RECOMENDAÇÃO DE CONTEXTO DE TAREFA EM DESENVOLVIMENTO DE SOFTWARE

Edson Mello Lucas

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Toacy Cavalcante de Oliveira

Rio de Janeiro
Junho de 2019

RECOMENDAÇÃO DE CONTEXTO DE TAREFA
EM DESENVOLVIMENTO DE SOFTWARE

Edson Mello Lucas

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Toacy Cavalcante de Oliveira, D.Sc.

Profa. Claudia Maria Lima Werner, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

Prof. Márcio de Oliveira Barros, D.Sc.

Prof. Elder José Reioli Cirilo, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
JUNHO DE 2019

Lucas, Edson Mello

Recomendação de Contexto de Tarefa em Desenvolvimento de Software / Edson Mello Lucas. – Rio de Janeiro: UFRJ/COPPE, 2019.

XIV, 141 p.: il.; 29,7 cm.

Orientador: Toacy Cavalcante de Oliveira

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 116-125.

1. Recomendação. 2. Contexto. 3. Desenvolvimento de Software. I. Oliveira, Toacy Cavalcante de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

A Deus pela vida e presença.

Aos meus pais Lecy (*in memorian*) e Edma pelo ensinamento dos valores que me fortalecem a cada conquista.

À minha esposa Nina, aos meus filhos Lucas e Mateus, pelo apoio e por entenderem a minha falta física em alguns momentos.

Aos diversos alunos, professores e técnico-administrativos que tive o prazer de conviver nesses muitos anos de dedicação ao Instituto Politécnico da Uerj (IPRJ), em especial aos professores Francisco Moura Neto, Pedro Watts e Ivan Bastos pelo apoio institucional e incentivo.

A todos os professores e companheiros das disciplinas que tive o privilégio de conviver no PESC/COPPE, em especial aos professores Guilherme Horta, Cláudia Werner e Ana Regina pelas frases de efeito que me fizeram refletir.

Ao pessoal administrativo e de suporte do PESC/COPPE que sempre deram o apoio necessário para o cumprimento das minhas obrigações acadêmicas.

Aos técnicos e gestores do Núcleo Avançado de Computação de Alto Desempenho (NACAD) da COPPE, Universidade Federal do Rio de Janeiro (UFRJ), pelo trabalho para disponibilizar os recursos computacionais que foram necessários em uma parte da minha pesquisa.

Aos meus colegas do grupo Prisma, André Campos, Fábio Basso, Raquel Pillat, Talita Gomes, Ivens Portugal, Renata Mesquita, Alciléia Rocha, Gláucia Melo, André Brito, Ulisses Telemaco e Rachel Vital pelo compartilhamento dos problemas e soluções.

Ao Professor e ser humano Toacy pela orientação, paciência e dedicação ao longo de todo o processo de doutoramento.

A todos os coautores de artigos, em especial ao Prof. Paulo Alencar por indicar referências importantes para minha pesquisa e, também, ao Prof. Daniel Schneider pela colaboração e motivação.

Aos professores Márcio Barros e Geraldo Xexéo pela participação em minha banca de Qualificação e disponibilidade para o diálogo, resultando em decisões importantes para o prosseguimento da pesquisa, e também ao Marlon Monçores por disponibilizar a implementação do algoritmo LNS e pela ajuda no seu entendimento.

À Prof. Cláudia Werner e aos professores Geraldo Xexéo, Márcio Barros e Elder Cirilo por aceitarem o convite para a minha banca de defesa e contribuírem com questionamentos e sugestões visando melhorias em minha pesquisa.

Finalmente, a todas as pessoas que de alguma forma contribuíram social e/ou tecnicamente.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

RECOMENDAÇÃO DE CONTEXTO DE TAREFA EM DESENVOLVIMENTO DE SOFTWARE

Edson Mello Lucas

Junho/2019

Orientador: Toacy Cavalcante de Oliveira

Programa: Engenharia de Sistemas e Computação

O fluxo de trabalho do desenvolvimento de software geralmente envolve desenvolvedores executando tarefas e manipulando artefatos. Quando os desenvolvedores recebem uma nova tarefa, eles geralmente visualizam um contexto de tarefa incluindo as tarefas similares à nova tarefa, artefatos que serão necessários alterar, e caso isso não seja suficiente, tentam descobrir quem melhor pode ajudar, isso tudo com base em suas experiências passadas. Dado que os projetos de software podem durar vários meses, acumulando uma grande quantidade de tarefas, artefatos e desenvolvedores, a visualização desse contexto de tarefa inicial pode ser difícil e passível de erros. Este trabalho apresenta um método para ajudar os desenvolvedores a definir o contexto de uma nova tarefa, combinando informações de interações em artefatos com as descrições textuais das tarefas. Este trabalho também apresenta um conjunto de modelos projetados para mensurar o quanto os desenvolvedores sabem sobre elementos de projeto de software, tais como artefatos, tarefas, tarefas similares e todo o projeto de software, baseando-se nas interações dos desenvolvedores. A avaliação indica que o método produz recomendações satisfatórias combinando as informações de edição em artefatos com a descrição textual das tarefas. Os resultados também indicam que os modelos podem ajudar os desenvolvedores a encontrar especialistas em partes específicas do projeto de software.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

RECOMMENDATION OF TASK CONTEXT
IN SOFTWARE DEVELOPMENT

Edson Mello Lucas

June/2019

Advisor: Toacy Cavalcante de Oliveira

Department: Computer Science and Systems Engineering

The software development workflow typically involves developers executing tasks and manipulating artifacts. When developers receive a new task they typically envision a task context including tasks similar to the new task, artifacts that will need to be changed, and if that is not enough, they try to figure out who can best help based on their past experiences. Given software projects may last several months, accumulating a vast amount of tasks, artifacts and developers, envisioning this initial task context may be difficult and error-prone. This work presents a method that helps developers defining the initial task context by combining interaction information over artifacts with text information of tasks. This work also presents a set of models designed to measure how much developers know about software project elements, such as artifacts, tasks, similar tasks and the whole software project, based on developers' interactions. The evaluation indicates that the method produces satisfactory recommendations by combining the editing information into artifacts with the textual description of the tasks. The results also indicate that the models can help developers find experts in specific parts of the software project.

Sumário

1 - INTRODUÇÃO 1

1.1 OBJETIVOS 5
1.2 QUESTÕES DE PESQUISA 6
1.3 METODOLOGIA 7
1.4 ORGANIZAÇÃO 12

2 - FUNDAMENTAÇÃO TEÓRICA 14

2.1 PROJETOS NA ENGENHARIA DE SOFTWARE 15
2.2 SISTEMAS DE RECOMENDAÇÃO PARA A ENGENHARIA DE SOFTWARE 18
2.3 CLUSTERIZAÇÃO 21
2.4 PROCESSAMENTO DE LINGUAGEM NATURAL 23
2.5 PERCEPÇÃO 27
2.6 CONHECIMENTO 29
2.7 CONSIDERAÇÕES FINAIS 31

3 - MODELOS TACIN DE PERCEPÇÃO DO CONHECIMENTO 32

3.1 MEDIDA DE CONHECIMENTO DO DESENVOLVEDOR SOBRE UM ARTEFATO 36
3.2 MEDIDA DE CONHECIMENTO DO DESENVOLVEDOR SOBRE UMA TAREFA 40
3.3 MEDIDA DE CONHECIMENTO DO DESENVOLVEDOR SOBRE UM GRUPO DE TAREFAS SIMILARES 43
3.4 MEDIDA DE CONHECIMENTO DO DESENVOLVEDOR SOBRE UM PROJETO 44
3.5 CONSIDERAÇÕES FINAIS 45

4- MÉTODO TACIN PARA RECOMENDAR CONTEXTO DE NOVA TAREFA 47

4.1 RECOMENDAÇÃO DE TAREFAS SIMILARES 53
4.2 RECOMENDAÇÃO DE ARTEFATOS 54
4.3 RECOMENDAÇÃO DE DESENVOLVEDORES 55
4.4 REPRESENTAÇÃO DO TACIN 56
4.5 CONSIDERAÇÕES FINAIS 58

5 – AVALIAÇÃO 60

5.1 COLETA DE DADOS 60
5.2 AVALIAÇÃO DOS MODELOS TACIN DE PERCEPÇÃO DO CONHECIMENTO 63
5.2.1 Materiais e Métodos 65
5.2.2 Resultados e Análises 66
5.2.3 Ameaças à Validade 70

5.3 AVALIAÇÃO DO MÉTODO TACIN PARA RECOMENDAR CONTEXTO DE NOVA TAREFA . 71
5.3.1 Materiais e Métodos ... *73*
5.3.2 Resultados e Análises ... *74*
5.3.3 Ameaças à Validade ... *91*
5.4 CONSIDERAÇÕES FINAIS ... 92

6- TRABALHOS RELACIONADOS ... 93

6.1 APRESENTAÇÃO DOS TRABALHOS RELACIONADOS ... 94
6.2 ANÁLISE DOS TRABALHOS RELACIONADOS ... 101
6.3 CONSIDERAÇÕES FINAIS ... 103

7 - DISCUSSÃO E RESULTADOS ... 105

7.1 MODELOS TACIN ... 105
7.2 MÉTODO TACIN ... 108
7.3 RESULTADOS ... 110
7.4 CONSIDERAÇÕES FINAIS ... 111

8 - CONCLUSÃO E TRABALHOS FUTUROS ... 112

REFERÊNCIAS ... 116

APÊNDICE A: REVISÃO (QUASI-SISTEMÁTICA) DA LITERATURA ... 126

APÊNDICE B: IMPLEMENTAÇÃO DE REFERÊNCIA DOS MODELOS TACIN ... 138

APÊNDICE C: IMPLEMENTAÇÃO DE REFERÊNCIA DO MÉTODO TACIN ... 140

Lista de Equações

EQUAÇÃO 1 - DEFINIÇÃO DE COBERTURA. 20
EQUAÇÃO 2 - DEFINIÇÃO DE PRECISÃO. 20
EQUAÇÃO 3 - DEFINIÇÃO DE F-MEASURE. 21
EQUAÇÃO 4 - DEFINIÇÃO DE REDUÇÃO. 21
EQUAÇÃO 5 - DEFINIÇÃO DE RC-MEASURE. 21
EQUAÇÃO 6 – A DEFINIÇÃO DA MEDIDA MQ. 23
EQUAÇÃO 7 – INTERAÇÃO NORMALIZADA. 36
EQUAÇÃO 8 - FUNÇÃO DE ESQUECIMENTO. 37
EQUAÇÃO 9 – FUNÇÃO DE INTERAÇÃO RECENTE. 37
EQUAÇÃO 10 - MEDIDA DE CONHECIMENTO DO DESENVOLVEDOR SOBRE UM ARTEFATO. .. 40
EQUAÇÃO 11 - MEDIDA DE CONHECIMENTO DO DESENVOLVEDOR SOBRE UMA TAREFA..... 43
EQUAÇÃO 12 - MEDIDA DE CONHECIMENTO DO DESENVOLVEDOR SOBRE UM GRUPO DE TAREFAS SEMELHANTES. 43
EQUAÇÃO 13 - MEDIDA DE CONHECIMENTO DO DESENVOLVEDOR SOBRE UM PROJETO. 45
EQUAÇÃO 14 – CÁLCULO DO PERCENTUAL DE REDUÇÃO DE ARTEFATOS NA RECOMENDAÇÃO. 54
EQUAÇÃO 15 – MEDIDA DE COBERTURA PARA A RECOMENDAÇÃO DE ARTEFATOS. 54

Lista de Figuras

FIGURA 1 - VISÃO GERAL DO PROBLEMA INVESTIGADO. 4
FIGURA 2 - HISTÓRICO DE ELEMENTOS DO PROJETO. 5
FIGURA 3 - CONTEXTO DE TAREFA. 6
FIGURA 4 - MODELO CONCEITUAL DO NÚCLEO DA ESPECIFICAÇÃO PROV-DM DA W3C. ... 17
FIGURA 5 – INTERAÇÕES DESENVOLVEDOR-ARTEFATO E DESENVOLVEDOR-DESENVOLVEDOR. 33
FIGURA 6 – EXEMPLO DE INTERAÇÃO DESENVOLVEDOR-ARTEFATO..... 34
FIGURA 7 - EXEMPLO DE INTERAÇÃO DESENVOLVEDOR-DESENVOLVEDOR..... 34
FIGURA 8 – EXEMPLO DE TAREFAS SIMILARES POR INTERAÇÃO DESENVOLVEDOR-ARTEFATO..... 35
FIGURA 9 - EXEMPLOS DE INTERAÇÕES DESENVOLVEDOR-ARTEFATO. 37
FIGURA 10 – O GRÁFICO DE Gd. 38
FIGURA 11 - GRÁFICO DE Rd ATÉ 10 DIAS. 38
FIGURA 12 - GRÁFICO DE Rd ATÉ 40 DIAS. 38
FIGURA 13 - GRÁFICO DE Rd ATÉ 100 DIAS. 39
FIGURA 14 - GRÁFICO DE Rd ATÉ UM ANO. 39
FIGURA 15 – ILUSTRAÇÃO DE INTERAÇÕES DESENVOLVEDOR-ARTEFATO PARA REALIZAR UMA TAREFA. 41
FIGURA 16 - ILUSTRAÇÃO DE INTERAÇÕES DESENVOLVEDOR-DESENVOLVEDOR..... 42
FIGURA 17 - VISUALIZAÇÃO DO PROJETO MYLYN DOCS EM 05/04/2017.. 48
FIGURA 18 - VISUALIZAÇÃO DAS 30 PRIMEIRAS TAREFAS DO PROJETO MYLYN DOCS. 49
FIGURA 19 - DETERMINAÇÃO DO CLUSTER OBJETIVO DE ACORDO COM A ANÁLISE VISUAL DAS 30 PRIMEIRAS TAREFAS DO PROJETO MYLYN DOCS..... 49
FIGURA 20 - DETERMINAÇÃO DO CLUSTER OBJETIVO DE ACORDO COM O ALGORITMO LNS (MONÇORES ET AL., 2018) SOBRE AS 30 PRIMEIRAS TAREFAS DO PROJETO MYLYN DOCS (AS 2 TAREFAS ISOLADAS NÃO APARECEM NA FIGURA). 51
FIGURA 21 – ILUSTRAÇÃO DA SIMILARIDADE TEXTUAL ENTRE A NOVA TAREFA E O CLUSTER DE TAREFAS SIMILARES POR INTERAÇÃO..... 52
FIGURA 22 – ILUSTRAÇÃO DO MÉTODO PARA RECOMENDAR TAREFAS PARA COMPOR O CONTEXTO DE UMA NOVA TAREFA EM DESENVOLVIMENTO DE SOFTWARE..... 53
FIGURA 23 – ILUSTRAÇÃO DO MÉTODO PARA RECOMENDAR ARTEFATOS PARA COMPOR O CONTEXTO DE UMA NOVA TAREFA EM PROJETO DE DESENVOLVIMENTO DE SOFTWARE. 55

FIGURA 24 – ILUSTRAÇÃO DOS MODELOS PARA RECOMENDAR DESENVOLVEDORES PARA COMPOR O CONTEXTO DE UMA NOVA TAREFA EM PROJETO DE DESENVOLVIMENTO DE SOFTWARE. 56
FIGURA 25 - REPRESENTAÇÃO DO TACIN EM BPMN. 57
FIGURA 26 - PERSPECTIVAS DE OBSERVAÇÃO DA COLETA DE DADOS. 61
FIGURA 27 – MODELO DE CLASSES DA IMPLEMENTAÇÃO DE REFERÊNCIA DOS MODELOS TACIN EXPRESSO EM UML (UNIFIED MODELING LANGUAGE). 139
FIGURA 28 - MODELO DE CLASSES DA IMPLEMENTAÇÃO DE REFERÊNCIA DO MÉTODO TACIN EXPRESSO EM UML. 140
FIGURA 29 -PROCESSO PARA CALCULAR A SIMILARIDADE TEXTUAL ENTRE UMA NOVA TAREFA E GRUPOS DE TAREFAS SIMILARES POR INTERAÇÃO EM ARTEFATOS. 141

