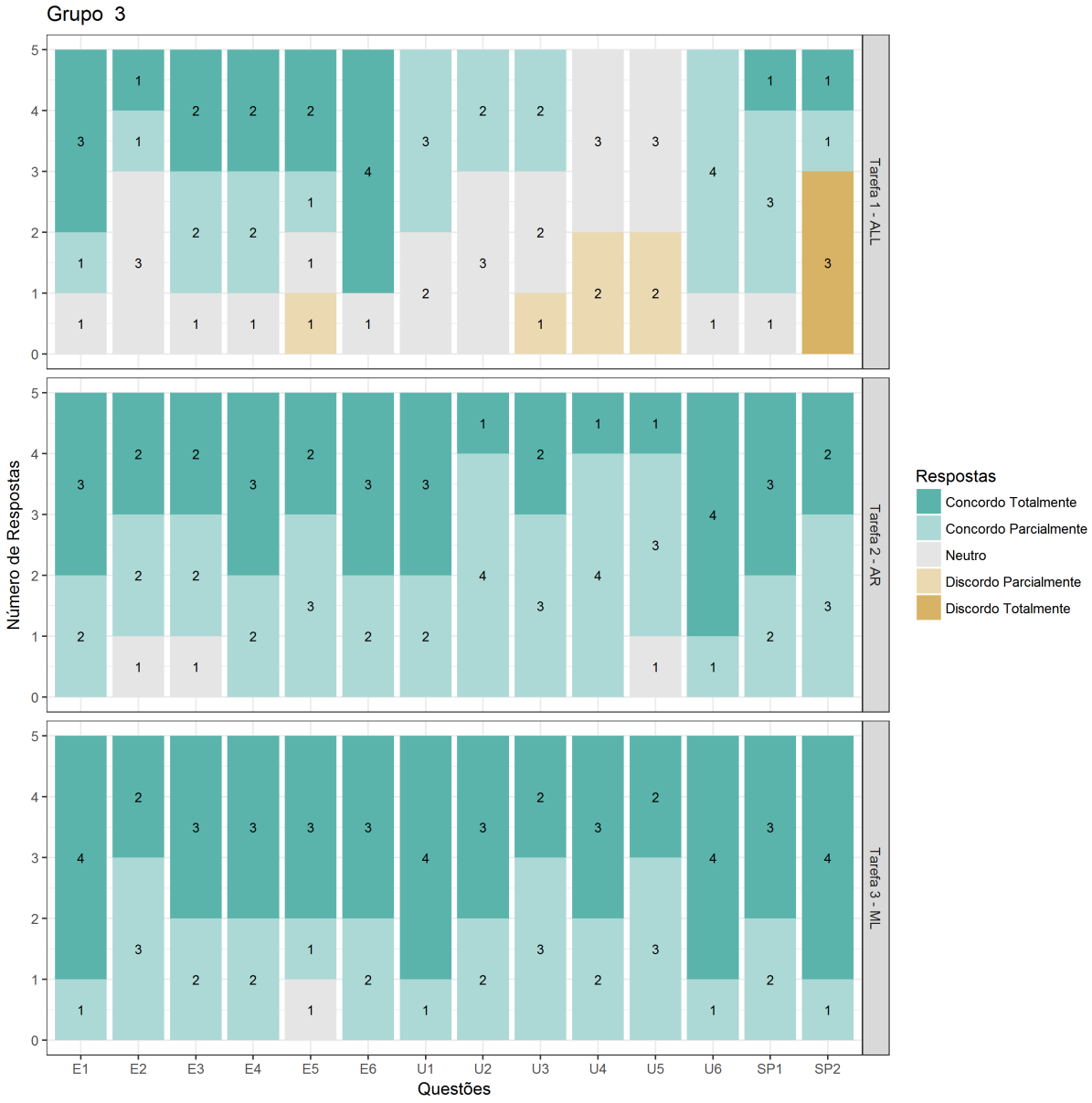
ficar que a ordem das técnicas influenciou no resultado do experimento desse trabalho. Uma das evidências disso é na iteração onde a ferramenta apresenta as predições com uso de alguma técnica, logo após não utilizar nenhuma técnica. Por exemplo, no Grupo 3, cujo resultado é apresentado na Figura 14, é possível observar uma diferença significativa na percepção da ferramenta pelo participante com a técnica de predição após o uso da ferramenta sem a técnica de predição. Na Tarefa 1, que não utilizou nenhuma técnica de predição, os participantes responderam “Neutro”, “Discordo Parcialmente” e



ALL: Sem o uso de técnica. AR: Regras de Associação. ML: Classificação.

Figura 13: Gráfico com o resultado do TAM sumarizado do Grupo 2.
Fonte: Autoria própria.

"Discordo Totalmente". Já na Tarefa 2, que utiliza a técnica de Regras de Associação, é possível notar a diferença de que não houve respostas "Discordo Parcialmente" ou "Discordo Totalmente". A maioria dos participantes do Grupo 3 responderam "Concordo Parcialmente" ou "Concordo Totalmente" para todas as questões do TAM, uma vez que as questões E2, E3 e U5 tiveram uma resposta "Neutra" cada. Por fim, na Tarefa 3, que utilizava a técnica de Classificação, o resultado demonstrou-se mais satisfatório, apesar de não ser uma diferença significativa, pois quase todos participantes do Grupo 3 con-



ALL: Sem o uso de técnica. AR: Regras de Associação. ML: Classificação.

Figura 14: Gráfico com o resultado do TAM sumarizado do Grupo 3.
Fonte: Autoria própria.

cordaram parcialmente ou totalmente com todas as questões do TAM, com exceção da questão U5, onde houve um participante que respondeu “Neutro”.

É importante ressaltar que a ordem em que o Grupo 3 foi submetido seria similar ao cenário real, onde os usuários que anteriormente não utilizavam uma ferramenta (Tarefa 1), começariam usando a ferramenta para auxiliar nas tarefas de manutenção de um software livre. Analisando desse ponto de vista, as evidências do experimento realizado mostram que as técnicas agregadas na ferramenta podem ter uma aceitação

satisfatória com relação ao contexto do experimento aplicado, isto é, na realização de tarefas de manutenção de software livre por novatos ou pessoas com pouca experiência com software livre.

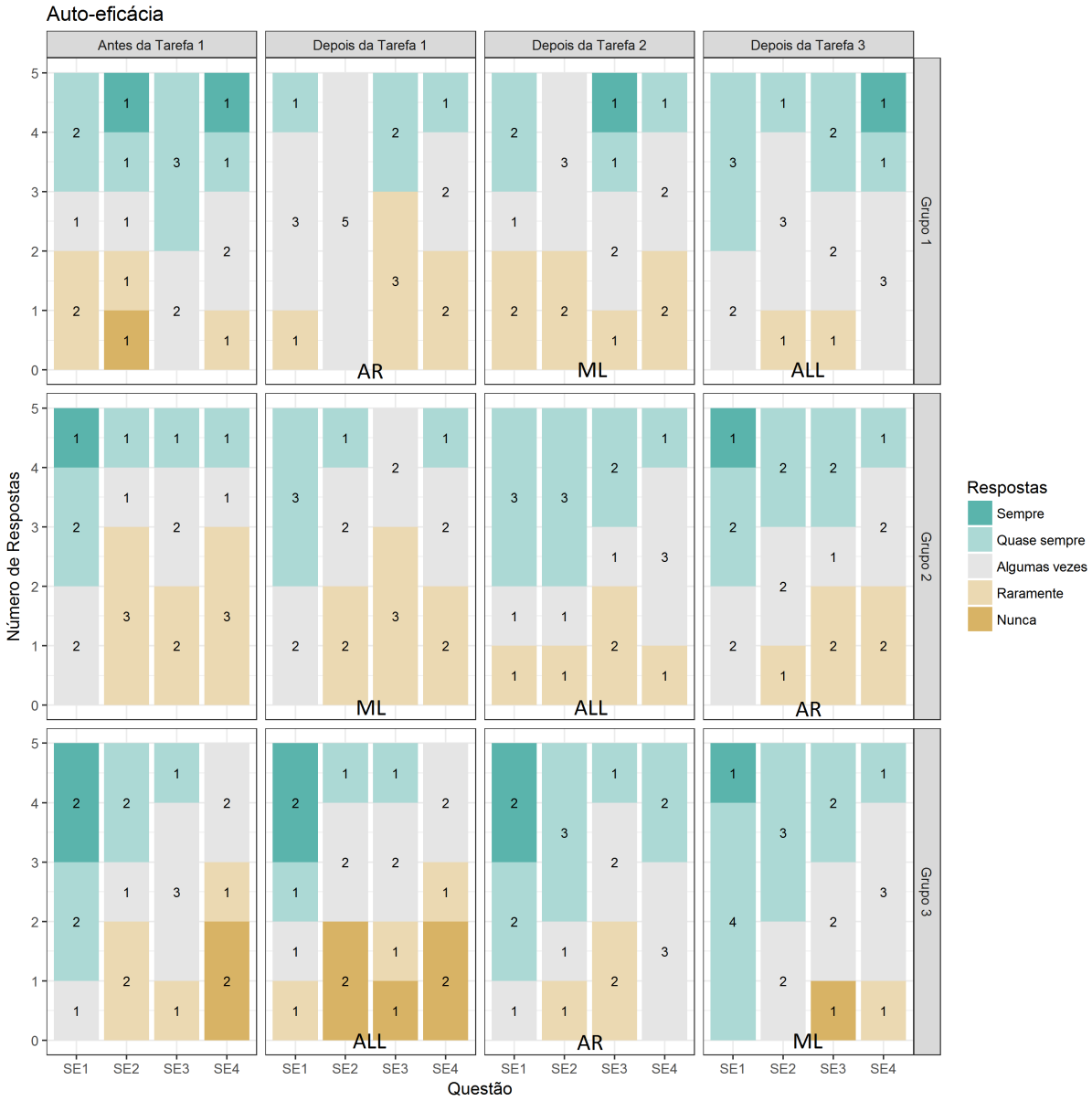
O teste de Wilcoxon também foi aplicado para cada grupo para aferir se houve diferença significativa no resultado do questionário TAM entre as técnicas. No entanto, não foram encontradas diferenças estatisticamente significativas, ou seja, com valor-p menor que $0,05$ ($p - value < 0,05$).

4.5.4 Resultado da Autoeficácia

A Figura 15 apresenta o gráfico que sumariza o resultado da autoavaliação da eficácia, ou autoeficácia (SE). A autoeficácia foi avaliada aplicando um questionário com questões respondidas antes de iniciar as tarefas e logo depois de terminar cada uma das três tarefas. O gráfico apresenta as respostas na escala de Likert com 5 níveis, agrupadas por Grupo e pelos momentos da aplicação. Tais momentos foram antes de iniciar, logo após finalizar a Tarefa 1, logo após finalizar a Tarefa 2 e, por fim, logo após finalizar a Tarefa 3.

A Figura 16 apresenta a soma da pontuação das questões de autoeficácia dos participantes nos quatro momentos em que o questionário foi aplicado. A pontuação da resposta de uma questão varia de 1 a 5, que equivalem a resposta “Raramente” e “Sempre”, respectivamente.

No geral, a média foi de $12,45$ pontos e a mediana foi de 13 pontos. Pode-se notar uma queda na autoeficácia em alguns casos entre o momento “Antes da Tarefa 1” e “Depois da Tarefa 1”. Por exemplo, é possível notar uma queda significante dos participantes P04 do Grupo 1 e P07 do Grupo 2, cujas pontuações de autoeficácia diminuiu 8 e 7 pontos, respectivamente. Foi observado também que os participantes do Grupo 3 mantiveram (P13 e P14) ou perderam (P11, P12 e P15) a autoeficácia. Os casos de queda da autoeficácia são dos participantes P01 e P02 do Grupo 1. A diminuição da autoeficácia pode ter ocorrido porque o participante enfrentou dificuldades para realizar a Tarefa e “não sabe que ele não sabe”, conforme notado por outros estudos relacionados à contribuição em projetos de software livre, como Davidson et al. (2014) e Steinmacher et al. (2016). No caso do Grupo 3, o que pode ter influenciado nesse resultado é a ordem das técnicas, pois o Grupo 3 iniciou o experimento com uma tarefa sem uso de técnica para predição de artefatos (ALL). O que reforça esse resultado são os casos de aumento da autoeficácia com o uso de técnicas, por exemplo, para o participante P03



ALL: Sem o uso de técnica. AR: Regras de Associação. ML: Classificação.

Figura 15: Resultado da Autoeficácia sumarizado por técnica.
Fonte: Autoria própria.

do Grupo 1 e P06 do Grupo 2.

Levando em consideração apenas a Tarefa 3, independente do grupo e da técnica que foi utilizada na tarefa, é importante ressaltar que essa tarefa não possuía descrição, o que pode ter sido uma das causas onde a autoeficácia dos participantes diminuiu, que foram os participantes P01, P06, P11, P12 e P13.