Lista de Tabelas

TABELA 1 – ATIVIDADES REALIZADAS NA PESQUISA. 7
TABELA 2 - VETORES DE DOCUMENTOS GERADOS PELO RAPIDMINER UTILIZANDO OS OPERADORES TRANSFORM CASES (IMPLEMENTA CASE NORMALIZATION), TOKENIZE, FILTER STOPWORDS, STEM E FILTER TOKENS (SELECIONA TOKENS COM NÚMERO DE CARACTERES ENTRE 4 E 25). 25
TABELA 3 - SIMILARIDADE TEXTUAL POR COSSENO ENTRE OS DOCUMENTOS DA TABELA 2, UTILIZANDO BTO, BT, TF E TF-IDF CALCULADOS COM A FERRAMENTA RAPIDMINER. 26
TABELA 4 - OS VALORES DE Rd PARA Δt DE 0-7 E DE 38-41. 39
TABELA 5 - OS VALORES DE Rd PARA Δt DE 63-67 E DE 361-365. 40
TABELA 6 - OS VALORES DE Kc, EM GRUPOS DE ATÉ 10 TAREFAS, COM Ks FIXADO EM 1 PARA TODAS AS TAREFAS. 44
TABELA 7 – EXEMPLO DA INFLUÊNCIA DO FATOR hm NO CÁLCULO DE Kc. 44
TABELA 8 – DESCRIÇÃO CURTA E DESCRIÇÃO DA TAREFA 245759 DO PROJETO MYLYN DOCS. 52
TABELA 9- PASSOS PARA EXTRAIR COMMITS DE UM REPOSITÓRIO GIT. 63
TABELA 10 - RESULTADO DA PRIMEIRA RODADA REFERENTE À 01/05/2015. 67
TABELA 11 – ANÁLISE DE SIMILARIDADE ENTRE A CLASSIFICAÇÃO POR TAREFAS REALIZADAS (LINHA DE BASE) E AS CLASSIFICAÇÕES POR INTERAÇÃO, E UTILIZANDO OS MODELOS Kc E Kp. 68
TABELA 12 – ANÁLISE DE SENSIBILIDADE DE Kp EM 11/11/2008. 70
TABELA 13 - ANÁLISE DE SENSIBILIDADE DE Kp EM 11/11/2011 70
TABELA 14 – AVALIAÇÃO DA RECOMENDAÇÃO DE ARTEFATOS PARA 20 TAREFAS DE ACORDO COM A COBERTURA, REDUÇÃO E RC-MEASURE. O GRUPO DE TAREFAS SIMILARES POR INTERAÇÃO QUE POSSUI MAIOR SIMILARIDADE TEXTUAL COM A NOVA TAREFA É SELECIONADO, OS ARTEFATOS DAS TAREFAS DO GRUPO SÃO ENTÃO RECOMENDADOS. 77
TABELA 15 - AVALIAÇÃO DA RECOMENDAÇÃO DE ARTEFATOS PARA 20 TAREFAS DE ACORDO COM A COBERTURA, REDUÇÃO E RC-MEASURE. OS DOIS GRUPOS DE TAREFAS SIMILARES POR INTERAÇÃO QUE POSSUEM MAIS SIMILARIDADE TEXTUAL COM A NOVA TAREFA SÃO SELECIONADOS, OS ARTEFATOS DAS TAREFAS DESSES GRUPOS SÃO ENTÃO RECOMENDADOS. 78
TABELA 16 - AVALIAÇÃO DA RECOMENDAÇÃO DE ARTEFATOS PARA 20 TAREFAS DE ACORDO COM A COBERTURA, REDUÇÃO E RC-MEASURE. OS TRÊS GRUPOS DE TAREFAS SIMILARES POR INTERAÇÃO QUE POSSUEM MAIS SIMILARIDADE TEXTUAL COM A NOVA

TAREFA SÃO SELECIONADOS, OS ARTEFATOS DAS TAREFAS DESSES GRUPOS SÃO ENTÃO RECOMENDADOS. 80
TABELA 17 - RESULTADO DE 20 RODADAS PARA AVALIAÇÃO DA CLUSTERIZAÇÃO DE TAREFAS SEMELHANTES POR INTERAÇÃO E DA CORRELAÇÃO ENTRE SIMILARIDADES POR INTERAÇÃO E TEXTUAL. 82
TABELA 18 – COMPARATIVO ENTRE AS TRÊS OPÇÕES DE RECOMENDAÇÃO, O PRIMEIRO (TABELA 14), O PRIMEIRO E SEGUNDO (TABELA 15) OU OS TRÊS PRIMEIROS GRUPOS (TABELA 16) DE TAREFAS SIMILARES POR INTERAÇÃO QUE POSSUEM MAIS SIMILARIDADE TEXTUAL COM A NOVA TAREFA. 83
TABELA 19 – COBERTURA MÉDIA DE ASSOCIAÇÃO DA NOVA TAREFA AO CLUSTER COM MAIOR SIMILARIDADE TEXTUAL EM 20 RODADAS. OS TEXTOS DE DESCRIÇÃO CURTA E DESCRIÇÃO DAS TAREFAS FORAM AVALIADAS DE ACORDO COM OT, TF E TF-ID. 85
TABELA 20 – COMPARAÇÃO DA RECOMENDAÇÃO DE ARTEFATOS E DESENVOLVEDORES ENTRE O TACIN (RECOMENDAÇÃO DO PRIMEIRO GRUPO) E O KNN $(k = 1)$ SOBRE O PROJETO HOMEBREW/HOMEBREW-CASK-DRIVERS. 87
TABELA 21 - COMPARAÇÃO DA RECOMENDAÇÃO DE ARTEFATOS E DESENVOLVEDORES ENTRE O TACIN (RECOMENDAÇÃO DO PRIMEIRO GRUPO) E O KNN $(k = 1)$ SOBRE O PROJETO HOMEBREW/HOMEBREW-FORMULA-ANALYTICS..... 88
TABELA 22 - COMPARAÇÃO DA RECOMENDAÇÃO DE ARTEFATOS E DESENVOLVEDORES ENTRE O TACIN (RECOMENDAÇÃO DO PRIMEIRO GRUPO) E O KNN $(k = 1)$ SOBRE O PROJETO BREWSCI/HOMEBREW-SCIENCE. 89
TABELA 23 - COMPARAÇÃO DA RECOMENDAÇÃO DE ARTEFATOS E DESENVOLVEDORES ENTRE O TACIN (RECOMENDAÇÃO DO PRIMEIRO GRUPO) E O KNN $(k = 1)$ SOBRE O PROJETO KERL/KERL..... 90
TABELA 24- COMPARAÇÃO COM OS TRABALHOS RELACIONADOS. 100
TABELA 25 – EXTRAÇÃO DE DADOS DOS ARTIGOS SELECIONADOS. 130
TABELA 26 - EXTRAÇÃO DE DADOS DOS ARTIGOS SELECIONADOS, CONTINUAÇÃO DA TABELA 25..... 133

1 - Introdução

Na Introdução, o autor situa o leitor no cenário de desenvolvimento de software. Brevemente apresenta algumas ferramentas de contexto nas áreas de Percepção e de Recomendação no escopo da Engenharia de Software. Depois descreve o problema mitigado, em seguida apresenta os objetivos desta tese, suas questões de pesquisa, a metodologia e finaliza apresentando a organização do texto em capítulos.

O desenvolvimento de software é centrado em pessoas e na maioria das vezes é conduzido de forma colaborativa (MISTRÍK *et al.*, 2010). Os desenvolvedores, durante o turno de trabalho, geralmente executam um conjunto de tarefas, interagem com outros desenvolvedores e manipulam artefatos. As interações Desenvolvedor-Desenvolvedor ocorrem face-a-face ou através de ferramentas de comunicação quando os desenvolvedores necessitam esclarecer algumas questões sobre a realização das tarefas. As interações Desenvolvedor-Artefato acontecem quando os desenvolvedores manipulam artefatos através da leitura e/ou escrita. Portanto, os desenvolvedores criam ou editam artefatos e, quando necessário, solicitam ajuda a outros desenvolvedores com o objetivo de realizar tarefas.

Uma tarefa é a menor unidade de trabalho que pode ser delegada e deve ter o tamanho adequado para permitir planejamento e controle ("IEEE Standard for Software Project Management Plans", 1998). Quando os desenvolvedores recebem uma tarefa para corrigir um erro ou adicionar nova funcionalidade ao software, eles geralmente tentam montar um contexto de tarefa incluindo as tarefas similares à nova tarefa, artefatos que serão necessários alterar, e caso isso não seja suficiente, tentam descobrir quem melhor pode ajudar. A construção desse contexto é baseada na experiência do desenvolvedor e na capacidade humana de lembrar dos artefatos e tarefas anteriores (ROEDIGER, 2007). No entanto, a identificação do contexto de uma nova tarefa consome muito tempo, principalmente para desenvolvedores novatos (ERLIKH, 2000, MINELLI *et al.*, 2015). Isso se deve, principalmente, ao fato de que uma nova tarefa tem, geralmente, uma indicação do "o que" deve ser realizado, mas não a indicação "do como" deve ser realizado contendo os insumos (artefatos, documentos, etc) necessários para a execução da tarefa.

Projetos de desenvolvimento de software podem durar meses ou anos, onde os desenvolvedores executam várias tarefas, interagem entre si e manipulam muitos artefatos. O acúmulo destas interações, artefatos e tarefas dá origem a um histórico de

elementos do projeto e é com base neste histórico que os desenvolvedores identificam intuitivamente o contexto de novas tarefas (MINELLI *e al.*, 2015). No entanto, os desenvolvedores não têm conhecimento do histórico de elementos do projeto ou o esquecem devido ao grande volume de informações e às limitações naturais da cognição humana que determinam nossas capacidades para lidar com o excesso de informação (ROEDIGER, 2007, THALHEIMER, 2010).

Na literatura, existem algumas definições de contexto. De forma genérica, contexto limita o escopo de solução para o problema sem intervir nele explicitamente (BRÉZILLON, 1999). No desenvolvimento de software, contexto mínimo de tarefa é formado pelo conjunto de artefatos que o desenvolvedor utiliza para realizar a tarefa (KERSTEN, 2007, OMORONYIA *et al.*, 2010). Outra variável de contexto muito explorada é a identificação de especialistas, quem sabe sobre partes do software (MORAES *et al.*, 2010, ZHU *et al.*, 2016).

Contexto também é um assunto tratado na área de Percepção (GUTWIN *et al.*, 1996) e também na de Recomendação (ROBILLARD *et al.*, 2014) ambas direcionadas para a Engenharia de Software. As ferramentas de Percepção direcionadas para a Engenharia de Software têm como objetivo aumentar o conhecimento do desenvolvedor sobre o ambiente de trabalho. Por exemplo, o Mylyn[1] faz uso das interações dos desenvolvedores sobre os artefatos em tempo real e modela o interesse do desenvolvedor pelos artefatos em um ambiente de desenvolvimento de software (KERSTEN, 2007). O Mylyn baseia-se no modelo de Grau de Interesse (sigla DOI em Inglês, *Degree Of Interest*) do desenvolvedor sobre os artefatos com base nos eventos de visualização, edição, seleção, comando, propagação e previsão no decorrer da tarefa. Já o modelo CRI de OMORONYIA (2008) atua em um ambiente colaborativo para produzir medidas de influências entre desenvolvedores, artefatos e tarefas a partir das interações de visualização, alteração e criação de artefatos realizadas pelos desenvolvedores. O CRI também permite visualizar a evolução das influências ao longo do tempo.

Uma forma usual de se utilizar o histórico de projetos na Engenharia de Software é através de Sistemas de Recomendação. As ferramentas de Recomendação na Engenharia de Software têm como objetivo ajudar os usuários a reduzir as incertezas do futuro através da análise do histórico do projeto, e de forma mais específica, segundo

[1] http://www.eclipse.org/mylyn/, visitado em 11/09/2018.

ROBILLARD *et al. (2014)*, trata-se de "um aplicativo de software que fornece itens de informações estimados como valiosos para uma tarefa de engenharia de software em um determinado contexto". Algumas propostas focam em conhecer profundamente os desenvolvedores para melhor descobrir os seus semelhantes, ou seja, possíveis colaboradores (CANFORA *et al.*, 2012). Outras propostas buscam relacionamentos entre desenvolvedores, tarefas e artefatos (ASHOK *et al.*, 2009). As propostas de recomendação tendem a utilizar as técnicas das áreas de Mineração de Dados (NAJAFABADI *et al.*, 2017), Aprendizado de Máquina (MYRÉN, 2017) e Processamento de Linguagem Natural (DAHL, 2013) para produzir recomendações adequadas.

Especificamente sobre trabalhos que recomendam contexto para nova tarefa, alguns trabalhos recomendam artefatos, isto é, código fonte, arquivos, classes, pacotes (ALMHANA *et al.*, 2016, ANDRIC *et al.*, 2004, ANTUNES *et al.*, 2012, CAICEDO e DUARTE, 2015, ČUBRANIĆ *et al.*, 2005b, YE *et al.*, 2014), enquanto outros recomendam especialistas em artefatos (FRITZ *et al.*, 2014, MORAES *et al.*, 2010). As propostas de XIA *et al.* (2013), XIE *et al.* (2012) e ZHU *et al.* (2016) recomendam especialistas em novas tarefas. ASHOK *et al.* (2009) e MALHEIROS *et al.* (2012) recomendam tarefas similares à nova tarefa, enquanto ASHOK *et al.* (2009) e WANG *et al.* (2017) também recomendam especialistas nas tarefas similares à nova tarefa. Na direção de recomendar contexto para uma nova tarefa de desenvolvimento de software, não foi identificado nenhum trabalho que recomende artefatos, tarefas similares à nova tarefa, especialistas em artefatos e especialistas em tarefas similares à nova tarefa, levando em consideração explicitamente o esquecimento humano como modelado explicitamente por diversos autores (MURRE e CHESSA, 2011, MURRE *et al.*, 2013; MURRE e DROS, 2015, RUBIN e WENZEL, 1996, RUBIN *et al.*, 1999, WIXTED e EBBESEN, 1991), como faz o método de recomendação apresentado nesta tese.

O objetivo central desta tese é recomendar um contexto para uma nova tarefa, no seu início, o mais próximo possível do contexto registrado após a sua realização. A Figura 1 ilustra o cenário de construção do contexto para uma nova tarefa. Nesse cenário, inicialmente, uma nova tarefa é descrita de forma textual. Com base nessa descrição e consultando o histórico de elementos do projeto, o desenvolvedor tenta construir um contexto inicial para realizar a nova tarefa. Essa construção o desenvolvedor faz, às vezes, utilizando as ferramentas de busca textual, em um ciclo de descoberta e lembrança sobre os elementos do projeto. No histórico de elementos do projeto, o contexto de tarefa está

parcialmente explícito, podendo existir correlações ainda não descobertas, mas é possível ter os artefatos que foram criados e alterados enquanto os desenvolvedores realizavam tarefas. Da mesma forma, após o término de uma nova tarefa, é possível identificar quem participou e quais artefatos foram criados e alterados.

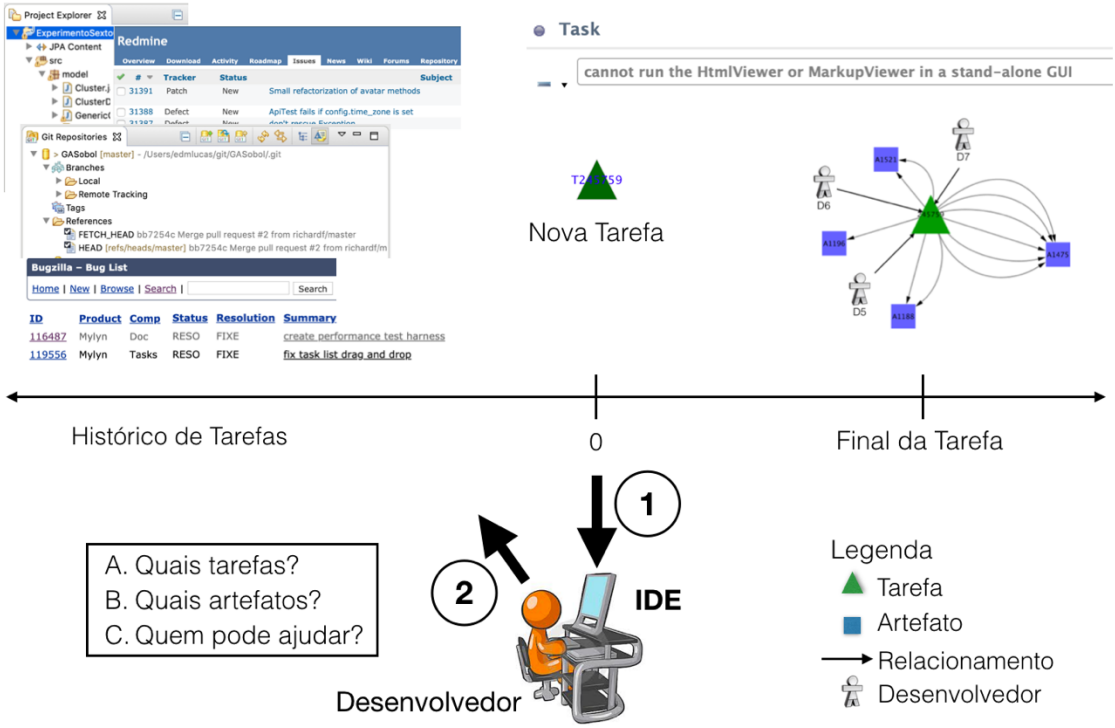

Figura 1 - Visão geral do problema investigado.

A solução proposta nesta tese para o problema apresentado faz uso das técnicas de Processamento de Linguagem Natural (PLN) e Clusterização considerando a modelagem do esquecimento humano. O PLN é formado por um conjunto de técnicas computacionais motivadas por teoria para a análise e representação automáticas das linguagens utilizadas por humanos (CAMBRIA e WHITE, 2014). A Clusterização é uma técnica computacional para organizar objetos em grupos. Objetos mais semelhantes tendem a ficar no mesmo grupo, enquanto objetos menos semelhantes, ou não semelhantes, tendem a ficar em grupos diferentes (HAN *et al.*, 2012, JAIN *et al.*, 1999). O esquecimento humano pode ser modelado por funções matemáticas com o objetivo de prever como a retenção de memória diminui ao longo do tempo (MURR e CHESSA, 2011, RUBIN e WENZEL, 1996).

De acordo com BJØRNER (2003), um método pode ser entendido como um conjunto de princípios para selecionar e aplicar um conjunto de técnicas e ferramentas para a construção de artefatos. Por isso, a solução proposta nesta tese, denominada

TACIN (*Task Context based on Interactions*, Contexto de Tarefa baseado em Interações), é apresentada como um método que usa as informações de um modelo de interação em artefatos e combina as técnicas de PLN com Clusterização, considerando a modelagem do esquecimento humano, para recomendar contexto para novas tarefas em projetos de desenvolvimento de software.

1.1 Objetivos

Para esta tese, o histórico de elementos do projeto é definido como o conjunto que contém três elementos (desenvolvedores, tarefas e artefatos do projeto de software), onde uma tarefa está relacionada com um desenvolvedor por associação, uma tarefa está relacionada com um artefato pela interação de edição do desenvolvedor sobre o artefato e um desenvolvedor está relacionado com outro desenvolvedor por interações de colaboração (por exemplo, via troca de mensagens textuais). A Figura 2 mostra essas relações entre tarefas, desenvolvedores e artefatos.

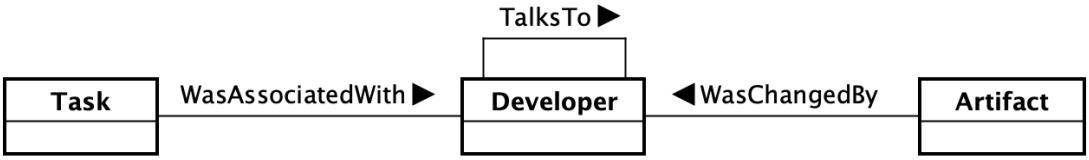

Figura 2 - Histórico de Elementos do Projeto.

Assim, no escopo desta tese, contexto de tarefa é definido como um conjunto de elementos contendo: 1- artefatos editados na realização da tarefa, assim como em ČUBRANIĆ *et al.* (2005a); 2- especialistas na tarefa, como visto em MCDONALD e ACKERMAN (2000); 3- tarefas similares por interação de edição em artefatos e; 4- especialistas nas tarefas similares (WANG *et al.*, 2017). A Figura 3 mostra que as interações de edição dos desenvolvedores sobre os artefatos criam associações entre os desenvolvedores, tarefas e artefatos. Os especialistas são os desenvolvedores que mais expressaram conhecimento ao participar das conversas sobre a realização das tarefas e/ou na produção de interações de edição nos artefatos.

Com base na definição de contexto de tarefa, o contexto de nova tarefa é composto por: 1- artefatos candidatos a serem editados pelos desenvolvedores na realização da nova tarefa; 2- desenvolvedores que têm potencial em apresentar conhecimento na nova tarefa; 3- tarefas realizadas similares à nova tarefa e; 4- por especialistas nas tarefas similares à nova tarefa.

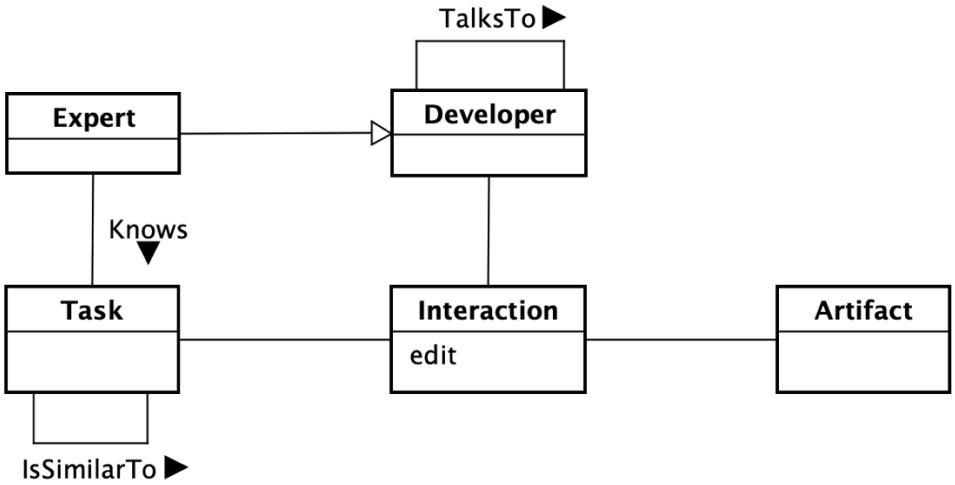

Figura 3 - Contexto de Tarefa.

Com base nessas definições, o objetivo desta tese é: recomendar contexto para uma nova tarefa em projeto de desenvolvimento de software. Dado que a identificação de especialistas na nova tarefa é baseada na identificação de especialistas no histórico de elementos do projeto e, considerando que o termo medição é definido como uma redução quantitativa da incerteza baseando-se em uma ou mais observações (HUBBARD, 2007), então também é definido o objetivo específico: medir o conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software.

1.2 Questões de Pesquisa

A questão de pesquisa desta tese está alinhada com o seu objetivo de recomendar contexto para novas tarefas de desenvolvimento de software:

> **QP**- A organização do histórico de elementos do projeto, em função da similaridade por interação dos desenvolvedores sobre artefatos, combinada com a similaridade textual, é suficiente para produzir recomendação de contexto de nova tarefa em projeto de desenvolvimento de software?

A QP tem uma subquestão de pesquisa, QP_1, sendo que o resultado da QP_1 influencia no resultado da QP. O objetivo específico desta tese baseia-se na ideia de utilizar os dados das interações dos desenvolvedores para medir conhecimento. Para isso, define-se a subquestão de pesquisa:

QP_1- Os dados do histórico das interações, considerando a modelagem do esquecimento humano, são suficientes para produzir medições de conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software?

1.3 Metodologia

A Tabela 1 apresenta as atividades realizadas nesta tese conforme a metodologia de PEFFERS *et al.* (2007) para conduzir pesquisa científica. Inicialmente, na Atividade I, *identificar o problema e motivação*, foi feito um mapeamento de trabalhos sobre contexto na Engenharia de Software. O mapeamento foi realizado utilizando *strings* de busca nas bases científicas (ACM[2], IEEE[3] e Scopus[4]) e com a técnica de Snowballing (JALALI e WOHLIN, 2012). O resultado identificou propostas que fazem uso de monitoramento das interações dos desenvolvedores (FRITZ *et al.*, 2010, KERSTEN, 2007, OMORONYIA, 2008), e recomendação de contexto na Engenharia de Software (ČUBRANIĆ *et al.*, 2005a, THOMPSON e MURPHY, 2014). A revisão também considerou a técnica de *Snowballing* (JALALI e WOHLIN, 2012), isto é, considerou as referências e as citações dos artigos visitados. O resultado confirmou que a recomendação de contexto ainda era um problema não bem resolvido na Engenharia de Software (THOMPSON e MURPHY, 2014). Além disso, a tentativa de perceber o conhecimento dos desenvolvedores sobre artefatos não considerava explicitamente o esquecimento humano (ROEDIGER, 2007) na modelagem do problema (FRITZ *et al.*, 2014).

Tabela 1 – Atividades realizadas na pesquisa.

I	*Identificar o problema e motivação*
Como	Mapeamento da literatura utilizando *strings* de busca nas bases científicas (ACM, IEEE e Scopus) e com a técnica de *Snowballing* (JALALI e WOHLIN, 2012).
Resultado	Recomendação de contexto ainda era um problema não bem resolvido na Engenharia de Software (THOMPSON e MURPHY, 2014). Medida de conhecimento dos desenvolvedores sobre artefatos não considerava

[2] https://dl.acm.org

[3] https://ieeexplore.ieee.org

[4] https://www.scopus.com/

	explicitamente o esquecimento humano (ROEDIGER, 2007) na modelagem do problema (FRITZ *et al.*, 2014).
II	*Definir os objetivos para uma solução*
Como	Observando os resultados do mapeamento da literatura.
Resultado	Primeiro objetivo: medir o conhecimento do desenvolvedor sobre artefato, tarefa, tarefas similares e todo o projeto de software. O segundo objetivo: recomendar contexto para uma nova tarefa em projeto de desenvolvimento de software. O segundo objetivo faz uso da solução do primeiro objetivo para recomendar desenvolvedores para compor o contexto da nova tarefa.
III	*Projeto e Desenvolvimento*
Como	Coleta de dados de um projeto de software livre[5] para guiar a construção e evolução de modelos para medir conhecimento e de um método para recomendar contexto de nova tarefa. Criar e calibrar os modelos considerando os estudos na área de modelagem de esquecimento e retenção de conhecimento (BARBOSA *et al.*, 2015, CHESSA e MURRE, 2009, LEE, 2004, KRÜGER *et al.*, 2018). Criar um método para recomendar contexto para uma nova tarefa de desenvolvimento de software. Ferramentas utilizadas[6]: RapidMiner Studio 9.0; Cytoscape 3.6.0; Linguagem Java 1.8.0_25; PostgreSQL 9.6; SQL (Structured Query Language); pgAdmin e RStudio Desktop. Parte desta pesquisa foi desenvolvida com o apoio do Núcleo Avançado de Computação de Alto Desempenho (NACAD) da COPPE, Universidade Federal do Rio de Janeiro (UFRJ).
Resultado	Modelos de percepção do conhecimento do desenvolvedor sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software. Um método para recomendar contexto para uma nova

[5] Dados do projeto http://www.eclipse.org/mylyn/ e do seu subprojeto http://projects.eclipse.org/projects/mylyn.docs

[6] RapidMiner Studio (https://www.rstudio.com); Cytoscape (http://cytoscape.org/); Linguagem Java (http://www.oracle.com/technetwork/java/javase/downloads/);
PostgreSQL (http://www.postgresql.org);
SQL (https://www.iso.org/standard/63555.html); pgAdmin (https://www.pgadmin.org/) e RStudio Desktop (https://www.rstudio.com).

	tarefa de desenvolvimento de software combinando similaridade de tarefas por interação e textual.
IV	*Demonstração*
Como	Inicialmente, o agrupamento de tarefas foi realizado de acordo com a similaridade textual entre as tarefas. Depois, foi gerada uma lista com os 10 desenvolvedores que mais apresentaram conhecimento sobre o Projeto.
Resultado	O agrupamento de tarefas com diferentes números de grupos não mostrou uma interferência significativa na percepção do conhecimento dos desenvolvedores sobre o projeto Mylyn. O desenvolvedor que mais apresentou conhecimento concordou com a posição dos cincos primeiros, mas discordou com a grande diferença de pontuação entre eles.
V	*Avaliação*
Como	Avaliação do modelo de percepção do conhecimento do desenvolvedor sobre todo o projeto de desenvolvimento de software. Avaliação do método para recomendar contexto para uma nova tarefa de desenvolvimento de software combinando similaridade de tarefas por interação e textual. Análise comparativa com os trabalhos relacionados.
Resultado	O modelo de percepção do conhecimento do desenvolvedor sobre projeto de software apresentou uma similaridade média de 72% com a linha de base tarefas realizadas no projeto. E a avaliação de recomendação de artefatos para compor o contexto da nova tarefa apresentou uma Cobertura média de 52%. O Tacin produziu resultados superiores quando foi comparado com um algoritmo base de recomendação. Não foi identificado nenhum trabalho relacionado que meça a similaridade entre as tarefas utilizando o histórico das interações realizadas pelos desenvolvedores. Os trabalhos também não consideraram explicitamente os modelos das características humanas de esquecimento nas recomendações.
VI	*Comunicação*
Como	Colaboração em ambientes de reutilização de software orientados a processos (LUCAS *et al.*, 2013, 2014, 2017). Artigo apresentado no VI Workshop de Teses e Dissertações da CBSoft (LUCAS e OLIVEIRA, 2016). E depois, na versão ampliada, submetida para o Journal of Systems and Software, com foco no Capítulo 3 desta tese. E sobre o

	Capítulo 4 foi publicado o artigo na conferência SEKE (LUCAS *et al.*, 2019).
Resultado	Artigos aceitos e avaliação positiva para a continuidade da pesquisa.

Depois, para a Atividade II, *definir os objetivos para uma solução*, observando os resultados do mapeamento da literatura, foram definidos dois objetivos. O primeiro objetivo visa medir o conhecimento do desenvolvedor sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software, utilizando as interações dos desenvolvedores, e considerando a modelagem do esquecimento humano (CHESSA e MURRE, 2009, LEE, 2004). O segundo objetivo recomenda contexto de nova tarefa em projeto de desenvolvimento de software utilizando os resultados do primeiro objetivo.

O *Projeto e Desenvolvimento*, Atividade III da Tabela 1, fez uso de dados de um projeto de desenvolvimento de software livre[7] para guiar a construção e evolução de modelos para medir o conhecimento e do método para recomendar contexto de nova tarefa, por dois motivos: primeiro, o projeto coleta as interações dos desenvolvedores sobre os artefatos em tempo real; segundo, há o *ranking* de quem mais realiza tarefas no projeto, escolhido como a linha de base para calibrar os modelos, porque admitimos que o desenvolvedor que mais realiza tarefas é também o que mais conhecimento tem sobre o projeto.