Apesar do Grupo 3 ter tido uma queda de autoeficácia na primeira tarefa, na qual não utilizava técnica para predição dos artefatos, em outros grupos houve 2 ca-

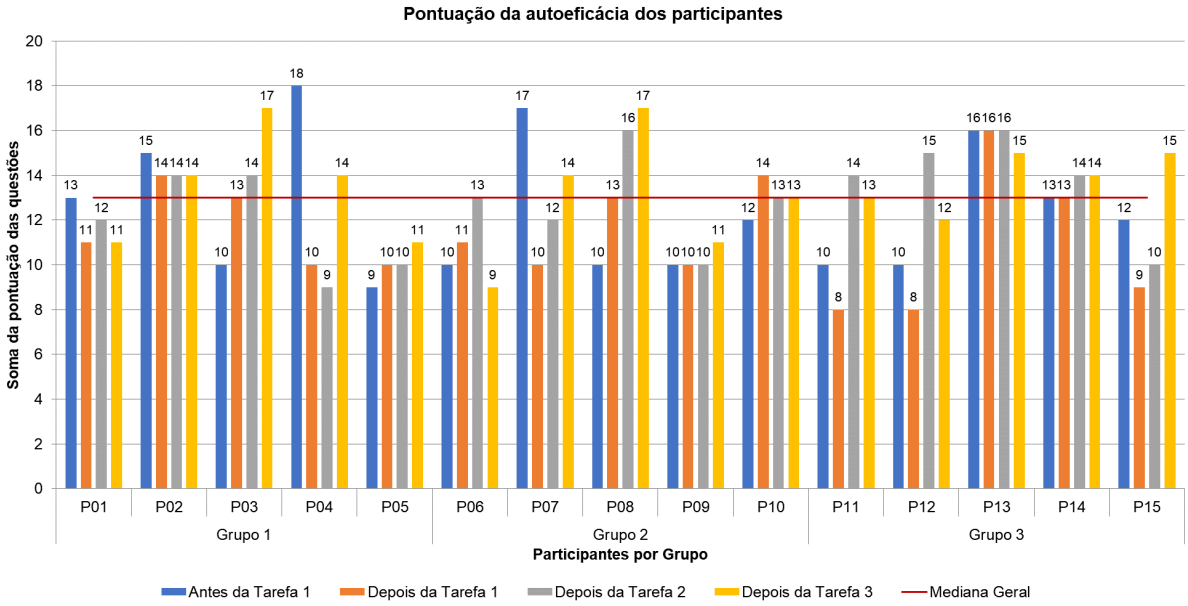


Figura 16: Resultado da Autoeficácia sumarizado por participante.
Fonte: Autoria própria.

sos de diminuição da autoeficácia quando nenhuma técnica de predição foi usada. O primeiro caso é do P01 do Grupo 1, que, embora seja uma diferença pequena, houve a autoeficácia diminuída de 12 para 11 pontos, entre a Tarefa 2 e a Tarefa 3. O outro caso foi com o P10 do Grupo 2, cuja autoeficácia diminuiu de 10 para 14 para 13 entre as Tarefas 1 e 2. Nos outros casos dos Grupos 1 e 2, a autoeficácia se manteve ou aumentou se comparado com a tarefa que não utilizava a técnica de Regras de Associação ou Classificação e a tarefa anterior, que usava alguma dessas técnicas.

Outro ponto a se notar no Grupo 3 é que entre o momento "Depois da Tarefa 1" e "Depois da Tarefa 2", houve um aumento da autoeficácia, com exceção do participante P13, que manteve a mesma autoeficácia. Para os outros grupos, houve apenas um participante em cada grupo que reduziu a autoeficácia, que são os participantes P04 do Grupo 1 e P10 do Grupo 2. Isso significa que há evidências do aumento da autoeficácia quando utilizada a técnica para predizer os artefatos propensos a mudarem.

Pode-se observar também que alguns participantes tiveram sua autoeficácia aumentada progressivamente, como no caso dos participantes P03 e P08. Esses dois participantes tiveram um aumento significativo, pois iniciaram com uma autoeficácia de 10 pontos de um total de 20, e ao final da última tarefa a autoeficácia era de 17 pontos.

Para aferir se houve diferença significativa na autoeficácia entre as tarefas por Grupo, foi feito o teste de Wilcoxon de forma pareada. No entanto, não foram encon-

tradas diferenças estatisticamente significativas, ou seja, com valor-p menor que $0,05$ ($p - value < 0,05$).

4.5.5 Resultado do Desempenho da Execução das Tarefas pelos Participantes

A Tabela 9 apresenta o desempenho das técnicas de Regras de Associação (AR), Classificação (ML) e sem técnica (ALL). Quando não há uso de técnica, foi considerado que não houve acertos nem erros, resultando em uma precisão e sensibilidade com valor 0 (zero). Em relação ao uso das técnicas, pode-se observar que a técnica de Regras de Associação obteve mais acertos do que a técnica de Classificação. Entretanto, a técnica de Regras de Associação possui mais predições falsos-positivas do que a técnica de Classificação, corroborando com os estudos preliminares do uso da técnica de Classificação (WIESE et al., 2015b, 2017).

O desempenho das técnicas se mostrou importante na ferramenta, já que alguns participantes foram influenciados pela predição e escolheram aqueles artefatos mais propensos a mudarem, de acordo com cada técnica. Alguns participantes confiaram nas predições apresentadas na ferramenta, como o participante P14 na Tarefa 2, que descreveu: *"Foi escolhido os arquivos com maiores probabilidades de mudança [sic]"*. O participante P09 também confiou nas predições apresentadas na Tarefa 3, conforme o seu *feedback*: *"Escolhi esse arquivo pois ele está entre os 5 mais prováveis e tem mais semelhança com a descrição [sic]"*. Outro caso foi com o participante P15 na Tarefa 3, que disse confiar totalmente na ferramenta, conforme o *feedback* fornecido: *"Me sinto confortável em confiar totalmente na ferramenta [sic]"*. Isso reforça que o desempenho da técnica é um dos fatores importante para os usuários, que pode ser relacionado à Utilidade Percebida do TAM.

Na Tabela 10, é apresentado o resultado do desempenho dos participantes em cada Tarefa usando determinada técnica. Além disso, os participantes foram agrupados por Grupo. O desempenho foi medido pelo número de acertos e erros baseado nas indicações de artefatos que o participante mudaria e nos artefatos que de fato mudaram para completar a tarefa. O tempo de realização da tarefa também foi analisado.

A primeira observação que se pode fazer é que os participantes não acertaram pelo menos um artefato para completar a tarefa quando não utilizavam alguma técnica para predizer os artefatos (ALL). Isso mostra uma evidência de que as técnicas podem ajudar novatos na manutenção do software.

Tabela 9: Desempenho das técnicas por tarefa.

Tarefa	Técnica	# Artefatos listados	# Artefatos preditos	# VP	# VN	# FP	# FN	Prec.	Sens.
Tarefa 1	ALL	5920	0	0	0	0	0	0,00	0,00
	AR	16	15	2	0	14	0	0,13	1,00
	ML	16	9	0	5	9	2	0,00	0,00
Tarefa 2	ALL	5920	0	0	0	0	0	0,00	0,00
	AR	58	17	1	40	16	1	0,06	0,50
	ML	60	1	0	58	0	2	0,00	0,00
Tarefa 3	ALL	5920	0	0	0	0	0	0,00	0,00
	AR	49	11	2	38	9	0	0,18	1,00
	ML	51	7	1	43	6	1	0,14	0,50

ALL: Sem o uso de técnica. AR: Regras de Associação. ML: Classificação. VP.: Verdadeiro Positivo. VN: Verdadeiro Negativo. FP: Falso Positivo. FN: Falso Negativo. Prec.: Precisão. Sens.: Sensibilidade.

Fonte: Autoria própria.

Os participantes que tiveram melhor desempenho foram P06, P07, P02 e P09, sendo que os participantes P06, P07 e P09, do Grupo 2, tiveram destaque no desempenho na Tarefa 3 utilizando a técnica de Regras de Associação. Os participantes P06 e P07 indicaram todos os 2 artefatos corretamente, sem que outro artefato fosse indicado incorretamente. Já o participante P02, do Grupo 1, indicou 2 artefatos, sendo que apenas um artefato foi correto. Por fim, participante P09, do Grupo 2, indicou e acertou um único artefato.

De maneira geral, os participantes P06, P07, P02 e P09 possuem alguns pontos em comum no perfil. Todos já cursaram alguma disciplina a respeito de software livre, possuem experiência com a linguagem Java e C/C++, usaram a IDE Eclipse, e não possuem experiência com o Apache CXF.

Analisando o perfil do participante P09, observa-se que ele nunca contribuiu ou tentou contribuir para um software livre, em contraste com outros 3 participantes que tiveram os melhores desempenhos. Além disso, ele é o único de todos os participantes que está a mais de 4 semestres de terminar o curso de BCC, em contraste com os outros 3 participantes, que estão a 2 semestres de concluir o curso. Analisando o comentário dado pelo participante P09 na ferramenta na Tarefa 2, que descreveu *"Escolhi esse arquivo, pois ele está entre os 5 mais prováveis e tem mais semelhanças com a descrição [sic]."*, mostra que, além da descrição da tarefa, a ferramenta também o guiou na escolha dos artefatos que mudaria ou analisaria para completar a tarefa.

Quanto ao perfil do participante P07, nota-se que ele possui experiência na in-

Tabela 10: Desempenho dos participantes nas tarefas.

Grupo 1							
Técnica	**Tarefa**		**P01**	**P02**	**P03**	**P04**	**P05**
AR	T1	Acertos	1	1	1	0	0
		Erros	5	1	5	3	5
ML	T2	Acertos	1	1	0	0	1
		Erros	4	2	1	2	2
ALL	T3	Acertos	0	0	0	0	0
		Erros	1	3	3	6	3
Grupo 2							
Técnica	**Tarefa**		**P06**	**P07**	**P08**	**P09**	**P10**
AR	T3	Acertos	2	2	1	1	0
		Erros	0	0	4	0	1
ML	T1	Acertos	0	0	0	0	0
		Erros	2	3	4	1	3
ALL	T2	Acertos	0	0	0	0	0
		Erros	3	3	7	2	21
Grupo 3							
Técnica	**Tarefa**		**P11**	**P12**	**P13**	**P14**	**P15**
AR	T2	Acertos	0	0	0	0	0
		Erros	3	2	11	3	6
ML	T3	Acertos	1	0	2	1	1
		Erros	2	2	3	3	3
ALL	T1	Acertos	0	0	0	0	0
		Erros	1	2	121	20	28

ALL: Sem o uso de técnica. AR: Regras de Associação. ML: Classificação.

Fonte: Autoria própria.

dústria de software, diferente dos outros 3 participantes, e se autoavalia como desenvolvedor com nível 4 numa escala de 1 a 5, onde 1 significa um desenvolvedor iniciante e 5 um desenvolvedor avançado. Apesar do participante P07 obter um resultado perfeito, o participante P06 obteve o mesmo resultado, mas não declarou possuir experiência na indústria de software e se autoavaliou como um desenvolvedor mediano (nível 3). Além disso, o participante P07 já teve entre 2 e 10 contribuições aceitas em softwares livres, diferente do P06 que já tentou contribuir, mas não teve sua(s) contribuição(ções) aceita(s).

Em relação ao perfil do participante P02, ele se autoavaliou como desenvolvedor avançado (nível 5) e não possui experiência na indústria de software. O comentário

descrito pelo participante P02 foi: *"A escolha dos arquivos foram baseadas na função do arquivo já modificado, os arquivos com maior probabilidade não foram marcados pois acredito que não seja problema com arquivos de teste. É difícil encontrar realmente quais arquivos modificar por alguns fatores, como não ter um conhecimento do projeto anteriormente e nem quais funções ele deveria exercer [sic]"*. Nesse comentário, o participante P02 descreve sucintamente que chegou ao resultado com base na classe previamente modificada. Além disso, o participante relata a dificuldade em encontrar os artefatos devido à falta de conhecimento prévio do projeto e qual o correto funcionamento da função. Esses dois fatores caracterizam algumas das barreiras enfrentadas pelos colaboradores novatos de projetos de software livre (STEINMACHER et al., 2016), o que pode ter afetado o desempenho dos participantes.

Em relação às técnicas, observa-se que as tarefas onde houve os melhores desempenhos pelos participantes foi com a técnica de Regras de Associação. No entanto, isso não significa que o uso da técnica de Regras de Associação seja melhor que a técnica de Classificação, uma vez que certas técnicas podem se adequar melhor a certas tarefas. Os indícios disso podem ser observados no desempenho dos participantes, mostrado na Tabela 10. Na Tarefa 3 os participantes conseguiram algum acerto em ambas as técnicas, o que não acontece nas outras Tarefas, 1 e 2. Na Tarefa 1 não houve acertos com o uso da técnica de Classificação (Grupo 2), mas houve acertos com o uso da técnica de Regra de Associação (Grupo 1). Por outro lado, na Tarefa 2 não houve acertos com o uso da técnica de Regras de Associação (Grupo 3), mas houve acertos com o uso da técnica de Classificação (Grupo 1).

Alguns participantes que marcaram mais erros foram o P13, P15, P10 e P14, respectivamente. A primeira observação é que os participantes P13, P15 e P14 foram do Grupo 3, e cometeram mais erros na Tarefa 1. Nessa tarefa, o Grupo 3 não utilizou técnica para predição de artefatos propensos a mudarem. Já o P10, do Grupo 2, errou mais na Tarefa 2, onde também não utilizava técnica para predição de artefatos propensos a mudarem.

Pode-se observar que o participante P13 do Grupo 3 indicou 121 artefatos que ele mudaria para completar a primeira tarefa. Analisando quanto ao perfil, nota-se que o P13 se autoavalia como um desenvolvedor intermediário (nível 3), e que já tentou contribuir, mas não houve aceite. Além disso, faltam 4 semestres para conclusão do curso de BCC, acima da mediana dos participantes, que é de 2 semestres. O participante P13 comentou o seguinte: *"Os arquivos tem algum tipo de relacionamento com o arquivo*

modificado ou com a issue (tarefa) relatada [sic]". Conforme o seu comentário, ele não relatou de forma objetiva o método que ele utilizou para indicar os artefatos que mudaria para completar a Tarefa 1. Devido ao resultado, além do *feedback* e do perfil do participante P13 na Tarefa 1, esse caso pode ser considerado um *outlier*.

A Tabela 11 apresenta o número de verdadeiro-positivo (VP), falso-positivo (FP) e falso-negativo (FN), além das medidas de precisão e sensibilidade no que diz respeito aos artefatos indicados pelos participantes. Pode-se observar que o participante P06 e P07, ambos do Grupo 2, tiveram uma precisão e sensibilidade igual a 1 na Tarefa 3, que significa que acertaram exatamente os artefatos que deviam ser modificados nessa tarefa. O participante P09, por sua vez, também na Tarefa 3, indicou e acertou 1 dos 2 artefatos corretos, o que resultou em uma precisão igual a 1 e uma sensibilidade igual a $0,50$. Em contraste, o participante P13 do Grupo 3, indicou corretamente os 2 artefatos da Tarefa 2, resultando numa sensibilidade igual a 1. No entanto, além desses dois artefatos, o participante P13 indicou um artefato a mais, que estava errado. Isso fez com que o valor da precisão ficasse igual a $0,40$.

Outro dado que foi capturado do experimento foi o tempo. O tempo foi contado a partir do momento da disponibilização da tarefa para o participante até o momento da submissão das respostas da tarefa na ferramenta. Esses dados são apresentados na Figura 17, onde é exibido o gráfico de barras com o tempo que cada participante de determinado grupo levou em cada tarefa aplicada no experimento.

Na Figura 17, pode-se observar que alguns participantes levaram mais tempo que a mediana. Alguns deles também podem ser observados como *outliers* na Figura 18, que apresenta a caixa. Os dois primeiros casos foram com os participantes P01 e P05 do Grupo 1 na Tarefa 2, onde foi usado a técnica de Classificação para predizer os artefatos na ferramenta. Eles levaram 79,08 e 78,73 minutos, respectivamente, acima da mediana, que foi de 49,32 minutos. De acordo com o perfil do P01, o que pode justificar esse tempo é a falta de experiência com o IDE Eclipse, a autoavaliação como desenvolvedor iniciante, e não ter realizado uma disciplina que discutisse sobre software livre. Apesar desse perfil, pode-se verificar que o participante P01 na Tarefa 1 levou menos da metade do tempo que na Tarefa 2 e, em ambas as tarefas, houve acerto de 1 dos 2 artefatos corretos. Além disso, o participante P05 levou mais tempo na Tarefa 3, que não utilizou técnica para predição dos artefatos.

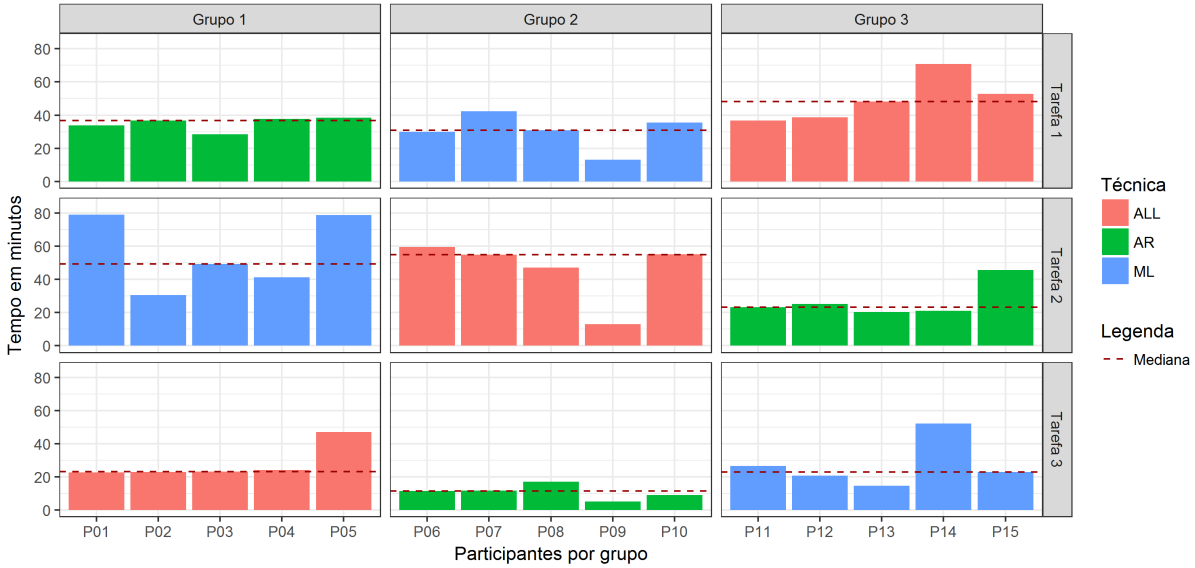
Diferente do participante P01, o participante P05 já tentou contribuir com correções de defeitos e documentação, mas não teve contribuições aceitas. Além disso, o

Tabela 11: Precisão e sensibilidade por participante nas tarefas.

Participante	Grupo	Tarefa	Técnica	# VP	# VN	# FP	# FN	Prec.	Sens.
P01	1	1	AR	1	10	5	1	0,17	0,50
P01	1	2	ML	1	55	4	1	0,20	0,50
P01	1	3	ALL	0	5919	1	2	0,00	0,00
P02	1	1	AR	1	14	1	1	0,50	0,50
P02	1	2	ML	1	57	2	1	0,33	0,50
P02	1	3	ALL	0	5917	3	2	0,00	0,00
P03	1	1	AR	1	10	5	1	0,17	0,50
P03	1	2	ML	0	59	1	2	0,00	0,00
P03	1	3	ALL	0	5917	3	2	0,00	0,00
P04	1	1	AR	0	13	3	2	0,00	0,00
P04	1	2	ML	0	58	2	2	0,00	0,00
P04	1	3	ALL	0	5914	6	2	0,00	0,00
P05	1	1	AR	0	11	5	2	0,00	0,00
P05	1	2	ML	1	57	2	1	0,33	0,50
P05	1	3	ALL	0	5917	3	2	0,00	0,00
P06	2	1	ML	0	14	2	2	0,00	0,00
P06	2	2	ALL	0	5917	3	2	0,00	0,00
P06	2	3	AR	2	47	0	0	1,00	1,00
P07	2	1	ML	0	13	3	2	0,00	0,00
P07	2	2	ALL	0	5917	3	2	0,00	0,00
P07	2	3	AR	2	47	0	0	1,00	1,00
P08	2	1	ML	0	12	4	2	0,00	0,00
P08	2	2	ALL	0	5913	7	2	0,00	0,00
P08	2	3	AR	1	43	4	1	0,20	0,50
P09	2	1	ML	0	15	1	2	0,00	0,00
P09	2	2	ALL	0	5918	2	2	0,00	0,00
P09	2	3	AR	1	48	0	1	1,00	0,50
P10	2	1	ML	0	13	3	2	0,00	0,00
P10	2	2	ALL	0	5899	21	2	0,00	0,00
P10	2	3	AR	0	48	1	2	0,00	0,00
P11	3	1	ALL	0	5919	1	2	0,00	0,00
P11	3	2	AR	0	55	3	2	0,00	0,00
P11	3	3	ML	1	47	2	1	0,33	0,50
P12	3	1	ALL	0	5918	2	2	0,00	0,00
P12	3	2	AR	0	56	2	2	0,00	0,00
P12	3	3	ML	0	49	2	2	0,00	0,00
P13	3	1	ALL	0	5799	121	2	0,00	0,00
P13	3	2	AR	0	47	11	2	0,00	0,00
P13	3	3	ML	2	46	3	0	0,40	1,00
P14	3	1	ALL	0	5900	20	2	0,00	0,00
P14	3	2	AR	0	55	3	2	0,00	0,00
P14	3	3	ML	1	47	3	1	0,25	0,50
P15	3	1	ALL	0	5892	28	2	0,00	0,00
P15	3	2	AR	0	52	6	2	0,00	0,00
P15	3	3	ML	1	57	3	1	0,25	0,50

ALL: Sem o uso de técnica. AR: Regras de Associação. ML: Classificação. VP: Verdadeiro Positivo. VN: Verdadeiro Negativo. FP: Falso Positivo. FN: Falso Negativo. Prec.: Precisão. Sens.: Sensibilidade.

Fonte: Autoria própria.



ALL: Sem o uso de técnica. AR: Regras de Associação. ML: Classificação.

Figura 17: Tempo por Participante, Grupo e Tarefa.

Fonte: Autoria própria.

participante P05 está a um semestre de concluir o curso, mais perto do que o P01. O participante P05 relatou que *"Apenas os arquivos relacionados ao HttpRequest e URI, pois são utilizados no arquivo WadlGenerator.java [sic]"*, o que demonstra que o participante possui experiência com a linguagem. Apesar disso, o participante P05 conseguiu acertar um dos dois artefatos corretos.

O participante P01 relatou o seguinte na Tarefa 2: *"Desta vez não foi tão difícil localizar o erro, uma vez que a implementação do código já possibilitava o entendimento do erro, ou seja, o motivo de seu aparecimento. A ferramenta Recominer mostrou sim os arquivos que se encontravam relacionados com o erro, entretanto foram apresentados muitos mais artefatos que não teriam a necessidade de serem apresentados [sic]"*. Nesse relato, o participante coloca como ponto negativo a quantidade de artefatos que foram apresentados. Isso ocorreu pois na Tarefa 2 haviam 58 artefatos apresentados na ferramenta, um número menor se comparado com a Tarefa 1, que apresentou 15 artefatos. Isso é esperado, uma vez que a técnica de Regras de Associação faz o pareamento dos artefatos no contexto do *commit*, um escopo menor que o contexto da tarefa utilizado pela técnica de Classificação. Devido ao relato do participante P01, o que pode ter ocorrido é que os participantes P01 e P05 fizeram uma análise minuciosa de mais artefatos, já que foram apresentados muito mais do que na primeira tarefa.

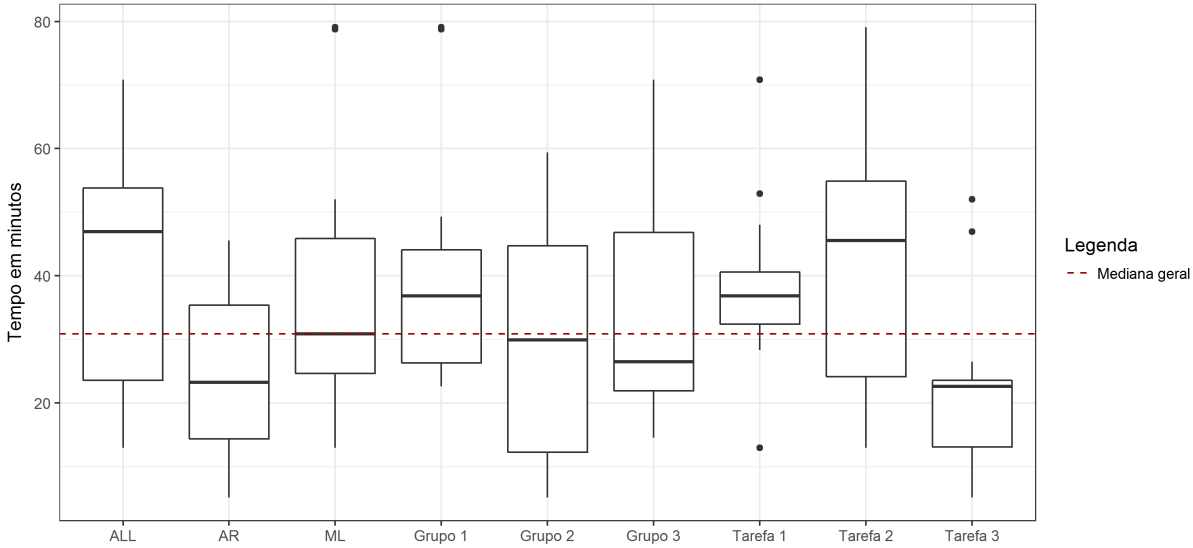
Além desses dois casos, o participante P14 na Tarefa 1, que não utilizou técnica para predizer os artefatos pela ferramenta, levou 70,81 minutos para completar a tarefa, acima da mediana de 48,05 minutos. Por ser a primeira tarefa sem o uso de técnicas para predição de artefatos, é possível que o participante tenha tido dificuldades em entender e realizar a tarefa. Esse resultado pode ter sido agravado também pelo perfil do participante, pois é semelhante ao perfil do P01. Ele nunca contribuiu com software livre e também não cursou uma disciplina que discutisse a respeito de software livre. O que o diferencia do participante P01 é que o participante P14 possui experiência com o IDE Eclipse e se autoavaliou como desenvolvedor entre o iniciante e o intermediário (nível 2). O participante P14 não explicou como chegou no resultado, conforme o seguinte comentário: *"Foi escolhido os artefatos que mais pensei ter relação com as classes referentes ao problema em questão [sic]"*. Pode-se observar também que o participante P14 teve um tempo maior na Tarefa 3, levando 52,03 minutos para completá-la, o que pode ser justificado pelo perfil, acima da mediana de 22,80 minutos.

Outra observação é em relação ao participante P09. É possível observar que o tempo levado para realizar a tarefa foi de quase 12,98 minutos para a Tarefa 1, 12,95 minutos para a Tarefa 2, e 5,15 minutos para a Tarefa 3, abaixo das medianas 30,87, 54,77 e 11,43, respectivamente. Uma das justificativas para esse caso é que o participante P09 nunca contribuiu com algum projeto de software livre e, também, é o único participante que está a mais de 4 semestres de concluir o curso. Mas, como visto anteriormente, apesar disso, o participante P09 conseguiu ter um bom desempenho na Tarefa 3, pois indicou e acertou um artefato que mudaria para completar a tarefa. A sensibilidade ficou com o valor 0,5 e a precisão em 1.

4.6 AMEAÇAS À VALIDADE

Esse estudo está sujeito a limitações e falhas, assim como outros estudos empíricos (EASTERBROOK et al., 2008).

Uma das ameaças à validade externa nesse estudo, é com relação à generalização, pois este trabalho se limitou a 3 tarefas de manutenção do software livre Apache CXF. Essas tarefas foram selecionadas aleatoriamente, mas, para uma maior generalização dos resultados, são necessários mais experimentos com outras tarefas, bem como outros softwares livres e, também, industriais. Para isso, pretende-se futuramente estender esse trabalho.



ALL: Sem o uso de técnica. AR: Regras de Associação. ML: Classificação.

Figura 18: Gráfico de Caixas do Tempo por Técnica, Grupo e Tarefa.
Fonte: Autoria própria.

Outra ameaça à validade externa é em relação aos participantes do experimento desse estudo. Esse trabalho limitou-se a 15 alunos da área de Ciência da Computação, embora houvessem 2 alunos com experiência na indústria e 3 alunos com experiência de sucesso em contribuições para softwares livres. Porém, todos os alunos participantes do experimento representaram colaboradores novatos, já que nenhum dos participantes tinham experiência com o software Apache CXF, seja contribuindo ou o utilizando.

Por serem submetidos a 3 tarefas, os participantes poderiam adquirir experiência tanto com a utilização da ferramenta, incluindo as técnicas de predição, quanto com o projeto durante o experimento. Para minimizar esse viés, os 15 alunos foram divididos em 3 grupos de 5 alunos cada. Dessa forma, foi possível com que cada grupo fosse submetido a diferentes técnicas em ordem diferente, mas com a mesma ordem para as tarefas. Ainda assim, não foi possível que algumas combinações fossem testadas no experimento. Nesse experimento, as predições da técnica de Regras de Associação sempre eram exibidas em determinada tarefa antes de outra tarefa que possuía os artefatos preditos pela técnica de Classificação. Os resultados poderiam ter sido diferentes caso tivesse sido o contrário, ou seja, apresentar os artefatos preditos pela técnica de Classificação antes dos artefatos preditos pela técnica de Regra de Associação.