A calibração dos modelos, inicialmente, considerou que o desenvolvedor retém 65% do conhecimento de um projeto de software após 10 anos de inatividade, inspirado no estudo de BARBOSA *et al.* sobre áreas de conhecimento (BARBOSA *et al.*, 2015). Depois, de acordo com o estudo de KRÜGER *et al.* (2018), levou-se em consideração que os desenvolvedores esquecem 50% de um arquivo de código fonte em 40 dias. Além disso, utilizou a informação de que os desenvolvedores tendem a esquecer os artefatos que não foram manipulados por eles recentemente (FRITZ *et al.*, 2007, 2010, MAALEJ, 2010).

Na Atividade IV, *Demonstração*, em uma análise inicial, os dados do projeto Mylyn foram coletados utilizando a consulta do Bugzilla do Eclipse[8] para selecionar todas as tarefas resolvidas que continham as informações das interações dos desenvolvedores

[7] Dados do projeto http://www.eclipse.org/mylyn/ e do seu subprojeto http://projects.eclipse.org/projects/mylyn.docs

[8] https://bugs.eclipse.org/bugs/query.cgi

sobre os artefatos (LUCAS e OLIVEIRA, 2016). A análise dos dados mostrou que a seleção ainda continha a informação correta de quem mais realizou tarefas no projeto. O agrupamento de tarefas foi realizado de acordo com a similaridade textual entre as 1804 tarefas selecionadas, considerando o campo *short_description* das tarefas. O agrupamento de tarefas com diferentes números de grupos (40, 60, 80 e 100) não mostrou uma interferência significativa na percepção do conhecimento dos desenvolvedores sobre o projeto Mylyn. Depois, foi gerada uma lista com os 10 desenvolvedores que mais apresentaram conhecimento sobre o Projeto. Para o primeiro da lista, foi perguntado por e-mail se ele concordava com a classificação. A resposta dada foi que para os dois primeiros nomes da lista, a pontuação era muito grande: ele esperava valores mais semelhantes para os cincos primeiros colocados.

Na *Avaliação*, Atividade V, foi realizado o agrupamento de tarefas considerando a similaridade de tarefas por interação de edição em artefatos e uma análise comparativa com os trabalhos relacionados. Em resumo, o modelo de conhecimento sobre projeto de desenvolvimento de software apresentou uma similaridade média de 72% com a linha de base tarefas realizadas no projeto. A avaliação do método para recomendar contexto para uma nova tarefa apresentou uma porcentagem de acerto de 52% (Cobertura média), no caso de recomendação de artefatos. Quando o método proposto foi comparado com um algoritmo base de recomendação, o resultado mostrou que o Tacin produz uma porcentagem média de acerto superior na recomendação de artefatos e desenvolvedores. Uma revisão sistemática da literatura foi realizada em julho de 2018 para atualizar os trabalhos relacionados desta tese (Apêndice A). O objetivo foi identificar trabalhos que têm como objetivo recomendar artefatos, tarefas e/ou desenvolvedores em projetos de desenvolvimento de software. O resultado selecionou 16 trabalhos relacionados e, dentre eles, não se identificou nenhum trabalho relacionado que medisse a similaridade entre as tarefas utilizando o histórico das interações realizadas pelos desenvolvedores. Além disso, os trabalhos não consideraram explicitamente os modelos das características humanas de esquecimento nas recomendações.

E, finalmente, a *Comunicação* da pesquisa, Atividade VI. Em trabalhos anteriores, a pesquisa explorou o desafio de como explicitar colaboração em ambientes de reutilização de software orientados a processos (LUCAS *et al.*, 2013, 2014, 2017). Naquela oportunidade, coletamos evidências de que é preciso investir esforços para dar suporte de percepção para os desenvolvedores em ambientes de reutilização de software. Nesta tese, a pesquisa evoluiu para a criação de modelos de percepção do conhecimento

do desenvolvedor em projetos de desenvolvimento de software, pesquisa devidamente comunicada à comunidade científica através das publicações (LUCAS e OLIVEIRA, 2016) e, na versão ampliada, submetida para o Journal of Systems and Software. Depois, sobre o Capítulo 4, um artigo foi publicado na conferência SEKE (LUCAS *et al.*, 2019).

1.4 Organização

A dissertação da presente tese segue organizada em capítulos. O Capítulo 2 mostra a definição de Projeto, Projeto de Software, Processo e Processo de Software. O Capítulo 2 também mostra como construir e avaliar sistemas de recomendação na Engenharia de Software, apresenta a definição de clusterização, mostra a técnica de Processamento de Linguagem Natural e finaliza apresentando os conceitos de Percepção e Conhecimento.

O Capítulo 3 define as interações Desenvolvedor-Artefato e Desenvolvedor-Desenvolvedor. Logo após, formaliza matematicamente os quatro modelos do Tacin para medir o conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software, levando em consideração o esquecimento dos desenvolvedores.

O Capítulo 4 faz uma análise visual dos dados de um projeto de desenvolvimento de software para mostrar o raciocínio utilizado na construção do método Tacin. Apresenta, em detalhe, o método para recomendar tarefas, artefatos e desenvolvedores para compor o contexto inicial de uma nova tarefa. E finaliza com uma representação do método Tacin.

O Capítulo 5 descreve como foi realizada a coleta de dados para avaliar o Tacin. Segue com a avaliação dos modelos de percepção do conhecimento e da avaliação do método para recomendar contexto de tarefa. Cada avaliação descreve os materiais e métodos utilizados, apresenta a análise dos resultados e descreve as ameaças à validade.

O Capítulo 6 mostra como os trabalhos relacionados foram selecionados, descreve-os ressaltando o objetivo da recomendação, como recomendam e informações sobre a avaliação. Compara os trabalhos de acordo com as características de utilização das técnicas de clusterização e Processamento de Linguagem Natural, informa se modelam similaridade por interação, esquecimento humano e reaprendizagem, se o trabalho mede conhecimento sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software, e se recomendam artefatos, tarefas similares, e também especialistas em artefato, tarefa e tarefas similares.

O Capítulo 7 faz uma discussão sobre os modelos para perceber conhecimento e sobre o método para recomendar contexto de nova tarefa, além de responder à subquestão de pesquisa e à questão de pesquisa desta tese.

O Capítulo 8 lista alguns trabalhos futuros e mostra como o Tacin também pode ser aplicado em outros domínios. Finalmente, conclui que o Tacin atende ao objetivo desta tese.

2 - Fundamentação Teórica

O capítulo inicia definindo Projeto, Projeto de Software, Processo e Processo de Software na Engenharia de Software. Segue apresentando como construir e avaliar Sistemas de Recomendação na Engenharia de Software. O capítulo também mostra a definição de clusterização e apresenta o algoritmo de clusterização utilizado nesta tese e, logo após, apresenta a técnica de Processamento de Linguagem Natural. Este capítulo finaliza apresentando os conceitos de Percepção e Conhecimento.

Na área de Interação Humano-Computador, especialmente quando aplicada à Engenharia de Software, há uma tendência de capturar as interações do desenvolvedor com o objetivo de ampliar a percepção do ambiente em projetos de desenvolvimento de software (KERSTEN, 2007, MAALEJ *et al.*, 2017, OMORONYIA, 2008). Por exemplo, o interesse de um desenvolvedor em um artefato pode variar com o tempo durante uma tarefa (KERSTEN, 2007; MAALEJ *et al.*, 2017). Da mesma forma, a influência de um desenvolvedor em um artefato também pode mudar com o tempo (OMORONYIA, 2008).

As interações são de diversos tipos e estão espalhadas ao longo do projeto de desenvolvimento de software. Uma visão importante, que está relacionada ao conceito de interação Desenvolvedor-Artefato, é a relação produtor-consumidor entre atividades, onde uma atividade consumidora consome produtos de uma atividade produtora (MALONE e CROWSTON, 1994).

A produção de artefatos pode ser registrada quando há ação de edição sobre eles, sendo que a produção ou alteração de um artefato pode fazer uso de artefatos que não serão necessariamente editados na execução da tarefa. Por exemplo, durante a execução de um projeto de desenvolvimento de software, que compreende vários desenvolvedores e artefatos, é possível que o conhecimento de um desenvolvedor sobre um artefato possa ser registrado em outro quando se refere ao primeiro, geralmente através da declaração de variáveis e uso de métodos ou rotinas combinadas com a importação de bibliotecas.

Além disso, consideramos que a ação de visualização de um artefato só faz sentido na produção de software, quando essa ação resulta em ações de edição posteriores, da mesma forma que os efeitos da ação de edição são usados para medir produção e produtividade (ARMOUR, 2004). Por essas razões, apenas as ações de edição são utilizadas nesta tese para representar as interações Desenvolvedor-Artefato. Essas ações podem ser capturadas por meio de ferramentas de monitoramento como o Mylyn

(KERSTEN, 2007) ou por meio da análise de *commits* das ferramentas de repositório de código (por exemplo, Git e SVN).

A seguir são apresentados os conceitos de projetos e sistemas de recomendação na Engenharia de Software, processamento de linguagem natural, clusterização, percepção e conhecimento.

2.1 Projetos na Engenharia de Software

Projeto é definido como um empreendimento temporário realizado para criar um único produto, serviço ou resultado[9]. Por ser um empreendimento temporário, conclui-se que um projeto tem início, meio e fim, além de envolver o trabalho de pessoas. No contexto de um projeto de software, o trabalho pode ser classificado como uma tarefa ou atividade. A tarefa é a menor unidade de trabalho que pode ser delegada e deve ter o tamanho estimado que permita planejamento e controle. Já uma atividade é o agrupamento de tarefas que pode fazer parte do processo de gerenciamento de projeto. Nessa direção, Projeto de Software é definido como um conjunto de atividades, tanto técnicas como de gerenciamento, necessárias para satisfazer os termos e condições de um contrato de projeto ("IEEE Standard for Software Project Management Plans", 1998).

Um processo é um conjunto de atividades parcialmente ordenadas para atingir um objetivo (BPMN, 2018, FEILER e HUMPHREY, 1993, SPEM, 2018). Geralmente, o Processo de Software pode ser definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software (FUGGETTA, 2000). O processo precisa de uma linguagem para ser descrito e uma tendência atual é utilizar uma linguagem de alto nível para fins de automação. Nessa direção, o Modelo e Notação de Processos de Negócio (BPMN) tem sido utilizado porque é um padrão OMG que oferece várias implementações (ACTIVITI, 2018, BPMN, 2018, OMG, 2018). Um exemplo de reuso de uma implementação do BPMN é descrito em (LUCAS *et al.*, 2017).

Na Engenharia de Software, o uso de processos é defendido em diferentes níveis para melhorar a qualidade do software produzido (FUGGETTA, 2000). Segundo ROCHA *et al.* (2001), os processos relacionados ao planejamento são organizados no Processo de Software da Organização, Processos de Software Especializados e Processo do Projeto de Software. O nível de modelo de processo mais próximo do entendimento

[9] https://www.pmi.org/pmbok-guide-standards/lexicon visitado em 11/07/2018

humano é o Modelo de Processo de Referência, por exemplo, o Processo Unificado (ISO/IEC/IEEE 12207:2017(E), 2017, LARMAN, 2004) e o MPS.Br - Modelo de Referência para Melhoria de Processos de Software Brasileiro (MPS.BR, 2012). O Processo de Software Padrão é definido a partir de Modelos de Processos de Referência dentro de uma organização de acordo com seus padrões de qualidade e é especializado de acordo com as necessidades específicas da organização, grupos de trabalho e projetos (ROCHA *et al.*, 2001). O Processo de Software Especializado é adaptado para atender às características específicas de um projeto, como modelos de ciclo de vida, características do projeto, características da equipe, disponibilidade de recursos, requisitos de qualidade do produto, entre outros, denominados como Processo de Projeto de Software. Um exemplo de adaptação de processo a um projeto de software pode ser visto em (PILLAT *et al.,* 2015).

O Processo de Software Instanciado é o processo mais próximo da automação, representando uma instância (execução) do Processo de Projeto de Software, onde algumas tarefas são executadas automaticamente e outras com a necessária intervenção das pessoas. Como o processo de desenvolvimento de software é intensivo em conhecimento (DI CICCIO *et al.*, 2015, QUMER *et al.*, 2008), muitas tarefas executadas nesse nível não puderam ser planejadas nos níveis anteriores. O Processo de Software Instanciado representa as tarefas executadas que são registradas no histórico de execução, que foram ou não previstas nas atividades do processo. Nessa direção, técnicas de Mineração de Processo são usadas para esclarecer as diferenças entre o processo planejado e o executado visando sua melhoria (GOMES *et al.*, 2014, SANTOS *et al.*, 2015). Portanto, o Processo de Software Instanciado precisa do suporte de ferramentas computacionais para auxiliar os desenvolvedores na execução das tarefas.

O processo de desenvolvimento de software é intensivo em conhecimento também porque é essencialmente uma atividade humana com suporte de ferramentas (DI CICCIO *et al.*, 2015, OMORONYIA, 2008). Assim, é possível capturar interações para extrair conhecimento sobre artefatos e desenvolvedores. Essa é uma nova tendência das novas ferramentas de suporte para o desenvolvimento de software, onde o monitoramento de interações é uma fonte de descobertas (HATTORI e LANZA, 2010, KERSTEN, 2007, ROBBES e LANZA, 2008).

Por outro lado, proveniência são informações sobre entidades, atividades e pessoas envolvidas na produção de um dado ou coisa, que podem ser usadas para formar avaliações sobre sua qualidade, credibilidade ou confiabilidade (BELHAJJAME *et al.*,

2013). Portanto, as informações sobre a produção de software podem ser organizadas através de um modelo de proveniência.

A Figura 4 mostra o modelo conceitual do núcleo da especificação PROV-DM da W3C (BELHAJJAME *et al.*, 2013). O modelo é composto por 3 tipos: 1- uma entidade é um tipo físico, digital, conceitual ou outro tipo de coisa com alguns aspectos fixos. Entidades podem ser reais ou imaginárias; 2- uma atividade ocorre em um período de tempo para consumir, processar, transformar, modificar, realocar, usar ou gerar entidades; e 3 - um agente é algo que tem alguma forma de responsabilidade, uma pessoa, um software em execução, uma empresa etc.

O modelo PROV-DM especifica sete relações que atuam sobre os tipos Entidade, Atividade e Agente. Uma entidade é gerada através de uma atividade (WasGeneraredBy), podendo ser derivada de uma versão anterior (WasDerivedFrom). Uma atividade está associada a um agente por responsabilidade (WasAssociatedWith) e também pode estar relacionada com uma entidade por utilização (Used). Atividades podem ter relações de dependências, por exemplo, uma atividade B precisa esperar o término de uma atividade A, caso B precise usar uma entidade produzida por A (WasInformedBy). Em casos onde as atividades não são conhecidas ou são irrelevantes, uma entidade é atribuída ao agente diretamente (WasAttributedTo). Por último, um agente pode ser responsável pela atividade de outro agente (ActedOnBehalOf).