Esse estudo também se limitou a avaliar a aceitação da ferramenta com o Mo-

delo de Aceitação de Tecnologia (TAM) em sua primeira versão, embora exista outras formas de avaliar a aceitação da ferramenta como a Teoria Unificada de Aceitação e Uso de Tecnologia (do inglês *Unified Theory of Acceptance and Use of Technology*) (VENKATESH et al., 2003). O TAM se tornou modelo-chave na compreensão dos preditores do comportamento humano para a aceitação ou rejeição potencial da tecnologia, pois diversos estudos enfatizaram sua ampla aplicabilidade em várias tecnologias (MARANGUNIĆ NIKOLA E GRANIĆ, 2015). Além disso, o TAM foi validado com a análise fatorial confirmatória, que mostrou que as questões e suas respostas estavam relacionados significativamente com os constructos de Utilidade e Facilidade de Uso, confirmando a validade do modelo.

Existe o risco também dos alunos não terem se esforçado em entender e realizar as tarefas, encontrando os artefatos para completá-la corretamente, uma vez que os alunos participaram por livre e espontânea vontade e não foram remunerados ou beneficiados por participarem do experimento. Isso foi garantido por meio de um termo de consentimento assinado pelos participantes. Para minimizar esse risco, a tela do computador durante a realização da tarefa foi gravada e, também, solicitou-se que o participante explicasse textualmente na ferramenta de como chegou nos artefatos escolhidos. Ainda com relação aos alunos, como alguns podem ser colegas, é difícil evitar que conversassem sobre o experimento. Para minimizar os riscos no experimento, os participantes foram informados sobre o impacto disso sobre o experimento e foi solicitado que não trocassem informações entre si sobre as suas tarefas.

Outra limitação é em relação as técnicas de predição que foram implementadas e aplicadas no experimento por meio da ferramenta. Nesse trabalho, foram abrangidas duas técnicas de predição de mudanças conjuntas de artefatos já estudadas na literatura, que são a Regras de Associação (ZIMMERMANN et al., 2005) e a Classificação com Floresta Randômica (WIESE et al., 2017). A apresentação da predição do artefato mudar ou não na ferramenta também pode ter influenciado nos resultados. Para a técnica de Classificação, foram apresentadas como prováveis de mudar as probabilidades resultantes do algoritmo maiores que 50%. Para a técnica de Regras de Associação, os artefatos que tivessem os 10 maiores valores de suporte e confiança foram apresentados como prováveis de mudar, e o restante dos artefatos foi apresentado como improváveis de mudar.

4.7 DISCUSSÃO DOS RESULTADOS

Nesse estudo empírico, ficou evidente que as técnicas ajudam os colaboradores novatos a realizar tarefas de manutenção em software livre. Com o uso das técnicas, os participantes puderam indicar artefatos que deveriam mudar para completar a tarefa corretamente, ao contrário de quando não foi utilizada nenhuma técnica, onde nenhum participante conseguiu indicar pelo menos um artefato corretamente. Além disso, os participantes realizaram as tarefas em menor tempo quando apresentavam predições resultantes das técnicas, se comparado com quando não era utilizada nenhuma técnica. Isso quer dizer que o esforço foi menor quando as técnicas foram utilizadas para mostrar as possíveis predições. No entanto, o desempenho do participante ficou dependente do desempenho da técnica, já que alguns participantes confiaram nas predições apresentadas na ferramenta.

Algumas informações sobre o perfil dos participantes foram coletadas como complemento para o resultado. De certa forma, o perfil do participante ajudou na análise dos resultados desse trabalho. Embora as informações coletadas tenham sido suficientes para esse estudo, seria interessante obter informações mais específicas para uma melhor análise, como o tempo de experiência na indústria, tempo de experiência na academia, tempo de experiência com a linguagem de programação, entre outros.

Os colaboradores novatos foram representados por alunos que eram inexperientes com o software Apache CXF, embora a maioria dos participantes já tenham tido alguma experiência com colaboração em software livre. É importante notar que todos os alunos já tinham experiência com a linguagem de programação Java, dominante no projeto Apache CXF. Essas experiências podem ter ajudado os participantes nas tarefas do experimento e, portanto, podem ter influenciado nos resultados.

Alguns participantes relataram dificuldades devido à falta de conhecimento prévio do projeto. De fato, essas dificuldades caracterizam algumas das barreiras enfrentadas pelos colaboradores novatos de projetos de software livre, seja em relação à (falta de) documentação ou (falta de) conhecimento técnico (STEINMACHER et al., 2016). Esses fatores podem ter afetado o desempenho dos participantes na realização das tarefas e, consequentemente, na autoeficácia. No entanto, foge do escopo desse trabalho uma análise mais aprofundada sobre esse assunto, sendo necessários estudos futuros.

No geral, a autoeficácia não houve alterações significativas, apesar da amostra ser pequena. Isso foi confirmado com o teste de *Wilcoxon*. Foi possível notar que,

inicialmente, alguns participantes tinham uma autoeficácia alta. Após a primeira a tarefa, a autoeficácia teve uma queda significativa. Isso era esperado e corrobora com o fenômeno que "você não sabe que não sabe", o que significa que as pessoas podem ser excessivamente otimistas em tarefas mal compreendidas (DAVIDSON et al., 2014).

No geral, evidências obtidas pelo questionário TAM mostraram que a aceitação da ferramenta foi positiva. O resultado do TAM mostrou que a maioria dos participantes concordaram que a ferramenta é fácil de usar (Facilidade de Uso), que é um dos fatores que influenciam na aceitação e uso no futuro da ferramenta. Em relação ao fator de Utilidade e de Uso no Futuro, houve mais concordância pelos participantes quando técnicas de predição foram usadas para predizer as mudanças conjuntas do que quando não foi utilizada nenhuma técnica.

Em relação ao *feedback*, a ferramenta coletou a intenção de mudança dos participantes, permitindo que o participante marcasse indicando que mudaria esses artefatos para completar a tarefa. Isso possibilitou a análise de desempenho dos participantes nas tarefas. Além disso, a ferramenta também coletou um *feedback* textual, que ajudou a analisar qualitativamente o desempenho dos participantes. Apesar disso, muitos desses *feedbacks* foram descritos sucintamente e não forneceram informações suficientes para uma melhor análise. Isso pode ter ocorrido devido à liberdade dada aos participantes dizer o que ele achava da tarefa e da ferramenta, apesar de ter sido dito para enfatizarem na justificativa do resultado. Nesse ponto, é necessário realizar uma melhoria na coleta do *feedback* do usuário na ferramenta, de forma que seja mais objetiva na captura de fatores que podem influenciar no resultado das predições para, então, melhorar o desempenho da técnica, entre outras melhorias que possam ser feitas.

4.8 CONSIDERAÇÕES FINAIS

Nesse capítulo, foi descrita a metodologia do experimento para avaliar a aceitação pelos participantes e o suporte fornecido pela ferramenta. A avaliação foi aferida por meio de determinantes básicos na aceitação de tecnologia: Utilidade, Facilidade de Uso e Uso no Futuro. Com esse experimento, também pode-se avaliar se as técnicas de predição podem ou não ajudar na manutenção de software, principalmente no que diz respeito aos colaboradores novatos em projetos de software livre. Para tanto, tarefas concluídas de manutenção do software livre Apache CXF foram selecionadas e aplicadas à ferramenta para predizer as mudanças conjuntas. Além disso, alunos que não tinham experiência com o software Apache CXF participaram do experimento, caracterizando

o perfil de colaboradores novatos.

O experimento consistiu, então, em aplicar o questionário de autoeficácia a fim de medir a confiança dos participantes em realizar as tarefas de manutenção com o uso da ferramenta. No entanto, no geral, a autoeficácia dos participantes não teve mudanças significativas devido ao uso da ferramenta. O que foi possível notar é que alguns participantes mostraram uma alta autoeficácia inicialmente e, ao se depararem com a primeira tarefa, houve uma queda da autoeficácia, pois pode ter ocorrido o fenômeno em o participante "não sabia que não sabia".

Para avaliar a aceitação da ferramenta, foi aplicado o questionário baseado no TAM e em estudos anteriores que o aplicaram. O questionário do TAM captura a percepção do usuário sobre a Facilidade de Uso e Utilidade com relação à determinada tecnologia, como, no contexto desse trabalho, uma ferramenta. Evidências do experimento com o TAM mostraram que a ferramenta, em conjunto com as técnicas de predição de mudanças conjuntas, é capaz de ajudar colaboradores novatos em tarefas de manutenção. Além disso, no geral, a ferramenta foi considerada fácil de usar, conforme as respostas dos participantes. Contudo, a Utilidade da ferramenta ficou sujeita ao uso ou não da técnica, pois houve mais concordância da Utilidade por parte dos participantes em momentos em que foi utilizada alguma técnica do que quando nenhuma foi utilizada. Isso reforça que o uso da ferramenta, somado às técnicas de predição de mudanças conjuntas, pode ajudar os colaboradores novatos na manutenção do software.

Além da documentação nesse trabalho de como foi realizado, os dados, o código-fonte e o executável da ferramenta Recominer, utilizados nesse experimento, foram disponibilizados no GitHub, no endereço `https://github.com/rodrigokuroda/recominer/releases/tag/0.7`. A disponibilidade dessas informações tem o objetivo de encorajar a reprodução e a extensão desse trabalho. Além disso, estão disponibilizados os dados coletados no experimento conduzido nesse estudo, bem como o endereço para as gravações de tela dos participantes no experimento.

5 CONSIDERAÇÕES FINAIS

5.1 CONCLUSÕES

Nesse estudo, foi desenvolvida a ferramenta denominada Recominer, que automatiza a predição de mudanças conjuntas utilizando as técnicas de Regras de Associação e Classificação. Apesar da existência de outras ferramentas que também predizem mudanças conjuntas como o ROSE (ZIMMERMANN et al., 2005) e ImpactMiner (DIT et al., 2014), a ferramenta Recominer utiliza outra técnica promissora que até então não havia sido implementada em uma ferramenta: a Classificação. Essa técnica se mostrou capaz de reduzir o número de falsos positivos (WIESE et al., 2015c, 2017). Além disso, essas ferramentas não foram avaliadas na prática, de forma a avaliar a aceitação e autoeficácia dos participantes. Essas ferramentas também não possibilitam a captura do *feedback* das predições de mudanças conjuntas pelo usuário, o que permite realizar uma avaliação do desempenho dos participantes e obter críticas construtivas sobre a ferramenta.

A avaliação consistiu em realizar uma prova de conceito com tarefas do projeto de software livre denominado CXF da *Apache Software Foundation*. Esse experimento foi elaborado com tarefas de manutenção (defeitos) de projetos de software livre e contou com a participação de alunos do curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná Campus Campo Mourão. Para avaliar a aceitação da ferramenta, foi utilizado o Modelo de Aceitação de Tecnologia (TAM) composto por questões sobre a Percepção de Utilidade, Percepção de Facilidade de Uso e a Auto-predição (ou intenção) de Uso no Futuro. Além disso, a autoeficácia foi avaliada, que indica a confiança da capacidade percebida de um indivíduo em realizar uma determinada tarefa.

O resultado do TAM mostrou que a aceitação ferramenta Recominer foi positiva quanto ao fator de Facilidade de Uso Percebida, mas que pode ter alguns pontos melhorados devido ao *feedback* dos participantes. Em relação ao fator de Utilidade Per-

cebida e a intenção de Uso no Futuro, a aceitação pelos participantes ficou dependente da técnica que a ferramenta empregou na tarefa que o participante realizava. Onde foi utilizada alguma técnica, houve maior concordância na Utilidade Percebida e na intenção de Uso no Futuro do que quando não foi utilizado nenhuma técnica. Entretanto, não houve mudanças significativas na autoeficácia dos participantes no geral, embora alguns participantes mostrarem um aumento na medida em que as tarefas eram realizadas, enquanto alguns outros mantiveram uma autoeficácia semelhante ao longo das tarefas.

Baseado em evidências do experimento realizado, a ferramenta se mostrou capaz de apoiar desenvolvedores novatos na manutenção de software, reforçado com o *feedback* dos participantes. Ao utilizar as técnicas para predizer os artefatos propensos a mudarem em determinada tarefa, os desenvolvedores foram capazes de indicar mais artefatos corretamente do que quando não foi utilizado nenhuma técnica. Um dos fatores que contribui para isso é a redução de esforço para realizar a análise de impacto, já que as técnicas fazem uma análise prévia baseado no histórico e, assim, filtrando os artefatos a serem inspecionados. Consequentemente, o experimento mostrou evidências de redução no tempo para completar uma tarefa de manutenção em um software livre.

As evidências do experimento reforçam que o desempenho das técnicas em diferentes contextos é importante para que sejam criadas ferramentas mais eficientes para predição de artefatos. Com melhorias das técnicas de predição de artefatos, as ferramentas podem ser mais aceitas e, assim, adotadas pelos desenvolvedores na manutenção do software.

Entretanto, esse estudo necessita de mais experimentos para uma maior generalização, aplicando a ferramenta em mais projetos e mais usuários potenciais. Nesse estudo, houve a participação de um número pequeno de 15 participantes, todos alunos do curso de Bacharelado em Ciência da Computação. Os participantes declararam que não tinham colaborado ou usado o projeto Apache CXF e, portanto, pode ser considerado como colaboradores novatos no projeto. Outra ameaça à validade está relacionada às tarefas. Nesse trabalho, o objetivo do experimento foi avaliar a ferramenta com tarefas já concluídas de manutenção de software. Foram selecionadas três tarefas consideradas mais fáceis com base em condições como número de linhas e quantidade de artefatos alterados, dado que os novatos iriam realizá-las.

Embora nesse trabalho a ferramenta não tenha sido avaliada em softwares da indústria, ela pode ser empregada também na manutenção de softwares. Profissionais

iniciantes na área podem se beneficiar com a ferramenta, auxiliando-os nas tarefas de manutenção e evolução dos softwares. Porém, outro estudo é necessário para avaliar a ferramenta nesse cenário.

5.2 CONTRIBUIÇÕES

Esse trabalho contribui com a ferramenta Recominer, que faz predições de mudanças conjuntas com informações contextuais obtidas de repositórios de software. Para tanto, foi implementada a técnica de Regras de Associação e a Classificação na ferramenta, sendo que essa última foi inédita. Essas técnicas podem apoiar desenvolvedores na análise de impacto de mudança de um software, ou seja, na manutenção do software. Com a ferramenta, desenvolvedores e colaboradores podem ser mais eficientes na manutenção do software. Evidências do experimento com usuários representando o perfil de colaboradores novatos mostraram que, com o uso a ferramenta Recominer, somado às técnicas de predição de mudanças conjuntas, os colaboradores se tornaram mais eficientes (mais acertos em menor tempo) do que quando não foi utilizada técnica alguma.

Outra contribuição resultante desse trabalho é em relação ao experimento aplicado. Nenhum dos estudos encontrados, que estão no contexto semelhante desse trabalho, aplicou uma avaliação da aceitação pelos participantes utilizando o Modelo de Aceitação de Tecnologia (TAM) e/ou Autoeficácia. Essa avaliação é importante para tentar predizer se uma tecnologia, que no escopo desse trabalho é a ferramenta Recominer, será aceita ou rejeitada pelos usuários. Além disso, o *feedback* pode mostrar alguns pontos da ferramenta que devem ser melhoradas para uma maior aceitação pelos usuários.

Durante a elaboração desse trabalho, em conjunto com outros autores, foram realizadas algumas contribuições para comunidade científica com estudos relacionados ao uso da técnica de classificação com informações contextuais para predição de mudanças conjuntas. Esses estudos preliminares, que motivaram e embasaram esse trabalho, foram publicados e são descritos brevemente a seguir:

- *WIESE, Igor S. ; KURODA, Rodrigo T.; NASSIF JUNIOR, Douglas; RE, Reginaldo ; OLIVA, Gustavo A.; GEROSA, Marco A. . Using Structural Holes Metrics from Communication Networks to Predict Change Dependencies. In: 20th International Conference on Collaboration and Technology (CRIWG), 2014, Santiago, Chile.* ***Procee-***

***dings of the 20th International Conference on Collaboration and Technology (CRIWG 2014)**, 2014. v. 8658. p. 294-310.* Esse estudo encontrou evidências de que as métricas de buracos estruturais, calculadas a partir de grafos que representam a comunicação dos colaboradores, podem predizer mudanças conjuntas de artefatos.

- *WIESE, Igor S.; KURODA, Rodrigo T.; RE, Reginaldo; OLIVA, Gustavo A. ; GEROSA, Marco A. Um estudo empírico do uso da comunicação para caracterizar a ocorrência de dependências de mudanças no projeto Rails. In: 2nd Workshop on Software Visualization, Evolution, and Maintenance - VEM, 2014, Maceio, AL.* ***2nd Workshop on Software Visualization, Evolution, and Maintenance (VEM)****. Maceió, 2014.* Esse estudo evidenciou que as métricas de comunicação, calculadas a partir do grafo construído com base no histórico da comunicação dos colaboradores, podem ajudar na classificação de mudanças conjuntas fortes e fracas. Uma mudança conjunta é forte quando há uma frequência historicamente alta da ocorrência dessa mudança. Dentre essas métricas, destacou-se as métricas globais do grafo e as métricas de buracos estruturais.

- *WIESE, Igor S.; KURODA, Rodrigo T.; RÉ, Reginaldo; BULHÕES, R. S.; OLIVA, Gustavo A.; GEROSA, Marco A. Do historical metrics and developers communication aid to predict change couplings?.* ***Revista IEEE América Latina****, v. 13, p. 1979-1988, 2015.* Esse estudo mostrou evidências que as métricas históricas, calculadas com informações extraídas dos repositórios de software, e de comunicação, calculadas a partir da construção de grafo com base no histórico da comunicação dos colaboradores, são úteis para a predição de mudanças conjuntas. Particularmente, um subconjunto das métricas se mostrou estatisticamente significante para a predição.

- *WIESE, Igor S.; KURODA, Rodrigo T.; RE, Reginaldo; OLIVA, Gustavo A.; GEROSA, Marco A. An Empirical Study of the Relation Between Strong Change Coupling and Defects Using History and Social Metrics in the Apache Aries Project. In:* ***11th Intl. Conf. on Open Source Systems****, 2015, Florença. Lecture and Notes in Computer Science (LNCS), 2015. v. 451. p. 3-12.* Esse estudo encontrou uma correlação positiva de mudanças conjuntas fortes, determinado pelo classificador com base na frequência em que ocorrem, com o número de defeitos.

- *WIESE, Igor S.; GEROSA, Marco A.; STEINMACHER, Igor F.; KURODA, Rodrigo T.; RE, Reginaldo ; OLIVA, Gustavo A. Informações contextuais do desenvolvimento de*

software na predição de propagação de mudanças. In: Congresso Brasileiro de Software: Teoria e Prática (CBSoft), 2015, Belo Horizonte. ***XXIX Simpósio Brasileiro de Engenharia de Software (SBES)****, 2015.* Esse estudo mostrou que as informações contextuais dos repositórios de software, extraídas das solicitações de mudança, comunicação entre os desenvolvedores e as modificações nos artefatos, podem ajudar na predição de mudanças conjuntas utilizando classificadores.

- *WIESE, Igor S.; RÉ, Reginaldo; KURODA, Rodrigo T.; STEINMACHER, Igor; TREUDE, Christoph; OLIVA, Gustavo A.; GEROSA, Marco A. Using contextual information to predict co-changes.* ***Journal of Systems And Software****, v.1, p. 1-15, 2017.* Esse estudo mostrou evidências de que os modelos de predição, construídos com classificadores e baseados em informações contextuais das mudanças de software, são precisos e reduzem o número de falsos-positivos quando comparado com outras técnicas. Assim, modelos construídos com a técnica de classificação usando informações contextuais podem ajudar na manutenção e na evolução do software.

5.3 TRABALHOS FUTUROS

Como trabalhos futuros, são necessários mais estudos com experimento empírico a fim de generalizar os resultados, tanto como para os projetos de software quanto para os potenciais usuários. Portanto, dada a importância desse estudo, pretende-se realizar futuramente outros experimentos com projetos de software livre de diferentes contextos e comunidades e software industriais.

Um experimento com desenvolvedores da comunidade de projetos de código abertos e profissionais da indústria também é interessante, uma vez que esses participantes representam outro perfil de potenciais usuários. Realizando esse experimento futuramente, pode-se obter opiniões e sugestões a respeito da ferramenta e, assim, realizar melhorias para que sua aceitação, como a facilidade de uso e utilidade, seja ainda melhor.

Em relação à ferramenta, existem muitas melhorias que podem ser feitas, pois o que foi desenvolvido nesse trabalho é uma versão inicial. Uma dessas melhorias seria implementar outras técnicas de predição de mudanças conjuntas e unificá-las, pois as abordagens híbridas podem ajudar na melhoria do desempenho das técnicas, conforme evidências de estudos anteriores (GETHERS et al., 2012; KAGDI et al., 2013). Além disso, é necessário que seja possível configurar as técnicas de predição de mudanças

conjuntas e parametrizar outras variáveis que possam influenciar na predição, permitindo assim que outros pesquisadores e usuários também possam utilizar a ferramenta e personalizá-la. Outras melhorias na ferramenta incluem o suporte a outros bancos de dados relacionais e não-relacionais para facilitar a integração com outras ferramentas, suporte para extração de dados de outros sistemas de gerenciamento de solicitação de mudanças e outros sistemas de controle de versão.

Outro ponto a melhorar na ferramenta é a coleta de *feedback*. Essa coleta não se mostrou efetiva nesse estudo, que foi feita utilizando um campo de texto livre para o usuário escrever. Uma das formas seria complementá-lo ou substituí-lo por questões mais objetivas para o usuário responder em relação às predições e a ferramenta. Uma ideia é formular questões com respostas pré-definidas para facilitar para o usuário e, assim, obter um *feedback* de forma mais objetiva. Dessa forma, pode-se melhorar as técnicas e a própria ferramenta de acordo com os *feedbacks* coletados.

Com os devidos recursos computacionais, futuramente pode-se disponibilizar a ferramenta *on-line*, colocando-a em produção, ou seja, permitir que desenvolvedores possam usá-la. A ferramenta, então, realizaria predições em tarefas de manutenção que estão em andamento e não foram concluídas nos diversos projetos de software livre. Dessa forma, a ferramenta pode disponibilizar predições e servir de apoio para todos os colaboradores, principalmente aos novatos de projetos de software livre.

REFERÊNCIAS

AGRAWAL, R.; IMIELIńSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. In: **Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93**. New York, New York, USA: ACM Press, 1993. p. 207–216.

ALMASRI, N.; TAHAT, L.; KOREL, B. Toward automatically quantifying the impact of a change in systems. **Software Quality Journal**, p. 1–40, 2016.

ARISHOLM, E.; BRIAND, L.; FOYEN, A. Dynamic coupling measurement for object-oriented software. **IEEE Transactions on Software Engineering**, v. 30, n. 8, p. 491–506, Ago 2004.

AVAZPOUR, I. et al. Dimensions and Metrics for Evaluating Recommendation Systems. **Recommendation Systems in Software Engineering**, p. 245–273, 2014.

BADRI, L.; BADRI, M.; ST-YVES, D. Supporting predictive change impact analysis: a control call graph based technique. In: **Software Engineering Conference, 2005. APSEC '05. 12th Asia-Pacific**. [S.l.: s.n.], 2005. p. 9.

BALL, T. et al. If Your Version Control System Could Talk... In: **ICSE Workshop on Process Modeling and Empirical Studies of Software Engineering**. [S.l.: s.n.], 1997.

BANDURA, A. **Social foundations of thought and action: a social cognitive theory**. [S.l.]: Prentice-Hall, 1986. (Prentice-Hall).

BANDURA, A. Social foundations of thought and action. **The health psychology reader**, Sage London, England, p. 94–106, 2002.

BENNETT, K. H.; RAJLICH, V. T. Software maintenance and evolution: A roadmap. In: **Proceedings of the Conference on The Future of Software Engineering**. New York, NY, USA: ACM, 2000. (ICSE '00), p. 73–87.

BETTENBURG, N.; HASSAN, A. E. Studying the impact of social interactions on software quality. **Empirical Softw. Engg.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 18, n. 2, p. 375–431, Abr 2013.

BIÇER, S.; BENER, A. B.; ÇAĞLAYAN, B. Defect prediction using social network analysis on issue repositories. In: **Proceedings of the International Conference on Software and Systems Process**. New York, New York, USA: ACM, 2011. (ICSSP '11), p. 63–71.

BIRD, C. et al. Putting It All Together: Using Socio-technical Networks to Predict Failures. In: **2009 20th International Symposium on Software Reliability Engineering**. Washington, DC, USA: IEEE, 2009. (ISSRE '09), p. 109–119.

BLUM, A. L.; LANGLEY, P. Selection of relevant features and examples in machine learning. **Artificial Intelligence**, v. 97, n. 1-2, p. 245–271, Dez 1997.

BOHNER, S. Software change impacts-an evolving perspective. **Proceedings of the International Conference on Software Maintenance**, p. 263–271, 2002.

BOHNER, S.; ARNOLD, R. **Software change impact analysis**. [S.l.]: IEEE Computer Society Press, 1996. (Practitioners Series).

BREIMAN, L. Random forests. **Machine learning**, p. 5–32, 2001.

BRIAND, L.; DALY, J.; WUST, J. A unified framework for coupling measurement in object-oriented systems. **Software Engineering, IEEE Transactions on**, v. 25, n. 1, p. 91–121, Jan 1999.

BRIAND, L.; WUST, J.; LOUNIS, H. Using coupling measurement for impact analysis in object-oriented systems. In: **Proceedings of the International Conference on Software Maintenance**. [S.l.]: IEEE, 1999. p. 475–482.

BUCKNER, J. et al. JRipples: A Tool for Program Comprehension during Incremental Change. In: **13th International Workshop on Program Comprehension (IWPC'05)**. [S.l.]: IEEE, 2005. p. 149–152.

BURT, R. **Structural Holes: The Social Structure of Competition**. [S.l.]: Harvard University Press, 1995.

CANFORA, G. et al. Using multivariate time series and association rules to detect logical change coupling: An empirical study. In: **2010 IEEE International Conference on Software Maintenance**. [S.l.]: IEEE, 2010. p. 1–10.

CANFORA, G. et al. How changes affect software entropy: an empirical study. **Empirical Software Engineering**, v. 19, n. 1, p. 1–38, 2014.

CARMINES, E. G.; ZELLER, R. A. **Reliability and validity assessment**. [S.l.]: Sage publications, 1979.

CHEN, K.; RAJICH, V. Ripples: tool for change in legacy software. In: **Proceedings of the International Conference on Software Maintenance**. [S.l.: s.n.], 2001. p. 230–239.

CRONBACH, L. J. Coefficient alpha and the internal structure of tests. **Psychometrika**, v. 16, n. 3, p. 297–334, 1951.

D'AMBROS, M.; LANZA, M.; ROBBES, R. On the Relationship Between Change Coupling and Software Defects. In: **2009 16th Working Conference on Reverse Engineering**. Washington, DC, USA: IEEE, 2009. (WCRE '09), p. 135–144.

D'AMBROS, M.; LANZA, M.; ROBBES, R. An extensive comparison of bug prediction approaches. In: **Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on**. [S.l.: s.n.], 2010. p. 31–41.

DAVIDSON, J. L. et al. Older adults and free/open source software: A diary study of first-time contributors. In: **Proceedings of The International Symposium on Open Collaboration**. New York, NY, USA: ACM, 2014. (OpenSym '14), p. 5:1–5:10. ISBN 978-1-4503-3016-9.

DAVIS, F. D. **A technology acceptance model for empirically testing new end-user information systems: theory and results**. Tese (Doutorado) — Massachusetts Institute of Technology, Sloan School of Management, 1986.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS Q.**, Society for Information Management and The Management Information Systems Research Center, Minneapolis, MN, USA, v. 13, n. 3, p. 319–340, set. 1989.

DE LUCIA, A.; FASANO, F.; OLIVETO, R. Traceability management for impact analysis. **Frontiers of Software Maintenance (FoSM)**, p. 21–30, 2008.

DIT, B. et al. Impactminer: A tool for change impact analysis. In: **Companion Proceedings of the 36th International Conference on Software Engineering**. New York, NY, USA: ACM, 2014. (ICSE Companion 2014), p. 540–543.

DURDIK, Z. et al. Sustainability guidelines for long-living software systems. In: **Software Maintenance (ICSM), 2012 28th IEEE International Conference on**. [S.l.: s.n.], 2012. p. 517–526.

EASTERBROOK, S. et al. Selecting Empirical Methods for Software Engineering Research. **Guide to Advanced Empirical Software Engineering**, p. 285–311, 2008.

FERENC, R. et al. Columbus - reverse engineering tool and schema for c++. In: **Proceedings of the International Conference on Software Maintenance**. [S.l.: s.n.], 2002. p. 172–181.