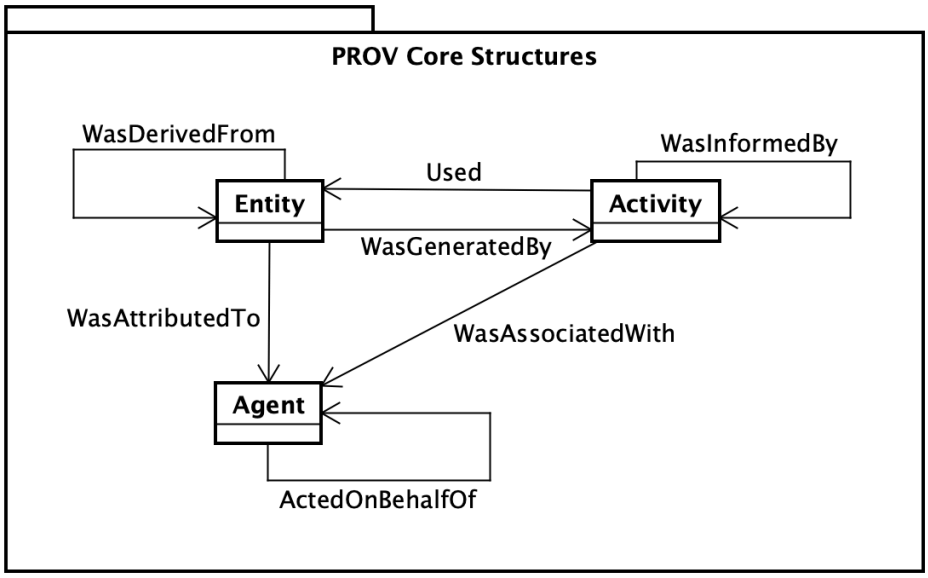

Figura 4 - Modelo conceitual do núcleo da especificação PROV-DM da W3C.

2.2 Sistemas de Recomendação para a Engenharia de Software

O desenvolvimento de software é um processo complexo. O conhecimento dos desenvolvedores, ou a falta dele, sobre elementos de software, como artefatos e tarefas, impacta na tomada de decisões, pois esse conhecimento influencia nas escolhas feitas ao longo do processo de software. O processo de desenvolvimento de software também pode ser visto como um processo emergente porque muitas ações surgem durante o processo como resultado da disponibilidade de informações adicionais, uma situação mitigada em processos ágeis de desenvolvimento de software (ABRANTES e TRAVASSOS, 2011). Nesse cenário, as aplicações para apoiar os desenvolvedores na busca por informações relevantes para a realização de tarefas em um dado contexto define a área Sistemas de Recomendação para a Engenharia de Software, com a sigla SRES (ROBILLARD e WALKER, 2014).

Segundo ROBILLARD *et al.* (2014), não há uma arquitetura de referência para SRESs porque os sistemas são construídos fortemente acoplados às fontes de dados de entrada. Dessa forma, eles indicam considerar o pré-processamento de dados, a captura do contexto, produção de recomendações e apresentação das recomendações para a construção de SRESs.

De acordo com PROKSCH *et al.* (2015), a construção de um SRES deve contemplar as seguintes atividades: enquadrar o problema; determinar as entradas; construir o recomendador, entregar as recomendações e avaliar a utilidade do recomendador. No enquadramento do problema é preciso descrever a tarefa, o contexto e os atores alvo. Nesse caso, a tarefa visa determinar um objetivo, por exemplo, ajudar na busca por artefatos que possivelmente serão editados ao adicionar uma nova funcionalidade ou correção de erro no software. O contexto descreve os artefatos e ferramentas de trabalho. E, finalmente, os atores alvos representam os beneficiários da recomendação. Como exemplo, para a construção do sistema de recomendação da presente tese, temos:

- *Enquadrar o problema*: O desenvolvedor ao receber uma tarefa para corrigir um erro ou adicionar uma nova funcionalidade ao software, inicia, normalmente, com uma busca por artefatos que precisará alterar para realizar a tarefa. Essa busca pode se basear na memória humana, lembrança dos artefatos já utilizados em tarefas anteriores, por isso limitado à experiência do desenvolvedor e na limitação humana sobre a capacidade de lembrar. Além disso, a busca consome tempo, mesmo quando se faz uso das ferramentas tradicionais de navegação e busca textual disponíveis nos ambientes de desenvolvimento

(por exemplo, o *Project Explorer* do Eclipse). Após a descoberta dos artefatos, caso surja uma dúvida sobre a compreensão de um artefato, uma outra busca se faz necessária, descobrir o desenvolvedor que mais tem conhecimento sobre o artefato, e também, em alguns casos, especialistas em tarefas similares;

- *Determinar as entradas*: Os valores de todas as interações de edição dos desenvolvedores sobre os artefatos e as interações entre os desenvolvedores quando realizaram tarefas no projeto de desenvolvimento de software. Os valores dessas interações podem ser capturados via ferramentas de monitoramento disponíveis no ambiente de desenvolvimento. Na falta das interações sobre artefatos, uma alternativa é utilizar as informações de envio de artefatos para o repositório de código do projeto (*commits*). Os artefatos submetidos ao repositório de código, em sua maioria, foram editados na realização das tarefas;

- *Construir o recomendador*: O recomendador tem como objetivo recomendar contexto para cada nova tarefa do projeto de desenvolvimento de software, como definido anteriormente: 1- artefatos candidatos a serem editados pelos desenvolvedores na realização da nova tarefa; 2- desenvolvedores que tem potencial em apresentar conhecimento na nova tarefa; 3- tarefas realizadas similares à nova tarefa e; 4- especialistas nas tarefas similares à nova tarefa. Primeiro, o recomendador organiza o histórico de elementos do projeto em grupos de tarefas semelhantes em interação de edição em artefatos. Tarefas que estão relacionadas aos mesmos artefatos tendem a ficar no mesmo grupo, enquanto tarefas que não estão relacionadas aos mesmos artefatos tendem a ficar em grupos separados. Depois, a nova tarefa é relacionada aos grupos por similaridade textual. O recomendador então recomenda as tarefas e os artefatos dos grupos com maior similaridade textual à nova tarefa (Cap. 4). Para determinar o atual conhecimento dos desenvolvedores sobre as tarefas e artefatos recomendados, são utilizados os modelos de percepção do conhecimento (Cap. 3), em função dos dados de interação de edição dos desenvolvedores, levando em consideração os limites da capacidade humana de lembrar;

- *Entregar as recomendações*: As recomendações devem ser entregues ao desenvolvedor assim que a nova tarefa é selecionada para ser realizada. Os três primeiros grupos similares textualmente à nova tarefa devem ser apresentados por ordem decrescente de similaridade. Para cada grupo, são listadas as tarefas e os seus artefatos

relacionados, apresentando também os desenvolvedores que possuem conhecimento sobre artefato, tarefa e tarefas similares;

- *Avaliar a utilidade do recomendador*: Primeiro, é preciso avaliar se os modelos de percepção do conhecimento sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software convergem para uma informação adequada. Depois, é preciso avaliar se as tarefas similares e seus respectivos artefatos são relevantes para a realização de novas tarefas.

GASPARIC e JANES (2016) conduziram uma revisão quasi-sistemática da literatura para reportar o que os sistemas de recomendação estão recomendando na Engenharia de Software. A revisão incluiu artigos de 2003 até 2013, resultou na seleção de 46 artigos e seguiu com a extração de informações sobre as entradas, saídas, esforço e benefícios de cada sistema de recomendação. Os autores concluíram que é ainda preciso mais esforço para produzir SRESs. Por exemplo, SRESs para indicar o que os desenvolvedores precisam fazer. E também que é importante considerar outras informações relevantes, tais como às referente a processo de software. Nesta tese, faz-se uso das informações referentes às interações de edição dos desenvolvedores sobre os artefatos de software, enquanto os desenvolvedores realizavam tarefas no projeto de desenvolvimento de software.

Os SRESs são avaliados por diferentes métricas (ALMHANA *et al.*, 2016, GUO *et al.*, 2016, XIA *et al.*, 2015, YE *et al.*, 2014, ZHU et al., 2016). As métricas Cobertura, Precisão e F-measure são utilizadas para avaliar a classificação dos itens recomendados (ROBILLARD *et al.*, 2014). Cobertura representa qual a porcentagem dos itens utilizados estava presente na lista de itens recomendados. Precisão representa qual fração dos itens recomendados são itens que foram utilizados. F-measure representa a média harmônica entre Precisão e Cobertura. As Equações 1, 2 e 3 explicitam Cobertura, Precisão e F-measure. Essas métricas são também populares na área de Recuperação da Informação (SAKAI, 2014).

$$Cobertura = \frac{|itensRecomendados \cap itensUtilizados|}{|itensUtilizados|} \quad (1)$$

$$Precisão = \frac{|itensRecomendados \cap itensUtilizados|}{|itensRecomendados|} \quad (2)$$

$$F - measure = 2 \times \frac{Cobertura \times Precisão}{Cobertura + Precisão} \quad (3)$$

Redução representa o percentual do total de itens disponíveis que foi excluído da recomendação, definida na Equação 4. Por exemplo, no contexto de recomendações de artefatos em um projeto de desenvolvimento de software, quando o desenvolvedor inicia a correção de um erro, se ele tiver à disposição 400 artefatos e se um SRES recomendar 10 artefatos, então a Redução do contexto de trabalho é $1 - 10/400 = 97{,}5\%$. A relação entre Precisão e Cobertura é similar à relação entre Cobertura e Redução. Quando o objetivo é aumentar uma métrica, em consequência, normalmente, diminui a outra. Por isso, definimos Rc-measure como uma média harmônica entre Redução e Cobertura, Equação 5.

$$Redução = 1 - \frac{|itensRecomendados|}{|totalItens|} \quad (4)$$

$$Rc - measure = 2 \times \frac{Cobertura \times Redução}{Cobertura + Redução} \quad (5)$$

Finalmente, o teste de Spearman (BEST e ROBERTS, 1975) pode ser realizado sobre distribuições não-paramétricas com escala ordinal, intervalar ou razão. Nesta tese, Spearman é utilizado para verificar a similaridade entre listas de classificação de desenvolvedores (*rankings*). O coeficiente de correlação de Spearman é próximo de 1 (um) quando as listas são bem parecidas e próximo de zero quando são bem diferentes (GOSHTASBY, 2012).

2.3 Clusterização

Clusterização é uma técnica computacional para organizar objetos de dados (elementos) em grupos (*clusters*) com o intuito de prover uma organização para apoiar o ser humano no entendimento da informação. Elementos mais semelhantes tendem a ficar no mesmo grupo, enquanto elementos menos semelhantes, ou não semelhantes, tendem a ficar em grupos diferentes (HAN *et al.*, 2012; JAIN *et al.*, 1999). Os grupos semelhantes formados pela clusterização não possuem uma identificação de acordo com o conteúdo dos grupos, essa técnica também é conhecida como aprendizado não supervisionado.

Na literatura existem muitos métodos para clusterização de dados. Um dos mais conhecido e utilizado é o *K-means* (BOONGOEN e IAM-ON, 2018). No entanto, não existe um algoritmo que apresente resultados ótimos para todos os conjuntos de dados. Nesta tese é utilizado o algoritmo LNS_SMC (MONÇORES *et al.*, 2018) porque apresentou uma boa eficácia utilizando a medida *Modularization Quality* (MQ) aplicada ao problema de clusterização de módulos de software. A medida MQ é utilizada para medir a qualidade de clusterização dos módulos de software (classes, código fonte) de acordo com a coesão e acoplamento entre os módulos. Coesão é o número de dependências internas que um arquivo de código fonte tem em relação ao seu pacote, enquanto acoplamento é o número de dependências externas. Por exemplo, explicitamente, uma dependência é descrita ao importar uma classe em Java. O objetivo da clusterização é reorganizar o software em pacotes que apresentem alta coesão e baixo acoplamento.

Por outro lado, para produzir os módulos de software, o desenvolvedor faz uso de diversos artefatos em seu ambiente de trabalho com o objetivo de realizar tarefas, em muitos casos, faz edições para criar novos artefatos, edita artefatos existentes, ou simplesmente faz referência a artefatos que não podem ser editados. Nesse cenário, é possível associar tarefas a artefatos por meio das edições realizadas. Além disso, é possível capturar o grau de edição realizada em determinado artefato (KERSTEN, 2007). Nesta tese, com base na informação de que as tarefas estão relacionadas aos artefatos através das interações de edição dos desenvolvedores nos artefatos enquanto realizam tarefas, define-se que duas tarefas que possuem artefatos em comum são semelhantes por interação em artefatos. Quanto mais artefatos as tarefas possuírem em comum, mais semelhantes serão, e por isso tendem a ficar no mesmo grupo de tarefas semelhantes em um processo de clusterização.

A Equação 6 explicita a medida MQ. O seu entendimento é descrito através da perspectiva do problema de clusterização de tarefas similares por interação de edição. Deixe *C* representar o resultado de um algoritmo de clusterização com *n clusters*, então $C = \{C_1, C_2, C_3, .., C_n\}$. A medida de qualidade MQ de um *cluster* C é definida como o somatório da qualidade de cada *cluster* C_k, $MF(C_k)$. O valor de MF está em função do número de associações internas (i) e externas (j) das tarefas de C_k. Por exemplo, quando todas as tarefas de um *cluster* C_k não tiverem artefatos comuns com tarefas do mesmo C_k, isto é, i é igual a zero e por isso MF tem valor zero, o pior caso. No melhor caso, todas as

tarefas de um *cluster* C_k estão relacionados com os mesmos artefatos e não possuem artefatos em comum com outros *clusters*. Dessa forma, $i > 0$, j é igual zero, e MF tem valor máximo igual a 1 (um).

$$MQ = \sum_{k=1}^{n} MF(C_k) \qquad MF(C_k) = \begin{cases} 0, & i = 0 \\ \dfrac{i}{i + \frac{j}{2}}, & i > 0 \end{cases} \tag{6}$$

O algoritmo LNS_SMC é a sigla em inglês para indicar que o algoritmo é classificado como uma heurística para buscar uma solução ótima explorando uma vizinhança grande (LNS) e foi aplicado ao problema de clusterização de módulos de software (SMC). A medida MQ é utilizada pelo LNS_SMC para comparar as soluções e escolher a com maior MQ. O algoritmo inicialmente avalia o MQ de uma solução, elegendo-a como ótima, depois faz uso de dois operadores para escolher o vizinho da solução ótima. O operador *destroy* é aplicado sobre a solução ótima para gerar uma solução incompleta, por exemplo a remoção de alguns elementos. Em seguida, é aplicado o operador *repair* na solução incompleta, gerando uma solução válida, por exemplo, a inserção dos nós removidos em outros *clusters* da solução incompleta. Depois, é verificado se o MQ da solução válida gerada é maior que o da atual solução ótima. Caso afirmativo, a solução válida se torna a atual solução ótima. Esses passos são executados até uma condição de parada.

2.4 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) é formado por um conjunto de técnicas computacionais motivadas por teoria para a análise e representação automáticas das linguagens utilizadas por humanos (CAMBRIA e WHITE, 2014). As técnicas e modelos projetados para um idioma, na maioria das vezes, não são facilmente generalizados para outros idiomas (GREEN e MANNING, 2010, LEVY e MANNING, 2003). De acordo com CAMBRIA e WHITE (2014), a pesquisa em PLN iniciou com o paradigma de análise das palavras, evolui para a análise dos conceitos e é esperado que os modelos possam entender as narrativas em um futuro breve.

O espaço vetorial é o modelo de representação textual mais conhecido, onde as palavras são representadas por vetores e a ordem em que elas aparecem no texto não é considerada (MINER *et al.*, 2012). Primeiro é preciso fazer o pré-processamento do texto. Nesse passo, o texto é quebrado em palavras, conhecido como *tokenize*. Em seguida há a remoção das palavras mais conhecidas do idioma, por exemplo em inglês, das palavras

the, by e *for* (*stopwords*). Prossegue-se eliminando os prefixos e sufixos, por exemplo para *talk*, *talking* e *talks* os sufixos são retirados e a forma reduzida passa a ser *talk* (o significado gramatical), passo conhecido como *stemming*. Dependendo da origem dos dados, talvez seja preciso fazer a correção ortográfica de algumas palavras (*spelling*). Em algumas situações, pode haver a necessidade da detecção das sentenças ao longo do texto (*sentence boundary detection*). E, finalmente, as palavras devem ir para um padrão, todas em minúsculas ou todas em maiúsculas (*case normalization*).