FISHBEIN, M.; AJZEN, I. **Belief, attitude, intention, and behavior: an introduction to theory and research**. [S.l.]: Addison-Wesley Pub. Co., 1975. (Addison-Wesley series in social psychology).

GALL, H.; HAJEK, K.; JAZAYERI, M. Detection of logical coupling based on product release history. In: **Proceedings of the International Conference on Software Maintenance**. [S.l.]: IEEE Computer Society, 1998. p. 190–198.

GALLAGHER, K.; LYLE, J. Using program slicing in software maintenance. **Software Engineering, IEEE Transactions on**, v. 17, n. 8, p. 751–761, Ago 1991.

GETHERS, M. et al. Integrated impact analysis for managing software changes. In: **Proceedings of the 34th International Conference on Software Engineering**. Piscataway, NJ, USA: IEEE Press, 2012. (ICSE '12), p. 430–440.

GHOTRA, B.; MCINTOSH, S.; HASSAN, A. Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models. In: **Proceedings of the 37th International Conference on Software Engineering (ICSE 2015)**. [S.l.: s.n.], 2015.

GIGER, E. et al. Method-level bug prediction. In: **Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on**. [S.l.: s.n.], 2012. p. 171–180.

GODFREY, M. W.; GERMAN, D. M. The past, present, and future of software evolution. In: **Frontiers of Software Maintenance (FoSM)**. [S.l.: s.n.], 2008. p. 129–138.

GRANGER, C. W. J. Investigating causal relations by econometric models and cross-spectral methods. **Econometrica**, Wiley, Econometric Society, v. 37, n. 3, p. 424–438, 1969.

GUYON, I. et al. An introduction to variable and feature selection. **Journal of Machine Learning Research**, v. 3, n. 1, p. 1157–1182, 2003.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining**. 3ª Edição. Boston: Morgan Kaufmann, Elsevier, 2012. 744 p.

HASSAN a.E.; HOLT, R. Predicting change propagation in software systems. In: **Proceedings of the International Conference on Software Maintenance**. [S.l.]: IEEE, 2004. p. 284–293.

HASSOUN, Y.; JOHNSON, R.; COUNSELL, S. A dynamic runtime coupling metric for meta-level architectures. **Proceedings of Eighth European Conference on Software Maintenance and Reengineering (CSMR)**, p. 339–346, 2004.

HATTORI, L. et al. On the Precision and Accuracy of Impact Analysis Techniques. **Seventh IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008)**, p. 513–518, 2008.

IEEE. International Standard - ISO/IEC 14764 IEEE Std 14764-2006 Software Engineering – Software Life Cycle Processes – Maintenance. **ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998)**, p. 1–46, 2006.

IEEE. **Systems and software engineering – Vocabulary**. 1. ed. Suíça: [s.n.], 2010. 418 p.

KAGDI, H.; GETHERS, M.; POSHYVANYK, D. Integrating conceptual and logical cou-

plings for change impact analysis in software. **Empirical Software Engineering**, v. 18, n. 5, p. 933–969, Out 2013.

KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. **Artificial Intelligence**, v. 97, n. 1-2, p. 273–324, 1997.

LAITENBERGER, O.; DREYER, H. M. Evaluating the usefulness and the ease of use of a web-based inspection data collection tool. In: **Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262)**. [S.l.: s.n.], 1998. p. 122–132.

LAKOS, J. **Large-scale C++ Software Design**. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1996.

LARMAN, C. **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development**. 3. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

LAW, J.; ROTHERMEL, G. Whole program path-based dynamic impact analysis. In: **Proceedings of the 25th International Conference on Software Engineering**. Washington, DC, USA: IEEE Computer Society, 2003. (ICSE '03), p. 308–318.

LI, B. et al. A survey of code-based change impact analysis techniques. **Software Testing, Verification and Reliability**, v. 23, n. 8, p. 613–646, 2013.

MARANGUNIĆ NIKOLA E GRANIĆ, A. Technology acceptance model: a literature review from 1986 to 2013. **Universal Access in the Information Society**, v. 14, n. 1, p. 81–95, 2015.

MENEELY, A. et al. 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**. New York, New York, USA: ACM Press, 2008. (SIGSOFT '08/FSE-16), p. 13.

MENS, T. et al. Challenges in software evolution. In: **Eighth International Workshop on Principles of Software Evolution (IWPSE'05)**. [S.l.: s.n.], 2005. p. 13–22.

OLIVA, G. A.; GEROSA, M. A. Experience report: How do structural dependencies influence change propagation? an empirical study. In: **Proceedings of the 26th IEEE International Symposium on Software Reliability Engineering**. [S.l.: s.n.], 2015. (ISSRE 2015).

OLIVA, G. A. et al. Towards a classification of logical dependencies origins. In: **Proceedings of the 12th international workshop and the 7th annual ERCIM workshop on Principles on software evolution and software evolution - IWPSE-EVOL '11**. New York, New York, USA: ACM Press, 2011. p. 31–40.

ORSO, A. et al. An empirical comparison of dynamic impact analysis algorithms. In: **Proceedings. 26th International Conference on Software Engineering**. [S.l.]: IEEE Comput. Soc, 2004. p. 491–500.

PANICHELLA, S. et al. How Developers' Collaborations Identified from Different Sources Tell Us about Code Changes. **IEEE International Conference on Software Maintenance and Evolution**, p. 251–260, 2014.

PARNAS, D. Designing Software for Ease of Extension and Contraction. **IEEE Transactions on Software Engineering**, SE-5, n. 2, p. 128–138, 1979.

POSHYVANYK, D.; MARCUS, A. The Conceptual Coupling Metrics for Object-Oriented Systems. In: **22nd IEEE International Conference on Software Maintenance**. [S.l.]: IEEE, 2006. p. 469–478.

POSHYVANYK, D. et al. Using information retrieval based coupling measures for impact analysis. **Empirical Softw. Engg.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 14, n. 1, p. 5–32, Feb 2009.

POWERS, D. M. W. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. In: . Adelaide, Australia: [s.n.], 2007.

PRESSMAN, R. S. **Software Engineering: A Practitioner's Approach**. 6. ed. New York, NY, USA: McGraw-Hill, Inc., 2005.

QUINLAN, J. R. Induction of decision trees. **Machine Learning**, v. 1, n. 1, p. 81–106, 1986.

R DEVELOPMENT CORE TEAM. **R: A Language and Environment for Statistical Computing**. Vienna, Austria, 2008. ISBN 3-900051-07-0. Disponível em: <http://www.R-project.org>.

RAHMAN, F.; DEVANBU, P. How, and why, process metrics are better. In: **35th International Conference on Software Engineering**. [S.l.]: IEEE, 2013. p. 432–441.

RAJLICH, V. A model for change propagation based on graph rewriting. In: **Proceedings of the International Conference on Software Maintenance**. [S.l.: s.n.], 1997. p. 84–91.

RAJLICH, V. Software evolution and maintenance. In: **Proceedings of the on Future of Software Engineering**. New York, NY, USA: ACM, 2014. (FOSE 2014), p. 133–144.

REN, X. et al. Chianti: A tool for change impact analysis of java programs. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 39, n. 10, p. 432–448, out. 2004. ISSN 0362-1340.

SCHNEIDEWIND, N. The state of software maintenance. **Software Engineering, IEEE Transactions on**, SE-13, n. 3, p. 303–310, Mar 1987.

STEINMACHER, I. et al. Overcoming open source project entry barriers with a portal for newcomers. In: **Proceedings of the 38th International Conference on Software Engineering**. New York, NY, USA: ACM, 2016. (ICSE '16), p. 273–284. ISBN 978-1-4503-3900-1.

STEINMACHER, I. et al. Increasing the self-efficacy of newcomers to open source software projects. In: **2015 29th Brazilian Symposium on Software Engineering**. [S.l.: s.n.], 2015. p. 160–169.

STEINMACHER, I. F. **Supporting newcomers to overcome the barriers to contribute to open source software projects.** Tese (Doutorado) — Instituto de Matemática e Estatística, Universidade de São Paulo, 2015.

SUN, X. et al. Static change impact analysis techniques. **J. Syst. Softw.**, Elsevier Science Inc., New York, NY, USA, v. 109, n. C, p. 137–149, Nov 2015.

SUN, X. et al. Change impact analysis based on a taxonomy of change types. In: **Com-**

puter Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual. [S.l.: s.n.], 2010. p. 373–382.

TONELLA, P. Using a concept lattice of decomposition slices for program understanding and impact analysis. **IEEE Transactions on Software Engineering**, v. 29, n. 6, p. 495–509, 2003.

VENKATESH, V. et al. User acceptance of information technology: Toward a unified view. **MIS Q.**, Society for Information Management and The Management Information Systems Research Center, Minneapolis, MN, USA, v. 27, n. 3, p. 425–478, set. 2003.

WAND, Y.; WEBER, R. An ontological model of an information system. **Software Engineering, IEEE Transactions on**, v. 16, n. 11, p. 1282–1292, Nov 1990.

WEISER, M. Program slicing. In: **Proceedings of the 5th International Conference on Software Engineering**. Piscataway, NJ, USA: IEEE Press, 1981. (ICSE '81), p. 439–449.

WIESE, I. S. et al. Using structural holes metrics from communication networks to predict change dependencies. In: BALOIAN, N. et al. (Ed.). **20th International Conference on Collaboration and Technology (CRIWG 2014)**. [S.l.]: Springer International Publishing, 2014. (Lecture Notes in Computer Science, v. 8658), p. 294–310.

WIESE, I. S. et al. Do historical metrics and developers communication aid to predict change couplings? **IEEE Latin America Transactions**, v. 13, n. 6, p. 1979–1988, June 2015.

WIESE, I. S. et al. An empirical study of the relation between strong change coupling and defects using history and social metrics in the apache aries project. In: _____. **Open Source Systems: Adoption and Impact: 11th IFIP WG 2.13 International Conference, OSS 2015, Florence, Italy, May 16-17, 2015, Proceedings**. Cham: Springer International Publishing, 2015. p. 3–12.

WIESE, I. S. et al. Informações contextuais do desenvolvimento de software na predição de propagação de mudanças. In: **XXIX Simpósio Brasileiro de Engenharia de Software (SBES 2015)**. [S.l.: s.n.], 2015. (Congresso Brasileiro de Software: Teoria e Prática (CBSoft)).

WIESE, I. S. et al. Using contextual information to predict co-changes. **Journal of Sys-**

tems and Software, v. 128, p. 220 – 235, 2017.

WOLF, T. et al. Predicting build failures using social network analysis on developer communication. In: **Proceedings of the 31st International Conference on Software Engineering**. Washington, DC, USA: IEEE Computer Society, 2009. (ICSE '09), p. 1–11.

YING, A. T. T. et al. Predicting Source Code Changes by Mining Change History. **IEEE Transactions on Software Engineering**, IEEE Press, Piscataway, NJ, USA, v. 30, n. 9, p. 574–586, 2004.

YU, Z.; RAJLICH, V. Hidden dependencies in program comprehension and change propagation. In: **Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on**. [S.l.: s.n.], 2001. p. 293–299.

ZHOU, Y. et al. A bayesian network based approach for change coupling prediction. In: **Proceedings of the 2008 15th Working Conference on Reverse Engineering**. Washington, DC, USA: IEEE Computer Society, 2008. (WCRE '08), p. 27–36.

ZIMMERMANN, T. et al. Mining version histories to guide software changes. **IEEE Transactions on Software Engineering**, v. 31, n. 6, p. 429–445, Jun 2005.

APÊNDICE A – FORMULÁRIO DO QUESTIONÁRIO TAM

Neste Apêndice, é apresentado o formulário do Questionário baseado no Modelo de Aceitação de Tecnologia aplicado no experimento conduzido nesse trabalho.

Formulário de Avaliação

Informe seu nome:

_____

☐ Concordo com a utilização dos dados deste formulário na pesquisa. Toda informação coletada é confidencial[1]

Identificação	**Usabilidade Percebida (Perceived Usefulness - PU)** **GRUPO: _____ | TAREFA: _____**	Discordo Totalmente	Discordo Parcialmente	Neutro	Concordo Parcialmente	Concordo Totalmente
U1	A ferramenta RECOMINER me fez encontrar mais *RAPIDAMENTE* os arquivos que devem ser modificados					
U2	Usar a ferramenta RECOMINER melhorou meu *DESEMPENHO* com o objetivo de encontrar arquivos que devem ser alterados juntos					
U3	Usar a ferramenta RECOMINER melhorou minha *PRODUTIVIDADE* para encontrar arquivos que mudam em conjunto					
U4	A utilização da ferramenta RECOMINER melhorou a EFETIVIDADE de encontrar arquivos que mudam em conjunto					
U5	Usar a ferramenta RECOMINER *FACILITOU* a documentação e gestão de casos de teste					
U6	A ferramenta RECOMINER foi *ÚTIL* para completar a tarefa proposta					

Identificação	**Facilidade de uso (Perceived Ease of Use - PEOU)**	Discordo Totalmente	Discordo Parcialmente	Neutro	Concordo Parcialmente	Concordo Totalmente
E1	Aprender a usar a RECOMINER foi fácil pra mim.					
E2	Eu acho que é fácil de obter as informações necessárias para encontrar os arquivos que precisam ser modificados para completar uma tarefa usando a RECOMINER.					
E3	Minha interação com a RECOMINER é clara e compreensível.					
E4	É fácil me tornar hábil (aprender a usar) a RECOMINER.					
E5	É fácil lembrar como usar a RECOMINER para encontrar os artefatos que devem ser modificados em uma tarefa.					
E6	Considero a RECOMINER fácil de usar.					

Identificação	**Uso no Futuro (Self-prediction of Future Use - SPFU)**	Discordo Totalmente	Discordo Parcialmente	Neutro	Concordo Parcialmente	Concordo Totalmente
S1	Supondo que a RECOMINER estivesse disponível para qualquer projeto, eu a usaria no futuro.					
S2	Eu prefiro usar a RECOMINER para encontrar os artefatos que devem ser modificados ao invés do "*find*/procurar" da IDE ou a função de *debug.*					

[1] Em caso de dúvida, por favor consulte o professor.

APÊNDICE B – FORMULÁRIO DO QUESTIONÁRIO DE AUTO-EFICÁCIA

Neste Apêndice, é apresentado o formulário do Questionário de Auto-Eficácia aplicado no experimento conduzido nesse trabalho.

Formulário de Avaliação

Informe seu nome:

_____

☐ Concordo com a utilização dos dados deste formulário na pesquisa. Toda informação coletada é confidencial[1]

Identificação	Auto-eficácia (Self-Efficacy – SE)	Sempre	Quase sempre	Algumas vezes	Raramente	Nunca
S1	Eu consigo ler uma *issue* (tarefa) e encontrar os artefatos a serem modificados					
S2	Quando altero um arquivo, eu consigo saber se outros arquivos devem ser alterados em conjunto					
S3	Eu tenho boas habilidades para escrever e mudar o código fonte					
S4	Dado que um alterei um arquivo, eu me sinto confortável elencando quais são outros potenciais arquivos que devem ser alterados para evitar problemas					

[1] Em caso de dúvida, por favor consulte o professor.

APÊNDICE C – TAREFAS APLICADAS NO EXPERIMENTO

Nesse Apêndice, são apresentados os detalhes das informações extraídas do Sistema de Gerenciamento de Solicitações de Mudanças do software livre Apache CXF. O sistema adotado pela comunidade do Apache CXF é o *Atlassian JIRA* e pode ser acessado pelo endereço https://issues.apache.org/jira/projects/CXF. As informações são das solicitações de mudanças CXF-3742, CXF-4220 e CXF-4872, e estão em inglês, pois optou-se por apresentar de forma como a solicitação de mudança é exibida no Jira.

[CXF-3742] Too many startServer() and clientDestroyed() got invoked when spring created and java code created client/server mixed used.	
Reference:	https://issues.apache.org/jira/browse/CXF-3742
Created:	15/Aug/11
Updated:	19/Oct/11
Resolved:	01/Sep/11
Status:	Closed
Project:	CXF
Component/s:	JAX-WS Runtime
Affects Version/s:	2.5
Fix Version/s:	2.4.3

Type:	Bug	**Priority:**	Major
Reporter:	Xilai Dai	**Assignee:**	Daniel Kulp
Resolution:	Fixed	**Votes:**	0
Labels:	CXF		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Environment:	Windows 7, JDK1.6		

Attachments:	testlifecycle_bundle.zip
Estimated Complexity:	Unknown

Description

From the testcase, there are 2 servers(one is spring created,the other is code created) and 2 clients(one is sprint created,the other is code created), when install/uninstall this bundle to osgi container, startServer() got invoked 3 times and clientDestroyed() got invoked 4 times!

Acesso em 16 de abril de 2017.

[CXF-4220] After loading XSDs from links in WADL, JAX-RS get all for all resources fail with 400	
Reference:	https://issues.apache.org/jira/browse/CXF-4220
Created:	02/Apr/12
Updated:	06/Apr/12
Resolved:	06/Apr/12
Status:	Closed
Project:	CXF
Component/s:	None
Affects Version/s:	2.5.2
Fix Version/s:	2.5.3, 2.6

Type:	Bug	**Priority:**	Major
Reporter:	Joanne Kubischta	**Assignee:**	Sergey Beryozkin
Resolution:	Fixed	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Environment:	Linux		

Estimated Complexity:	Unknown

Description

We use a CustomCXFNonSpringJaxrsServlet with one overriding method in order to use our hand written WADL.

```
@Override
protected void setSchemasLocations(JAXRSServerFactoryBean bean,
ServletConfig servletConfig)
{
super.setSchemasLocations(bean, servletConfig);

// Set the WADL document location
if(servletConfig.getInitParameter(JAXRSUtils.DOC_LOCATION) != null)

{ bean.setDocLocation(servletConfig.getInitParameter(
JAXRSUtils.DOC_LOCATION)); }

}
```

In the web.xml of the user resource, the XSDs are referenced as follows -
<init-param>
<param-name>jaxrs.schemaLocations</param-name>
<param-value>
classpath:schemas/xml.xsd
classpath:schemas/atom.xsd
classpath:schemas/user.xsd
classpath:schemas/user-role.xsd
</param-value>
</init-param>

In the wadl file, the grammar section is as follows -
<grammars>
<include href="xml.xsd" />
<include href="atom.xsd" />
<include href="user.xsd" />
<include href="user-role.xsd" />
</grammars>

Here are the steps to reproduce the error -

1. Verify get all and get a single resource work for all resources including user.
2. Access user wadl from https://server:port/api/rest/users?_wadl successfully but the grammar section is changed to -
<grammars>
<include href="https://server:8443/api/rest/users//xml.xsd" />
<include href="https://server:8443/api/rest/users//atom.xsd" />
<include href="https://server:8443/api/rest/users//user.xsd" />
<include href="https://server:8443/api/rest/users//user-role.xsd" />
</grammars>
3. Repeat step 1. Result is good.
4. Click the first link and view xml.xsd from a browser. Note xml.xsd is availale from the internet and does not have a schemaLocation tag.
5. Repeat step 1. Result is good.
6. Click the link to atom.xsd or any of rest of the links and the XSD loads correctly. Note all those XSDs are only available in the classpath of user.war. Below is an excerpt of how schema is referenced in the XSD -
<xs:import namespace="urn:api:rest:user-role"
schemaLocation="https://server:8443/api/rest/users//" />
<xs:import namespace="http://www.w3.org/2005/Atom"
schemaLocation="https://server:8443/api/rest/users//" />
7. Do a get all request using https://server:8443/api/rest/users and get the following error -
400: Bad Request
org.xml.sax.SAXParseException: Premature end of file.
8. Do a get all request to other resources and get the same 400.
9. Restart JBoss and problem goes away.

Acesso em 16 de abril de 2017.

[CXF-4872] JSONProvider can not drop a root element if DOM Document is copied to JSON	
Reference:	https://issues.apache.org/jira/browse/CXF-4872
Created:	04/Mar/13
Updated:	09/Apr/13
Resolved:	04/Mar/13
Status:	Closed
Project:	CXF
Component/s:	JAX-RS
Affects Version/s:	None
Fix Version/s:	2.7.4, 3.0.0-milestone1

Type:	Bug	**Priority:**	Minor
Reporter:	Sergey Beryozkin	**Assignee:**	Sergey Beryozkin
Resolution:	Fixed	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Estimated Complexity:	Unknown

Acesso em 16 de abril de 2017.

APÊNDICE D – PREDIÇÕES DE CADA TAREFAS E DE CADA TÉCNICA

Neste Apêndice, é apresentado o resultado das predições realizadas pelas técnicas de Regras de Associação e Classificação de cada tarefa que foi aplicado no experimento conduzido nesse trabalho. O resultado da predição (coluna RP) significa como **propenso a mudar**, quando o valor é igual a C, e **não é propenso a mudar** quando o valor é igual a N. Os artefatos com linhas destacadas são aqueles artefatos que foram modificados de fato para completar a tarefa.

Predições da técnica de Regras de Associação da Tarefa 1 apresentadas na ferramenta (1/1)

#	Artefato	Suporte	Confiança	RP
1	rt/frontend/jaxws/src/test/java/org/apache/cxf/jaxws/spring/clients.xml	4	0,571	C
2	rt/frontend/jaxws/src/test/java/org/apache/cxf/jaxws/spring/SpringBeansTest.java	4	0,571	C
3	rt/frontend/jaxws/src/test/java/org/apache/cxf/jaxws/spring/ClientHolderBean.java	3	0,429	C
4	rt/frontend/simple/src/main/java/org/apache/cxf/frontend/spring/ClientProxyFactoryBeanDefinitionParser.java	2	0,286	C
5	rt/frontend/jaxws/src/main/java/org/apache/cxf/jaxws/spring/JaxWsProxyFactoryBeanDefinitionParser.java	2	0,286	C
6	rt/frontend/simple/src/test/java/org/apache/cxf/frontend/spring/SpringBeansTest.java	2	0,286	C
7	rt/ws/rm/src/test/java/org/apache/cxf/ws/rm/rmmanager.xml	1	0,143	C
8	rt/core/src/main/resources/META-INF/cxf/cxf.xml	1	0,143	C
9	rt/frontend/jaxws/src/main/java/org/apache/cxf/jaxws/spring/EndpointDefinitionParser.java	1	0,143	C
10	common/common/src/main/java/org/apache/cxf/configuration/spring/AbstractBeanDefinitionParser.java	1	0,143	C
11	rt/core/src/main/java/org/apache/cxf/bus/spring/BusWiringBeanFactoryPostProcessor.java	1	0,143	C
12	common/common/src/main/java/org/apache/cxf/configuration/spring/BusWiringType.java	1	0,143	C
13	rt/transports/http-jetty/src/main/java/org/apache/cxf/transport/http_jetty/spring/JettyHTTPServerEngineFactoryBeanDefinitionParser.java	1	0,143	C
14	rt/transports/http-jetty/src/main/java/org/apache/cxf/transport/http_jetty/spring/JettyHTTPServerEngineBeanDefinitionParser.java	1	0,143	C
15	rt/ws/rm/src/main/java/org/apache/cxf/ws/rm/spring/RMManagerBeanDefinitionParser.java	1	0,143	C
16	rt/frontend/jaxws/src/test/java/org/apache/cxf/jaxws/spring/ClientHolderBean.java	1	0,143	C

Predições da técnica de Regras de Associação da Tarefa 2 apresentadas na ferramenta (1/2)

#	Artefato	Suporte	Confiança	RP
1	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/WadlGeneratorTest.java	27	0,659	C
2	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/BookStore.java	15	0,366	C
3	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/jaxb/Book.java	10	0,244	C
4	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/provider/JAXBElementProviderTest.java	6	0,146	C
5	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/ResourceUtils.java	6	0,146	C
6	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/FormInterface.java	6	0,146	C
7	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/AbstractJAXBProvider.java	4	0,098	C
8	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/BookStoreWithSingleSlash.java	4	0,098	C
9	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/jaxb/Chapter.java	3	0,073	C
10	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/JAXRSUtils.java	3	0,073	C
11	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/impl/RequestPreprocessor.java	3	0,073	C
12	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/BookStore.java	3	0,073	C
13	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/HttpUtils.java	3	0,073	C
14	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/URITemplate.java	3	0,073	C
15	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/utils/HttpUtilsTest.java	3	0,073	C
16	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/URITemplateTest.java	3	0,073	C
17	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerBookTest.java	3	0,073	C
18	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/atom/AtomFeedProvider.java	2	0,049	N
19	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/atom/AtomEntryProvider.java	2	0,049	N
20	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/servlet/CXFNonSpringJaxrsServlet.java	2	0,049	N
21	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/interceptor/JAXRSOutInterceptor.java	2	0,049	N
22	systests/jaxrs/src/test/resources/jaxrs_atom/WEB-INF/beans.xml	2	0,049	N
23	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerResourceCreatedSpringProviderTest.java	2	0,049	N
24	rt/rs/extensions/providers/src/test/java/org/apache/cxf/jaxrs/provider/json/JSONProviderTest.java	2	0,049	N
25	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/wadl/Description.java	2	0,049	N
26	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/AbstractJAXRSFactoryBean.java	2	0,049	N
27	api/src/main/java/org/apache/cxf/common/jaxb/JAXBUtils.java	2	0,049	N
28	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/spring/JAXRSServerFactoryBeanTest.java	2	0,049	N
29	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/AbstractConfigurableProvider.java	2	0,049	N
30	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/ProviderFactory.java	2	0,049	N

Predições da técnica de Regras de Associação da Tarefa 2 apresentadas na ferramenta (2/2)

#	Artefato	Suporte	Confiança	RP
1	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/ProviderFactory.java	2	0,049	N
2	rt/databinding/jaxb/src/main/java/org/apache/cxf/jaxb/JAXBDataBinding.java	2	0,049	N
2	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/spring/servers.xml	2	0,049	N
3	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerSpringBookTest.java	2	0,049	N
4	systests/jaxrs/src/test/resources/jaxrs/WEB-INF/beans.xml	2	0,049	N
5	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSSoapBookTest.java	2	0,049	N
6	rt/transports/http/src/main/java/org/apache/cxf/transport/servlet/servicelist/UnformattedServiceListWriter.java	2	0,049	N
7	rt/transports/http/src/main/java/org/apache/cxf/transport/servlet/servicelist/FormattedServiceListWriter.java	2	0,049	N
8	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/JAXBElementProvider.java	2	0,049	N
9	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/impl/RequestPreprocessorTest.java	2	0,049	N
10	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/resources/Book.java	1	0,024	N
11	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/impl/UriInfoImpl.java	1	0,024	N
12	api/src/test/java/org/apache/cxf/common/util/XmlSchemaPrimitiveUtilsTest.java	1	0,024	N
13	rt/transports/http/src/test/java/org/apache/cxf/transport/servlet/ServletControllerTest.java	1	0,024	N
14	api/src/main/java/org/apache/cxf/common/util/XmlSchemaPrimitiveUtils.java	1	0,024	N
15	rt/transports/http/src/main/java/org/apache/cxf/transport/servlet/ServletController.java	1	0,024	N
16	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/Orders.java	1	0,024	N
17	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/jaxb/ObjectFactory.java	1	0,024	N
18	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/ext/Description.java	1	0,024	N
19	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/AbstractAtomProvider.java	1	0,024	N
20	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/atom/AbstractAtomProvider.java	1	0,024	N
21	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerResourceJacksonSpringProviderTest.java	1	0,024	N
22	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/wadl/ElementClass.java	1	0,024	N
23	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/wadl/XMLName.java	1	0,024	N
24	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/wadl/DocTarget.java	1	0,024	N
25	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/wadl/Descriptions.java	1	0,024	N
26	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/JAXRSServerFactoryBean.java	1	0,024	N
27	systests/jaxrs/src/test/resources/jaxrs_soap_rest/WEB-INF/beans.xml	1	0,024	N
28	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/BookEnum.java	1	0,024	N

Predições da técnica de Regras de Associação da Tarefa 3 apresentadas na ferramenta (1/2)