Uma vez realizado o pré-processamento do texto, é possível representá-lo no espaço vetorial. Por exemplo, a Tabela 2-A ilustra se um documento apresenta ou não determinado termo do conjunto de termos resultantes do processo de pré-processamento. Os documentos são dispostos como linhas e os termos são dispostos, ordenados alfabeticamente, como colunas. Para cada par linha X coluna, o valor do par recebe 1 (um) se o documento apresenta o termo e 0 (zero), caso contrário.

A representação binária não permite informar o número de vezes que um termo ocorreu em um documento. A Tabela 2-B ilustra a representação considerando o número de ocorrências com que os termos aparecem nos documentos, conhecida como TO (*Term Occurrences*). Por exemplo, o formato reduzido da palavra *September* aparece duas vezes no terceiro documento.

Para considerar as frequências com que os termos aparecem nos documentos, utiliza-se a medida TF (*Term Frequency*), resultado da divisão do número de ocorrências do termo no documento pelo número total de termos presentes no documento. A Tabela 2-C mostra os valores de TF normalizados. O termo *abl* ocorre uma vez no terceiro documento (doc_3) e o doc_3 tem seis termos, incluindo os termos repetidos (veja Tabela 2-B), assim os TFs para o doc_3 são: 1/6+1/6+0/6+2/6+1/6+1/6. O TF normalizado é obtido dividindo o valor do TF pela raiz quadrada de cada TF, para o doc_3 temos $\sqrt{(1/6)^2 + (1/6)^2 + (0/6)^2 + (2/6)^2 + (1/6)^2 + (1/6)^2} = 0{,}47$, então TF normalizado para o temo *abl* no doc_3 é $\frac{1/6}{0{,}47} = 0{,}35$, ou simplesmente chamado TF[10]. Para o termo *septemb*, também no doc_3, o TF= $\frac{2/6}{0{,}47} = 0{,}71$. Para mais detalhes sobre TF e outras medidas tais como IDF e TF-IDF, leia esta referência (MANNING *et al.*, 2008).

[10] O RapidMiner calcula o TF como normalizado.

Tabela 2 - Vetores de documentos gerados[11] pelo RapidMiner[12] utilizando os operadores Transform Cases (implementa *case normalization*), Tokenize, Filter Stopwords, Stem e Filter Tokens (seleciona tokens com número de caracteres entre 4 e 25).

Tabela 2–A, *Binary Term Occurrences*		abl	finish	plan	septemb	thesi	write
doc_1	My thesis was planned to finish in September.	0	1	1	1	1	0
doc_2	I am writing my thesis.	0	0	0	0	1	1
doc_3	I was able to finish my thesis writing in September. September 21.	1	1	0	1	1	1

Tabela 2–B, *Term Occurrences*		abl	finish	plan	septemb	thesi	write
doc_1	My thesis was planned to finish in September.	0	1	1	1	1	0
doc_2	I am writing my thesis.	0	0	0	0	1	1
doc_3	I was able to finish my thesis writing in September. September 21.	1	1	0	2	1	1

[11] Tabela 2-A: Operador *Process Documents from Data* com parâmetro *Binary Term Occurrences.* Tabela 2-B: Operador *Process Documents from Data* com parâmetro *Term Occurrences.* Tabela 2-C: Operador *Process Documents from Data* com parâmetro *Term Frequency.*

[12] RapidMiner Studio 9.0.001 com licença educacional.

Tabela 2–C, *Term Frequency*

	abl	finish	plan	septemb	thesi	write
doc_1	0	0,5	0,5	0,5	0,5	0
doc_2	0	0	0	0	0,71	0,71
doc_3	0,35	0,35	0	0,71	0,35	0,35

Uma vez os documentos representados em forma vetorial, é possível calcular a similaridade entre os documentos. A medida de similaridade mais utilizada calcula a similaridade entre os documentos através do cosseno entre os vetores que os representam. O cosseno de similaridade entre dois documentos, $sim(d_1, d_2)$, é calculado pela divisão entre o produto vetorial $(\overrightarrow{d_1} \cdot \overrightarrow{d_2})$ e o produto dos módulos dos vetores $(||d_1|| \cdot ||d_2||)$. Então, para calcular $sim(doc_1, doc_2)$ com representação vetorial utilizando TF, Tabela 2-C, calcula-se $\overrightarrow{d_1} \cdot \overrightarrow{d_2} = (0 \times 0 + 0{,}5 \times 0 + 0{,}5 \times 0 + 0{,}5 \times 0 + 0{,}5 \times 0{,}71 + 0 \times 0{,}71) = 0{,}355$, e os módulos de $doc_1 = \sqrt{0^2 + 0{,}5^2 + 0{,}5^2 + 0{,}5^2 + 0{,}5^2 + 0^2} = 1$ e $doc_2 = \sqrt{0^2 + 0^2 + 0^2 + 0^2 + 0{,}71^2 + 0{,}71^2} = 1{,}00409$. Assim $sim(doc_1, doc_2) = 0{,}355/(1 \times 1{,}00409) = 0{,}354$. Os valores limites de $sim(d_1, d_2)$ são 0 (zero) e 1 (um), zero quando os documentos não possuem similaridade e 1 (um) quando a similaridade é máxima. A Tabela 3 ilustra os valores do cosseno de similaridade entre doc_1, doc_2 e doc_3 nas representações com *Binary Term Occurrences, Term Occurrences, Term Frequency* e TF-IDF.

Tabela 3 - Similaridade textual por cosseno entre os documentos da Tabela 2, utilizando BTO, BT, TF e TF-IDF calculados com a ferramenta RapidMiner[13].

	doc_1		doc_2		doc_3	
doc_1	**BTO** 1	**TO** 1	0,354	0,354	0,671	0,707
	TF 1	**TF-IDF** 1	0,354	0	0,707	0,269
doc_2			1	1	0,632	0,500
			1	1	0,500	0,274
doc_3					1	1
					1	1

[13] RapidMiner Studio 9.0.001 com licença educacional.

No entanto, LI e JAN (2013) mostraram que a medida de similaridade por cosseno apresenta alguns problemas. Eles propuseram uma extensão combinando *cosine similarity* and *Weighted Hamming Distance*. A combinação pode gerar várias extensões, dentre as quais uma extensão (chamada pelos autores de *dw-cosine*) mostrou-se melhor que *cosine similarity* quando aplicada a um conjunto de textos.

2.5 Percepção

DOURISH e BELLOTTI (1992) definiram Percepção (*Awareness*) como um entendimento das atividades dos outros, o que fornece um contexto para a sua própria atividade. A palavra atividade foi mantida como na definição original, mas para melhor compreensão, a descrição de Percepção deve ser entendida como uma compreensão das **tarefas** dos outros para fornecer um contexto para suas próprias **tarefas**. Isso se justifica porque uma tarefa acontece em tempo de execução, bem como a identificação da percepção, em que o contexto abrange as necessidades individuais de ter conhecimento sobre o ambiente para melhor atender aos objetivos do grupo. Mais alinhado com o escopo desta tese, STOREY *et al.* (2005) observaram que a Percepção consiste em saber quem está trabalhando no projeto, o que eles estão fazendo, que artefatos eles estão manipulando, e como o trabalho individual pode afetar o trabalho dos outros. Isso indica que o contexto muda com o tempo em função de que novas tarefas ou pessoas podem ser adicionadas ao projeto.

A Percepção está classificada em tipos: Percepção do Ambiente de Trabalho (*Workspace Awareness*): atualizando o conhecimento sobre as interações dos outros participantes em ambiente compartilhado; Percepção Informal (*Informal Awareness*): informação geral, de quem está no ambiente, o que está fazendo e o que fará; Percepção da Estrutura de Grupo (*Group-Structural Awareness*): conhecimento dos papéis e responsabilidades das pessoas; Percepção Social (*Social Awareness*): informações sobre a presença e atividades (tarefas) das pessoas em um ambiente compartilhado; Percepção do Contexto (*Context Awareness*): a evolução da informação do estado interno e externo que caracteriza totalmente a situação de cada entidade em um ambiente compartilhado (OMORONYIA *et al.*, 2010).

GUTWIN *et al.* (1996) apresentaram onze elementos da Percepção do Ambiente de Trabalho. Cada elemento está associado a pelo menos uma pergunta que precisa ser respondida para melhorar a percepção. As perguntas são iniciadas por *que*, *onde* e *quem*.

Por exemplo, duas perguntas sobre mudanças são feitas: 1 - Quais mudanças estão fazendo? 2 - Onde as mudanças estão sendo feitas? A lista completa foi apresentada em GUTWIN *et al.* (1996) e reproduzida em OMORONYIA *et al.* (2010), que também acrescenta os elementos de Percepção Social e Percepção do Contexto. A Percepção do Contexto, segundo OMORONYIA *et al.* (2010), baseia-se em informações sobre tarefas (história, tempo, duração e paralelismo). A identificação de especialistas está diretamente relacionada aos elementos habilidade e identificação. O elemento identificação é caracterizado pela resposta à pergunta "Quem está participando da tarefa?" E o elemento habilidade é endereçado pela pergunta "Quais tarefas eles podem realizar?".

A Percepção tem um papel importante no trabalho colaborativo. A Colaboração pode ser entendida como o ato de trabalhar em conjunto (SOLIMAN *et al.*, 2005). Essa definição é tão genérica que SOLIMAN *et al.* (2005) definiram ingredientes para auxiliar na sua identificação: pessoas, ambiente compartilhado, tempo, objetivo comum, foco no objetivo, linguagem comum, conhecimento na área do objetivo e interação. Por outro lado, FUKS *et al.* (2007) exploram o modelo 3C que foi originalmente apresentado por ELLIS *et al.* (1991). O modelo 3C de FUKS *et al.* (2007) é composto por Comunicação, Coordenação e Cooperação. A Coordenação faz a conexão entre Comunicação e Cooperação para fomentar a Colaboração, e Cooperação no contexto de grupo de trabalho é um conjunto de operações durante uma sessão em um ambiente de trabalho compartilhado. Nesse sentido, o modelo 3C enfatiza a importância da Percepção, ou seja, a percepção das pessoas em relação às suas tarefas e às outras pessoas em um contexto de colaboração.

A Percepção também tem um papel importante na Engenharia de Software Colaborativa (ESC). A ESC faz uso de um conjunto de técnicas e ferramentas com o objetivo de unir os esforços das pessoas na construção de software (WHITEHEAD *et al.*, 2010). Nesse cenário, pessoas com habilidades distintas colaboram usando seus conhecimentos em diferentes níveis, de acordo com o papel assumido no projeto. As ferramentas de percepção permitem que os desenvolvedores se conscientizem das ações dos outros, ajudando a evitar conflitos. Um exemplo é o Syde, um plugin do Eclipse que monitora as ações dos desenvolvedores e aponta possíveis conflitos que podem surgir devido às edições simultâneas de código feitas pelos desenvolvedores (HATTORI e LANZA, 2010). Um desafio atual na ESC consiste na definição de métodos apropriados para identificar quais desenvolvedores têm conhecimento para realizar tarefas específicas.

2.6 Conhecimento

O conhecimento é um ativo invisível e intangível e não pode ser observado diretamente (HUNT, 2003). O conhecimento implícito representa o conhecimento adquirido através da experiência, envolvendo, por exemplo, pontos de vista, compromissos, atitudes etc. O conhecimento explícito é adquirido nas escolas e universidades e é registrado nos livros e manuais. O conhecimento implícito é muito pessoal e difícil de ser expresso, enquanto que o conhecimento explícito pode ser expresso através de uma linguagem (NONAKA e TAKEUCHI, 1996). Esses dois tipos de conhecimento estão presentes na realização dos projetos, variando de proporção de projeto para projeto (KOSKINEN *et al.*, 2003).

A Gestão do Conhecimento apoia o conhecimento do como, do quem, do quê, do quando e do por que (RUS e LINDVALL, 2002), e software é desenvolvido por pessoas com diferentes níveis de conhecimento (FUGGETTA e DI NITTO, 2014). Por isso, é necessário gerenciar e compartilhar conhecimento para manter a produtividade. A produtividade pode ser afetada, tanto quando um desenvolvedor experiente sai, quanto quando um recém-chegado é adicionado a um projeto. Nesse cenário, um problema recorrente é a identificação de especialistas (BANI-SALAMEH *et al.*, 2010, EHRLICH e SHAMI, 2008, WHITEHEAD *et al.*, 2010). Por exemplo, um especialista pode ajudar um novato no desempenho de uma tarefa. Essa ajuda pode ser via comunicação direta entre novatos e especialistas (através de chat, e-mail etc) e/ou observando os artefatos produzidos pelos especialistas.

A identificação de especialistas tem como objetivo descobrir quem tem conhecimento sobre elementos de software, como artefatos e tarefas. Portanto, é necessário definir medidas para caracterizar a quantidade de conhecimento dos desenvolvedores (HATTORI *et al.*, 2012, ROBBES e RÖTHLISBERGER, 2013). Dessa forma, com base em medidas de conhecimento, é possível, em princípio, incentivar o compartilhamento de conhecimento entre os mais experientes e os novatos, e especialmente no início de uma nova tarefa, quando os novatos precisam de mais ajuda (SOUZA e REDMILES, 2011).

O conhecimento está diretamente relacionado à capacidade de aprendizado, reaprendizado, esquecimento e à memorização da informação. O aprendizado e o esquecimento dependem de muitas variáveis, como o tipo de material que está sendo aprendido, a motivação para aprender e o conhecimento prévio dos aprendizes (THALHEIMER, 2010). Portanto, há a necessidade de considerar a depreciação do conhecimento ao longo do tempo (BARBOSA *et al.*, 2015, HATTORI *et al.*, 2012,

ZHANG e ZHOU, 2009). Na área da Psicologia Cognitiva existem muitos modelos genéricos para a memória, a chamada "curva do esquecimento", mas não há consenso sobre o assunto (ROEDIGER, 2007). BARBOSA *et al.* (2015) definiram um método para medir o conhecimento com base na curva de aprendizado. O modelo de depreciação do conhecimento de BARBOSA *et al.* (2015) indica que um especialista mantém 65% do conhecimento original após 10 anos de inatividade.

Especificamente sobre arquivos de código fonte, KRÜGER *et al.* (2018) investigaram a influência do esquecimento nos desenvolvedores. O estudo foi realizado com 60 programadores de 10 projetos do GitHub. O resultado indicou que as edições dos desenvolvedores têm correlação positiva com o conhecimento dos desenvolvedores sobre os arquivos de código fonte. A conclusão é que os desenvolvedores, em média, esquecem 50% de um arquivo em 40 dias, caso não revisitem o arquivo nesse período. Além disso, o valor mediano de 65 dias foi informado como uma boa aproximação para o esquecimento do conteúdo de um arquivo de código fonte.

Tem havido um crescente interesse sobre a forma matemática precisa das curvas de esquecimento (MURRE e CHESSA, 2011, MURRE *et al.*, 2013, MURRE e DROS, 2015, RUBIN e WENZEL, 1996, RUBIN *et al.*, 1999, WIXTED e EBBESEN, 1991). Diversos experimentos foram realizados para avaliar as funções candidatas para se adequar aos dados experimentais. Essas funções candidatas incluem (TIBBLES, 2017), dentre outras, onde *a* e *b* são parâmetros em função do tempo *t*:

Linear: $y = a - bt$

Exponencial: $y = a\, e^{-bt}$

Hiperbólica: $y = \frac{1}{(a + bt)}$

Logaritmo: $y = a - b \log(t)$

Potência: $y = a\, t^{-b}$

Exponencial-Raiz Quadrada: $y = a\, e^{-b\sqrt{t}}$

Hiperbólica-Raiz Quadrada: $y = \frac{1}{(a + b\sqrt{t})}$

Esses estudos confirmaram a inadequação de modelos lineares para prever como a retenção de memória diminui ao longo do tempo (RUBIN e WENZEL, 1996) e mostraram que uma das funções que produzem um bom ajuste é a função exponencial (MURRE e CHESSA, 2011).