#	Artefato	Suporte	Confiança	RP
1	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/AbstractJAXBProvider.java	24	0,511	C
2	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/JAXBElementProvider.java	22	0,468	C
3	rt/rs/extensions/providers/src/test/java/org/apache/cxf/jaxrs/provider/json/JSONProviderTest.java	18	0,383	C
4	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerBookTest.java	10	0,213	C
5	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/InjectionUtils.java	10	0,213	C
6	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/json/utils/JSONUtils.java	7	0,149	C
7	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/provider/JAXBElementProviderTest.java	6	0,128	C
8	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/BookStore.java	5	0,106	C
9	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/PrimitiveTextProvider.java	4	0,085	C
10	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/Customer.java	4	0,085	C
11	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/utils/JAXRSUtilsTest.java	4	0,085	C
12	parent/pom.xml	3	0,064	N
13	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/utils/InjectionUtilsTest.java	2	0,043	N
14	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/provider/PrimitiveTextProviderTest.java	2	0,043	N
15	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/HttpUtils.java	2	0,043	N
16	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/ResourceUtils.java	2	0,043	N
17	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/JAXBUtils.java	2	0,043	N
18	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/JAXRSUtils.java	2	0,043	N
19	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/json/DataBindingJSONProvider.java	2	0,043	N
20	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/aegis/AegisJSONProvider.java	2	0,043	N
21	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/provider/SourceProviderTest.java	2	0,043	N
22	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/SourceProvider.java	2	0,043	N
23	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/BookServer.java	2	0,043	N
24	api/src/main/java/org/apache/cxf/staxutils/StaxUtils.java	2	0,043	N
25	rt/rs/extensions/providers/src/test/java/org/apache/cxf/jaxrs/resources/jaxb/package-info.java	2	0,043	N
26	rt/transports/http-jetty/src/main/java/org/apache/cxf/transport/http_jetty/JettyHTTPServerEngineFactory.java	2	0,043	N
27	api/src/main/java/org/apache/cxf/common/util/ASMHelper.java	2	0,043	N
28	rt/transports/local/src/main/java/org/apache/cxf/transport/local/LocalTransportFactory.java	2	0,043	N
29	api/src/main/java/org/apache/cxf/endpoint/ClientImpl.java	2	0,043	N

Predições da técnica de Regras de Associação da Tarefa 3 apresentadas na ferramenta (2/2)

#	Artefato	Suporte	Confiança	RP
1	api/src/main/java/org/apache/cxf/helpers/HttpHeaderHelper.java	2	0,043	N
2	rt/features/clustering/src/main/java/org/apache/cxf/clustering/FailoverTargetSelector.java	2	0,043	N
3	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/ClassResourceInfo.java	2	0,043	N
4	rt/core/src/main/java/org/apache/cxf/bus/osgi/CXFExtensionBundleListener.java	2	0,043	N
5	api/src/main/java/org/apache/cxf/configuration/spring/AbstractSpringBeanMap.java	2	0,043	N
6	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/json/utils/PrefixRespectingMappedNamespaceConvention.java	2	0,043	N
7	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerSpringBookTest.java	2	0,043	N
8	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/fortest/jaxb/ObjectFactory.java	1	0,021	N
9	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/fortest/jaxb/Book.java	1	0,021	N
10	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/FormatResponseHandler.java	1	0,021	N
11	common/common/src/main/java/org/apache/cxf/staxutils/DocumentDepthProperties.java	1	0,021	N
12	common/common/src/main/java/org/apache/cxf/staxutils/DepthRestrictingStreamReader.java	1	0,021	N
13	api/src/main/java/org/apache/cxf/staxutils/DepthRestrictingStreamReader.java	1	0,021	N
14	api/src/main/java/org/apache/cxf/staxutils/DocumentDepthProperties.java	1	0,021	N
15	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/fortest/jaxb/packageinfo/package-info.java	1	0,021	N
16	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/WadlGeneratorTest.java	1	0,021	N
17	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/impl/AbstractResponseContextImpl.java	1	0,021	N
18	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/ext/Nullable.java	1	0,021	N
19	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/ProviderFactory.java	1	0,021	N
20	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/AbstractConfigurableProvider.java	1	0,021	N

Predições da técnica de Classificação da Tarefa 1 apresentadas na ferramenta (1/1)

#	Artefato	Probabilidade	RP
1	rt/ws/rm/src/test/java/org/apache/cxf/ws/rm/rmmanager.xml	0,558	C
2	common/common/src/main/java/org/apache/cxf/configuration/spring/AbstractBeanDefinitionParser.java	0,544	C
3	rt/ws/rm/src/main/java/org/apache/cxf/ws/rm/spring/RMManagerBeanDefinitionParser.java	0,536	C
4	rt/transports/http-jetty/src/main/java/org/apache/cxf/transport/http_jetty/spring/JettyHTTPServerEngineFactoryBeanDefinitionParser.java	0,532	C
5	rt/core/src/main/java/org/apache/cxf/bus/spring/BusWiringBeanFactoryPostProcessor.java	0,530	C
6	rt/transports/http-jetty/src/main/java/org/apache/cxf/transport/http_jetty/spring/JettyHTTPServerEngineBeanDefinitionParser.java	0,520	C
7	rt/core/src/main/resources/META-INF/cxf/cxf.xml	0,514	C
8	rt/frontend/jaxws/src/main/java/org/apache/cxf/jaxws/spring/EndpointDefinitionParser.java	0,510	C
9	common/common/src/main/java/org/apache/cxf/configuration/spring/BusWiringType.java	0,510	C
10	rt/frontend/jaxws/src/test/java/org/apache/cxf/jaxws/spring/ClientHolderBean.java	0,946	N
11	rt/frontend/jaxws/src/main/java/org/apache/cxf/jaxws/spring/JaxWsProxyFactoryBeanDefinitionParser.java	0,914	N
12	rt/frontend/simple/src/test/java/org/apache/cxf/frontend/spring/SpringBeansTest.java	0,900	N
13	rt/frontend/simple/src/main/java/org/apache/cxf/frontend/spring/ClientProxyFactoryBeanDefinitionParser.java	0,898	N
14	rt/frontend/jaxws/src/test/java/org/apache/cxf/jaxws/spring/SpringBeansTest.java	0,794	N
15	rt/frontend/jaxws/src/test/java/org/apache/cxf/jaxws/spring/ClientHolderBean.java	0,756	N
16	rt/frontend/jaxws/src/test/java/org/apache/cxf/jaxws/spring/clients.xml	0,724	N

Predições da técnica de Classificação da Tarefa 2 apresentadas na ferramenta (1/3)

#	Artefato	Probabilidade	RP
1	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/BookStore.java	0,740	C
2	api/src/main/java/org/apache/cxf/common/jaxb/JAXBUtils.java	1,000	N
3	rt/databinding/jaxb/src/main/java/org/apache/cxf/jaxb/JAXBDataBinding.java	1,000	N
4	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/JAXRSServerFactoryBean.java	1,000	N
5	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/ProviderFactory.java	1,000	N
6	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/impl/UriInfoImpl.java	1,000	N
7	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/resources/Book.java	1,000	N
8	rt/transports/http/src/main/java/org/apache/cxf/transport/servlet/ServletController.java	1,000	N
9	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerSpringBookTest.java	1,000	N
10	systests/jaxrs/src/test/resources/jaxrs/WEB-INF/beans.xml	1,000	N
11	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/AbstractConfigurableProvider.java	1,000	N
12	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/AbstractJAXRSFactoryBean.java	1,000	N
13	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/spring/JAXRSServerFactoryBeanTest.java	1,000	N
14	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/spring/servers.xml	1,000	N
15	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/Orders.java	1,000	N
16	api/src/main/java/org/apache/cxf/common/util/XmlSchemaPrimitiveUtils.java	1,000	N
17	api/src/test/java/org/apache/cxf/common/util/XmlSchemaPrimitiveUtilsTest.java	1,000	N
18	rt/transports/http/src/test/java/org/apache/cxf/transport/servlet/ServletControllerTest.java	1,000	N
19	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/wadl/XMLName.java	1,000	N
20	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/wadl/ElementClass.java	1,000	N
21	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/wadl/Descriptions.java	1,000	N
22	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/wadl/DocTarget.java	1,000	N
23	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/BookEnum.java	1,000	N
24	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerBookTest.java	0,550	N
25	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/URITemplateTest.java	0,558	N
26	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/URITemplate.java	0,562	N
27	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/BookStore.java	0,582	N
28	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/HttpUtils.java	0,586	N
29	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/utils/HttpUtilsTest.java	0,586	N
30	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/jaxb/Book.java	0,716	N

Predições da técnica de Classificação da Tarefa 2 apresentadas na ferramenta (2/3)

#	Artefato	Probabilidade	RP
1	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/FormInterface.java	0,810	N
2	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/provider/JAXBElementProviderTest.java	0,874	N
3	rt/rs/extensions/providers/src/test/java/org/apache/cxf/jaxrs/provider/json/JSONProviderTest.java	0,920	N
4	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/JAXRSUtils.java	0,930	N
5	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/servlet/CXFNonSpringJaxrsServlet.java	0,946	N
6	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/atom/AtomFeedProvider.java	0,948	N
7	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/WadlGeneratorTest.java	0,948	N
8	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/interceptor/JAXRSOutInterceptor.java	0,950	N
9	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerResourceCreatedSpringProviderTest.java	0,952	N
10	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerResourceJacksonSpringProviderTest.java	0,956	N
11	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/atom/AtomEntryProvider.java	0,958	N
12	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/AbstractJAXBProvider.java	0,958	N
13	systests/jaxrs/src/test/resources/jaxrs_atom/WEB-INF/beans.xml	0,960	N
14	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/ext/Description.java	0,960	N
15	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/jaxb/Chapter.java	0,962	N
16	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/AbstractAtomProvider.java	0,962	N
17	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/atom/AbstractAtomProvider.java	0,968	N
18	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/jaxb/ObjectFactory.java	0,970	N
19	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/impl/RequestPreprocessor.java	0,972	N
20	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/jaxb/ObjectFactory.java	0,972	N
21	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/ResourceUtils.java	0,978	N
22	rt/transports/http/src/main/java/org/apache/cxf/transport/servlet/servicelist/UnformattedServiceListWriter.java	0,978	N
23	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/wadl/Description.java	0,982	N
24	rt/transports/http/src/main/java/org/apache/cxf/transport/servlet/servicelist/FormattedServiceListWriter.java	0,982	N
25	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/JAXBElementProvider.java	0,986	N
26	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/impl/RequestPreprocessorTest.java	0,990	N
27	systests/jaxrs/src/test/resources/jaxrs_soap_rest/WEB-INF/beans.xml	0,992	N
28	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSSoapBookTest.java	0,998	N
29	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/BookStoreWithSingleSlash.java	0,998	N
30	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/BookEnum.java	0,998	N

Predições da técnica de Classificação da Tarefa 3 apresentadas na ferramenta (1/2)

#	Artefato	Probabilidade	RP
1	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerBookTest.java	0,778	C
2	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/JAXBElementProvider.java	0,616	C
3	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/AbstractJAXBProvider.java	0,614	C
4	rt/rs/extensions/providers/src/test/java/org/apache/cxf/jaxrs/provider/json/JSONProviderTest.java	0,564	C
5	api/src/main/java/org/apache/cxf/configuration/spring/AbstractSpringBeanMap.java	0,998	N
6	rt/features/clustering/src/main/java/org/apache/cxf/clustering/FailoverTargetSelector.java	0,998	N
7	api/src/main/java/org/apache/cxf/endpoint/ClientImpl.java	0,998	N
8	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/JAXRSUtils.java	0,998	N
9	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/model/ClassResourceInfo.java	0,998	N
10	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/utils/JAXRSUtilsTest.java	0,998	N
11	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/JAXBUtils.java	0,998	N
12	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/FormatResponseHandler.java	0,998	N
13	rt/core/src/main/java/org/apache/cxf/bus/osgi/CXFExtensionBundleListener.java	0,998	N
14	rt/rs/extensions/providers/src/test/java/org/apache/cxf/jaxrs/resources/jaxb/package-info.java	0,998	N
15	common/common/src/main/java/org/apache/cxf/staxutils/DocumentDepthProperties.java	0,996	N
16	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/FormatResponseHandler.java	0,996	N
17	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/BookServer.java	0,994	N
18	api/src/main/java/org/apache/cxf/staxutils/DepthRestrictingStreamReader.java	0,994	N
19	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/provider/JAXBElementProviderTest.java	0,992	N
20	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/model/wadl/WadlGeneratorTest.java	0,992	N
21	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/fortest/jaxb/packageinfo/package-info.java	0,992	N
22	common/common/src/main/java/org/apache/cxf/staxutils/DepthRestrictingStreamReader.java	0,992	N
23	api/src/main/java/org/apache/cxf/staxutils/DocumentDepthProperties.java	0,992	N
24	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/Customer.java	0,990	N
25	api/src/main/java/org/apache/cxf/staxutils/StaxUtils.java	0,988	N
26	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/BookStore.java	0,976	N
27	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/json/utils/PrefixRespectingMappedNamespaceConvention.java	0,924	N
28	systests/jaxrs/src/test/java/org/apache/cxf/systest/jaxrs/JAXRSClientServerSpringBookTest.java	0,922	N

Predições da técnica de Classificação da Tarefa 3 apresentadas na ferramenta (2/2)

#	Artefato	Probabilidade	RP
1	parent/pom.xml	0,874	N
2	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/ext/Nullable.java	0,774	N
3	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/AbstractConfigurableProvider.java	0,754	N
4	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/impl/AbstractResponseContextImpl.java	0,754	N
5	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/ProviderFactory.java	0,736	N
6	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/json/utils/JSONUtils.java	0,526	N
7	api/src/main/java/org/apache/cxf/common/util/ASMHelper.java	1,000	N
8	api/src/main/java/org/apache/cxf/helpers/HttpHeaderHelper.java	1,000	N
9	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/provider/SourceProvider.java	1,000	N
10	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/provider/SourceProviderTest.java	1,000	N
11	rt/transports/http-jetty/src/main/java/org/apache/cxf/transport/http_jetty/JettyHTTPServerEngineFactory.java	1,000	N
12	rt/transports/local/src/main/java/org/apache/cxf/transport/local/LocalTransportFactory.java	1,000	N
13	rt/transports/local/src/main/java/org/apache/cxf/transport/local/LocalTransportFactory.java	1,000	N
14	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/InjectionUtils.java	1,000	N
15	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/HttpUtils.java	1,000	N
16	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/provider/PrimitiveTextProviderTest.java	1,000	N
17	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/ResourceUtils.java	1,000	N
18	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/fortest/jaxb/Book.java	1,000	N
19	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/fortest/jaxb/ObjectFactory.java	1,000	N
20	rt/frontend/jaxrs/src/test/java/org/apache/cxf/jaxrs/utils/InjectionUtilsTest.java	1,000	N
21	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/aegis/AegisJSONProvider.java	1,000	N
22	rt/rs/extensions/providers/src/main/java/org/apache/cxf/jaxrs/provider/json/DataBindingJSONProvider.java	1,000	N
23	rt/frontend/jaxrs/src/main/java/org/apache/cxf/jaxrs/utils/JAXBUtils.java	1,000	N