2.7 Considerações Finais

Esta tese investiga o quanto as interações registradas em projetos de desenvolvimento de software podem contribuir para medir o conhecimento dos desenvolvedores e também produzir recomendações adequadas de contexto para novas tarefas. Nesse cenário, este capítulo apresentou a definição de Projeto e como está relacionado com Processo e Processo de Software.

O capítulo seguiu introduzindo a área de Sistemas de Recomendação para a Engenharia de Software, indicando uma forma de construção e evidenciando as métricas utilizadas no capítulo de avaliação desta tese (Capítulo 5). Depois, também foram apresentados os conceitos de clusterização e o algoritmo de clusterização utilizado na avaliação (Capítulo 5). Além disso, o capítulo discutiu algumas técnicas utilizadas no processamento de linguagem natural, em especial as utilizadas no capítulo de avaliação.

Este capítulo ainda esclareceu os conceitos de Percepção e Conhecimento, mostrando a diferença entre os conhecimentos implícito e o explícito, além de listar algumas modelagens de conhecimento discutidas na literatura.

Os conceitos e técnicas apresentados neste capítulo visam facilitar o entendimento dos Modelos Tacin de Percepção do Conhecimento (Capítulo 3), do Método Tacin para Recomendar Contexto de Nova Tarefa (Capítulo 4) e das avaliações (Capítulo 5).

3 - Modelos Tacin de Percepção do Conhecimento

Este capítulo trata do objetivo específico desta tese, que consiste em identificar especialistas em projetos de desenvolvimento de software: medir o conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software. O capítulo define as interações Desenvolvedor-Artefato, Desenvolvedor-Desenvolvedor e tarefas similares com base nas interações Desenvolvedor-Artefato. Logo após, formaliza matematicamente os quatro modelos para medir o conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software, levando em consideração explicitamente o esquecimento dos desenvolvedores.

Os modelos de percepção do conhecimento apresentados neste capítulo utilizam o grau de interação, dos desenvolvedores com artefatos e entre desenvolvedores, para medir o conhecimento dos desenvolvedores sobre um projeto de desenvolvimento de software ou parte dele. Essas interações são denominadas de Desenvolvedor-Artefato (I_{da}) e Desenvolvedor-Desenvolvedor (I_{dd}). A interação do tipo I_{da} acontece quando o desenvolvedor realiza uma ação de edição em um artefato para realizar uma tarefa. A interação I_{dd} acontece quando os desenvolvedores conversam sobre a realização da tarefa.

A Figura 5 mostra a representação das interações I_{da} e I_{dd} com base no modelo de proveniência da W3C apresentado no Capítulo 2. Definimos os tipos Artefato (*Artifact*), Tarefa (*Task*) e Desenvolvedor (*Developer*). O tipo Artefato é uma especialização do tipo Entidade (*Entity*) para representar arquivos digitais tais como código fonte e documentação. O tipo Desenvolvedor representa uma pessoa do tipo Agente (*Agent*) que tem conhecimento e autorização para realizar tarefas. A Tarefa é uma especialização de Atividade (*Activity*) e pode representar uma correção de erro ou a adição de uma nova funcionalidade ao software. Dependendo da ferramenta utilizada, a tarefa pode aparecer com outros nomes. No Bugzilla, uma tarefa é chamada de *Bug* e no GitHub aparece como *Issue*.

Na Figura 5, a interação I_{da} está representada pelo relacionamento *WasChangedBy* entre Artefato e Tarefa. Esse relacionamento indica que um artefato pode ser alterado por uma ou mais tarefas. Para cada alteração em um artefato, são registrados o grau (*degree*) e o tempo final da alteração (*time*). Essas informações dependem do ambiente de desenvolvimento, por exemplo, o grau pode ser expresso pelo número de

linhas alteradas, pelo número de caracteres, ou pelo resultado de um modelo de interação, por exemplo, o Mylyn (KERSTEN, 2007). A Figura 6 mostra um exemplo de interação I_{da}. O registro dessa interação mostra que o desenvolvedor *Maria* alterou o artefato *main.java* enquanto realizava a tarefa com *id* igual a 345. A alteração aconteceu em 10/10/2017 às 13h31min10s com grau 5.

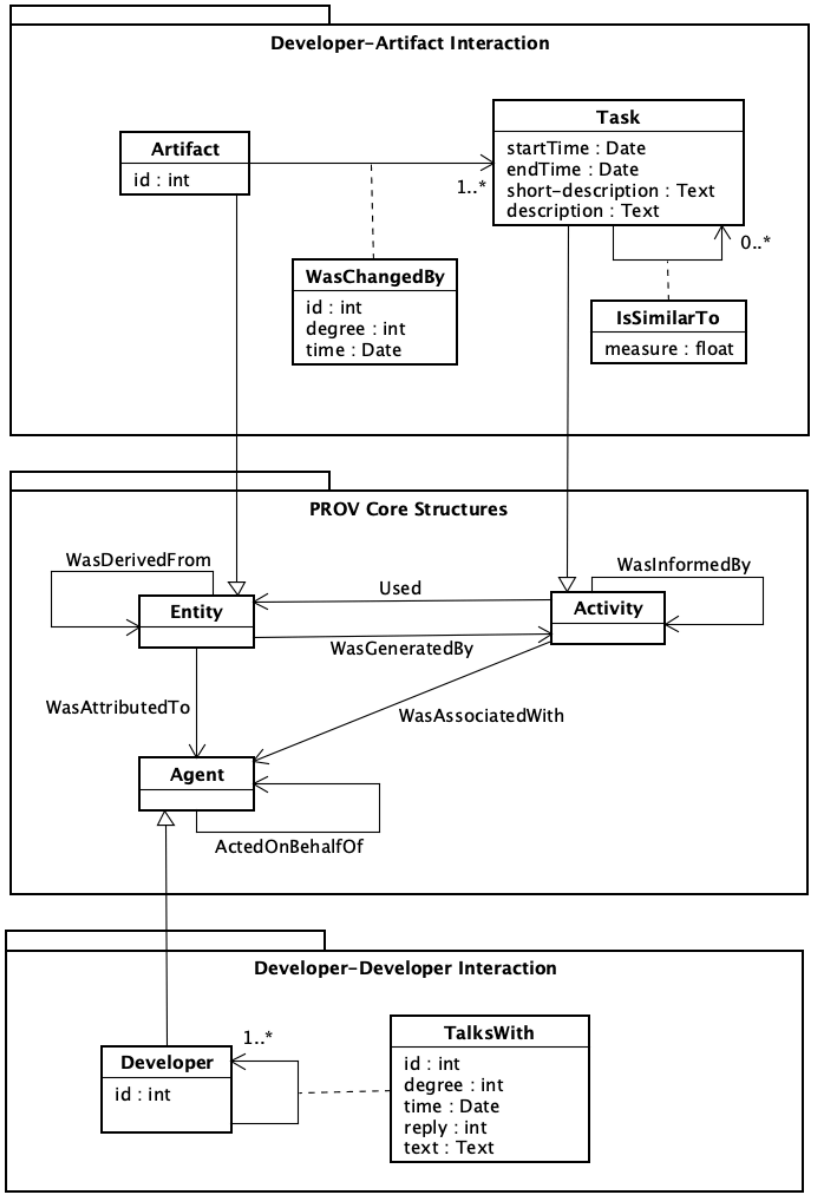

Figura 5 – Interações Desenvolvedor-Artefato e Desenvolvedor-Desenvolvedor.

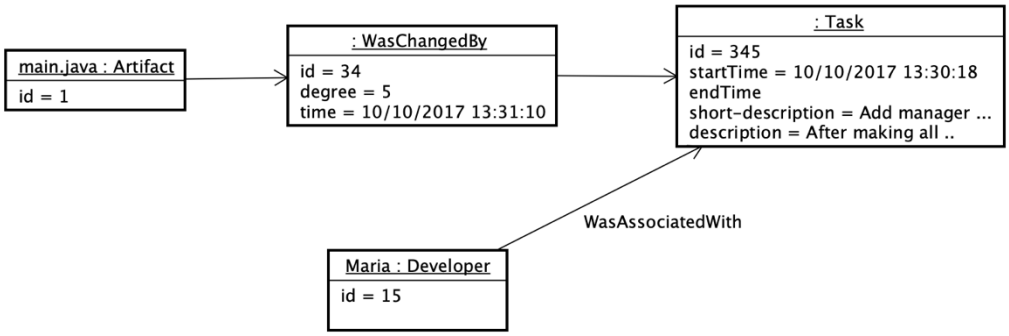

Figura 6 – Exemplo de interação Desenvolvedor-Artefato.

A interação I_{ad} representa as conversas dos desenvolvedores sobre a realização de cada tarefa. O modelo da Figura 5 expressa que um desenvolvedor pode conversar com um ou mais desenvolvedores. Uma conversa é composta por comunicações entre os envolvidos, com registro do grau, do tempo e o atributo *reply* indica que a comunicação tem relação com outra comunicação. Essas informações dependem do ambiente de desenvolvimento em que o projeto é conduzido. No GitHub, por exemplo, as conversas são registradas de forma textual através da troca de mensagens. A Figura 7 mostra um exemplo de interação I_{ad}, o registro da conversa entre *Maria* e *João* sobre a realização da tarefa 345. *Maria* envia uma mensagem textual para *João* no dia 10/10/2017 às 13h33min17s. A resposta de *João* acontece no dia 11/10/2017 às 10h13min10s.

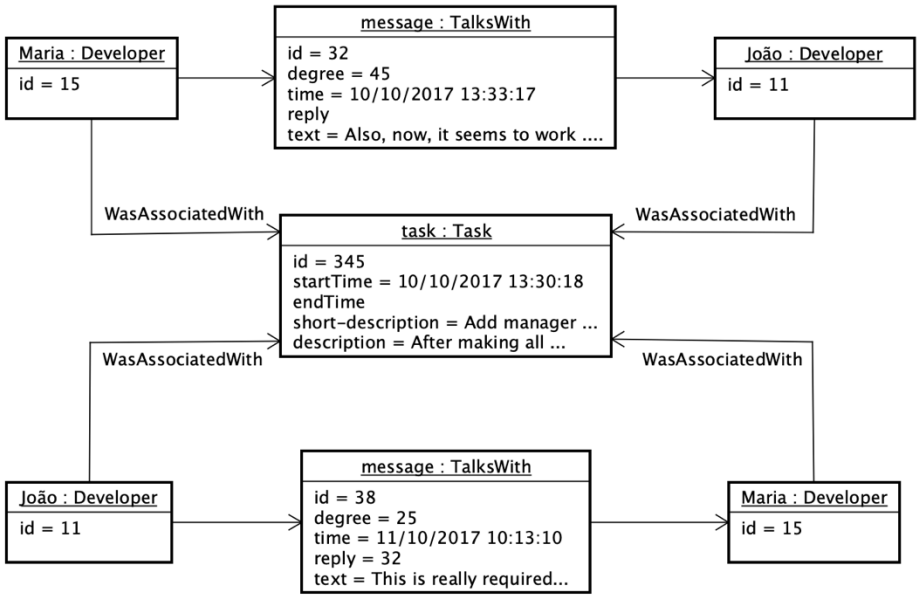

Figura 7 - Exemplo de interação Desenvolvedor-Desenvolvedor.

O modelo da Figura 5 também expressa que uma tarefa pode ter similaridade com outras tarefas. Nesta tese, as tarefas do histórico do projeto de software são organizadas em grupos de tarefas similares de acordo com as interações do tipo I_{da}. A Figura 8 ilustra a similaridade entre duas tarefas considerando essas interações. Uma tarefa está representada por um triângulo e um artefato por um quadrado. As interações I_{da} estão representadas por setas das tarefas para os artefatos. Dessa forma, define-se que as tarefas *T239875* e *T245189* têm similaridade com peso igual a 3 porque os desenvolvedores realizaram interações I_{da} sobre os três artefatos *A419*, *A425* e *A1243* enquanto trabalhavam nessas tarefas.

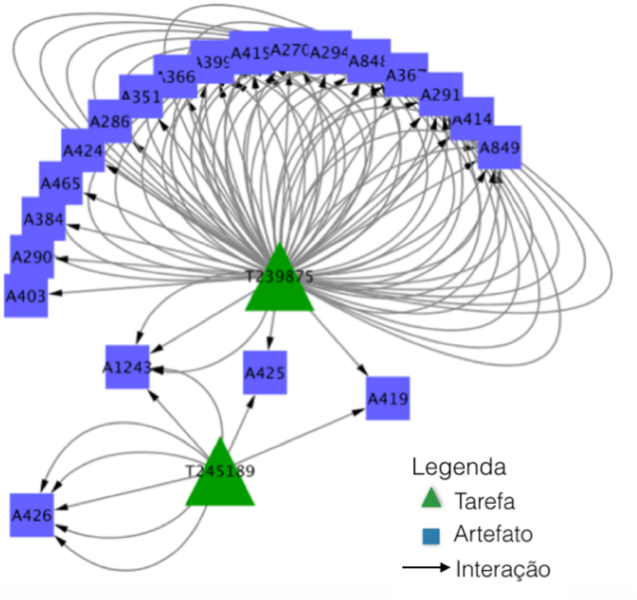

Figura 8 – Exemplo de tarefas similares por interação Desenvolvedor-Artefato[14].

A seguir são apresentados os quatro modelos orientados ao conhecimento. Os modelos usam as interações do desenvolvedor para expressar o seu conhecimento sobre os elementos do projeto de software, como artefatos, tarefas, tarefas semelhantes e todo o projeto de software. Considerando que o termo medição é definido como uma redução quantitativa de incertezas baseada em uma ou mais observações (HUBBARD, 2007), pode-se então dizer que os modelos propostos medem o conhecimento do desenvolvedor com base nas interações I_{da} e I_{dd}, considerando a depreciação do conhecimento ao longo

[14] Visualização produzida com o apoio da ferramenta Cytoscape 3.6.0 (https://cytoscape.org/).

do tempo, ou seja, as interações mais antigas contribuem menos para a medida do conhecimento do que as interações mais recentes.

3.1 Medida de Conhecimento do Desenvolvedor sobre um Artefato

O modelo K_a mede o conhecimento do desenvolvedor sobre um artefato fazendo uso das interações I_{da}. De acordo com o modelo da Figura 5, define-se $I_{da}(d, a, f_{endtime})$ como o grau da interação do desenvolvedor *d* sobre o artefato *a* registrado no tempo final ($f_{endtime}$) da interação. Como cada artefato possui um grau de dificuldade de produção e manutenção, então cada interação precisa ser normalizada pela atual interação máxima registrada sobre o artefato.

A Equação 7 define II_{da} como a interação normalizada de uma interação I_{da}. Deixe *D* representar o conjunto de todos os desenvolvedores que interagiram por meio de interação I_{da} com o artefato *a* até a data-hora *t*. Então $II_{da}\ (d, a, t)$ é o resultado da divisão de $I_{da}\ (d, a, t)$ pela interação máxima I_{da} dentre todos os desenvolvedores sobre o artefato *a* até o tempo *t*. Por exemplo, de acordo com a Figura 9, o desenvolvedor *d1* produziu três interações no artefato *A1*, registradas em 31/07/2007 às 13h, 15h30 e 16h, com graus iguais a 23, 15 e 18, respectivamente. O desenvolvedor *d2* produziu duas interações no artefato *A1*, a primeira registrada em "31/07/2007 15h45" com grau 33 e a outra em "11/08/2007 13h" com grau 50. Uma vez que I_{Da}= (23,15,18,33,50), a interação máxima registrada ($max_{d \in D}\ I_{da}$) foi 50, realizada pelo desenvolvedor *d2*. Então as medidas de II_{da} para *d1* são $\{\frac{23}{50}, \frac{15}{50}, \frac{18}{50}\}$ e para *d2* são $\{\frac{33}{50}, \frac{50}{50}\}$. Portanto, a II_{da} representa o valor de uma interação I_{da} de um desenvolvedor em relação às outras interações I_{da} produzidas pelos outros desenvolvedores em um mesmo artefato.

$$II_{da}(d, a, t) = \frac{I_{da}(d, a, t)}{max_{d \in D, \forall t_i}\ I_{da}(d, a, t_i)} \quad (7)$$

O modelo K_a também considera o esquecimento e a reaprendizagem do desenvolvedor. A função hiperbólica foi escolhida porque é uma forma funcional simples que apresenta regularidade para modelar esquecimento[15] (CHESSA e MURRE, 2009, LEE, 2004). A Equação 8 modela o quanto o desenvolvedor lembra das suas interações considerando a reaprendizagem, R_d. A constante de esquecimento *b* foi definida em 0,025 para calibrar a depreciação do conhecimento, de acordo com o estudo de KRÜGER *et al.*

[15] do original em inglês *memory forgetting*

(2018). Esse estudo indica que os desenvolvedores, em média, esquecem 50% de um arquivo em 40 dias, caso não revisitem o arquivo nesse período. O Δt_i é o número de dias que passaram desde o registro de uma interação $I_{da}(d, a, f_{endtime})$.

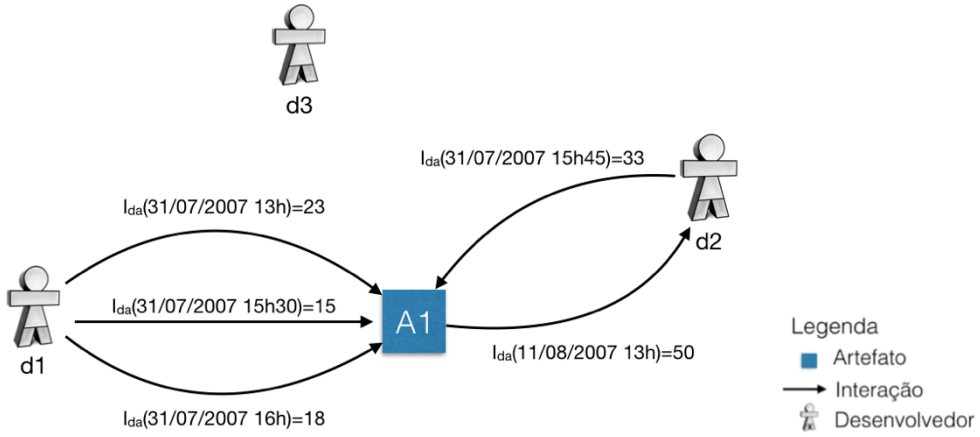

Figura 9 - Exemplos de interações Desenvolvedor-Artefato.

A função que modela a interação recente do desenvolvedor sobre um artefato e seus vizinhos é definida como $G_d(n) = \frac{7}{e^n+6}$, onde *n* é o número de dias, calculado nos últimos 7 dias, em que o desenvolvedor *d* produziu interação I_{da}: 1- sobre o artefato *a* e/ou; 2- sobre os artefatos relacionados com as tarefas que têm relação por meio de interação I_{da} com o artefato *a*. Quanto menor for G_d, menor é a depreciação do conhecimento do desenvolvedor sobre o artefato *a*, isto é, consideramos que houve reaprendizagem sobre as tarefas relacionadas com o artefato *a* nesse período. A Equação 9 apresenta G_d e a Figura 10 mostra o seu gráfico.

$$R_d(n) = \frac{1}{(G_d(n) \cdot b \cdot \Delta t) + 1} \tag{8}$$

$$G_d(n) = \frac{7}{e^n + 6} \tag{9}$$

$$onde\ n\ é\ um\ inteiro\ tal\ que\ 0 \leq n \leq 7$$

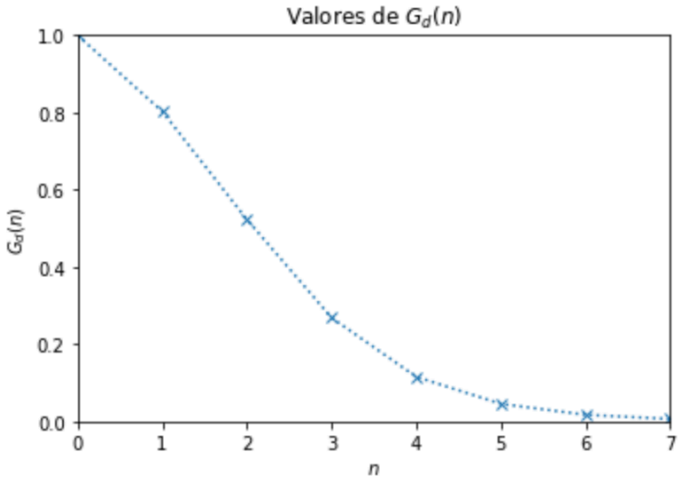

Figura 10 – O gráfico de G_d.

As Figuras 11-14 mostram os valores de R_d para os primeiros 10, 40, 100 dias e primeiro ano. A reaprendizagem tem o mesmo comportamento para artefatos que tiveram interações recentes e distantes no tempo. Para n igual a zero, não houve reaprendizagem, mostrando uma curva com valores menores para R_d. Por outro lado, quando o desenvolvedor produzir, em todos os últimos sete dias, interação do tipo I_{da} sobre um artefato a e/ou sobre os artefatos relacionados com as tarefas que têm relação por meio de interação I_{da} com o artefato a, n igual a 7, então os valores da curva R_d terão valores maiores. Os gráficos das Figuras 11-14 também mostram que quanto maior for o valor de n, maiores são os valores de R_d.

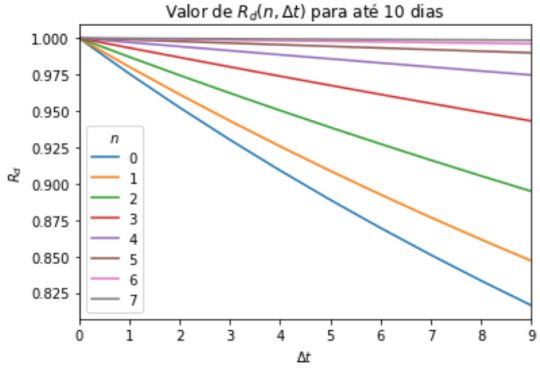

Figura 11 - Gráfico de R_d até 10 dias.

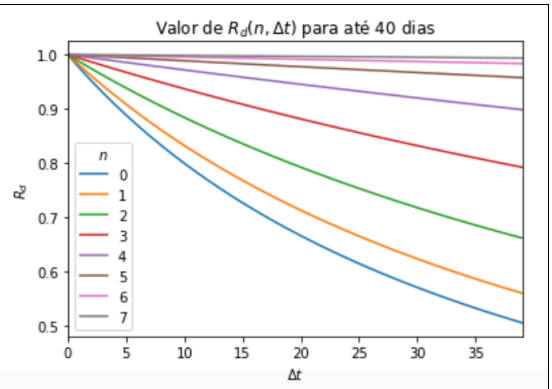

Figura 12 - Gráfico de R_d até 40 dias.

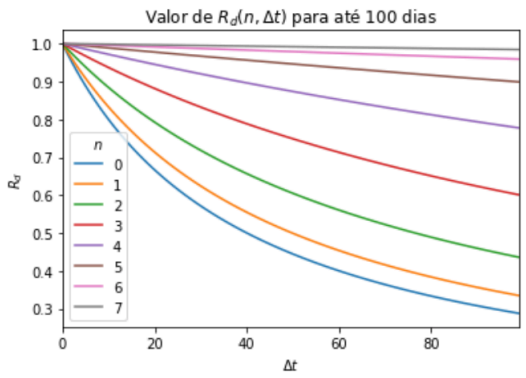

Figura 13 - Gráfico de R_d até 100 dias.

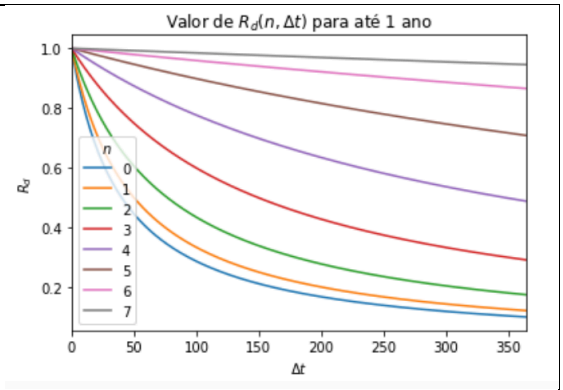

Figura 14 - Gráfico de R_d até um ano.

As Tabelas 4 e 5 mostram as influências dos valores de n e Δt em R_d. Para n igual a zero, sem reaprendizagem nos últimos sete dias, o desenvolvedor se lembra de 97,6% da interação I_{da} realizada no dia anterior ($\Delta t = 1$). No quadragésimo dia ($\Delta t = 40$), o desenvolvedor se lembra de 50% da interação e de somente 10% após um ano. Para n igual a sete, com reaprendizagem, o desenvolvedor se lembra totalmente das interações I_{da} realizadas nos últimos 3 dias. No quadragésimo dia, se lembra de 99,4% da interação e em um ano, lembra-se de 94,5%. As Tabelas 4 e 5 também apresentam os valores para todos os valores de n (0-7) e para outros valores de Δt (1-7, 38-41, 63-67, 361-365).

Tabela 4 - Os valores de R_d para Δt de 0-7 e de 38-41.

	Δt em dias											
n	0	1	2	3	4	5	6	7	38	39	40	41
0	1	0,976	0,952	0,930	0,909	0,889	0,870	0,851	0,513	0,506	0,500	0,494
1	1	0,980	0,961	0,943	0,926	0,909	0,893	0,877	0,567	0,561	0,555	0,549
2	1	0,987	0,975	0,962	0,950	0,939	0,927	0,916	0,668	0,662	0,657	0,651
3	1	0,993	0,987	0,980	0,974	0,968	0,961	0,955	0,797	0,793	0,788	0,784
4	1	0,997	0,994	0,991	0,989	0,986	0,983	0,980	0,901	0,899	0,896	0,894
5	1	0,999	0,998	0,997	0,995	0,994	0,993	0,992	0,959	0,958	0,957	0,956
6	1	1	0,999	0,999	0,998	0,998	0,997	0,997	0,984	0,984	0,983	0,983
7	1	1	1	1	0,999	0,999	0,999	0,999	0,994	0,994	0,994	0,994

Tabela 5 - Os valores de R_d para Δt de 63-67 e de 361-365.

	Δt em dias									
n	63	64	65	66	67	361	362	363	364	365
0	0,388	0,385	0,381	0,377	0,374	0,100	0,100	0,099	0,099	0,099
1	0,442	0,438	0,434	0,430	0,426	0,121	0,121	0,121	0,120	0,120
2	0,548	0,545	0,541	0,537	0,533	0,175	0,174	0,174	0,174	0,173
3	0,703	0,700	0,696	0,693	0,690	0,292	0,292	0,291	0,291	0,290
4	0,846	0,844	0,842	0,840	0,838	0,490	0,489	0,488	0,488	0,487
5	0,933	0,932	0,931	0,930	0,929	0,710	0,709	0,709	0,708	0,707
6	0,974	0,973	0,973	0,973	0,972	0,866	0,866	0,866	0,865	0,865
7	0,990	0,990	0,990	0,990	0,989	0,946	0,946	0,946	0,945	0,945

Após a análise de R_d, definimos $K_a(d, a, t)$ para representar a medida de conhecimento do desenvolvedor *d* sobre um artefato *a* em uma data-hora *t*. O modelo K_a é formado por dois fatores, onde o primeiro calcula a interação normalizada (II_{da}) para cada interação I_{da} produzida até *t*. O segundo fator introduz o esquecimento das interações (R_d) registradas no primeiro fator. Dessa forma, o conhecimento do desenvolvedor sobre um artefato é diluído no tempo, de modo que as interações mais recentes contribuem mais do que as interações mais distantes no tempo. É razoável pensar assim, pois os desenvolvedores tendem a esquecer os artefatos que não foram manipulados por eles recentemente (FRITZ *et al.*, 2007, 2010, MAALEJ, 2010).

A Equação 10 formaliza $K_a(d, a, t)$ como a soma do produto de II_{da} por R_d para todas as interações I_{da} produzidas pelo desenvolvedor *d* sobre o artefato *a* até a data-hora *t*. O Δt_i calcula o número de dias que se passaram desde o registro da interação do desenvolvedor (f_i) até *t*, onde *t* é a data-hora na qual o conhecimento do desenvolvedor deseja ser estimado. De acordo com K_a, quanto mais interações do tipo I_{da} um desenvolvedor tiver sobre um artefato, mais conhecimento ele poderá ter sobre ele. Isso é verdade porque II_{da} é um número sempre positivo.

$$K_a(d, a, t) = \sum_{f_i \leq t} \frac{I_{da}(d, a, f_i)}{max_{d \in D, \forall t_i \leq t}\, I_{da}(d, a, t_i)} \cdot \frac{1}{(G_d(n) \cdot\ b \cdot \Delta t_i) + 1} \qquad (10)$$

$$onde\ d \in D\ e\ \Delta t_i = t - f_i$$

3.2 Medida de Conhecimento do Desenvolvedor sobre uma Tarefa

Durante um projeto de desenvolvimento de software, os desenvolvedores criam e alteram artefatos por meio das interações I_{da}. Além disso, os desenvolvedores interagem com

outros desenvolvedores com o objetivo de realizar tarefas, as interações I_{dd}. O modelo K_s mede o conhecimento do desenvolvedor sobre uma tarefa fazendo uso dessas interações.

A Figura 15 ilustra os desenvolvedores que interagiram com os artefatos por meio de interações I_{da} para realizar uma tarefa. O desenvolvedor *d5* interagiu com o artefato *A1188* duas vezes: a primeira interação ocorreu no tempo 45 com grau de interação 15, e a segunda interação, no tempo 54, com grau 11. O desenvolvedor *d5* também interagiu com o artefato *A1475* nos tempos 60 e 80, com medidas do grau de interação iguais a 26 e 18, respectivamente. De acordo com dados do campo *attachments* do Bugzilla sobre o projeto Mylyn[16], mais de um desenvolvedor contribuiu com interações I_{da} para realizar tarefas, como é o caso das tarefas[17] 166406, 210686, 234065 e 248490. Então, há indícios de que a percepção do conhecimento sobre tarefas deve considerar as interações do tipo I_{da}.

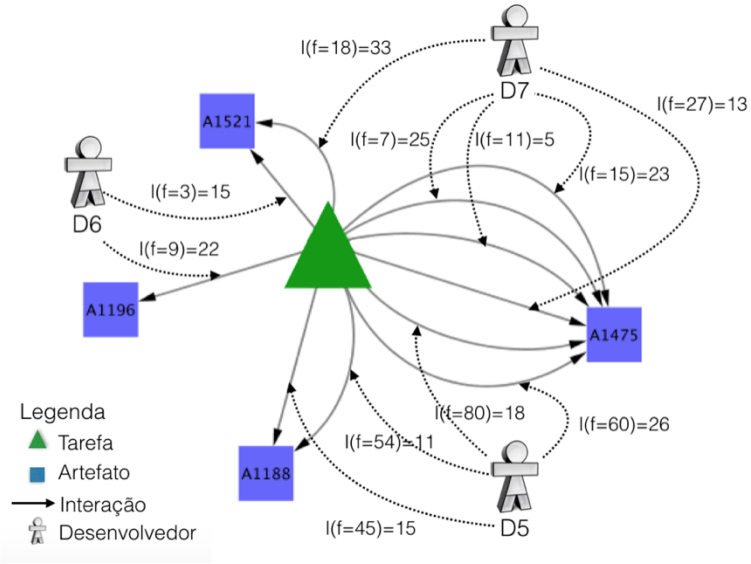

Figura 15 – Ilustração de interações Desenvolvedor-Artefato para realizar uma tarefa.

A Figura 16 ilustra as interações I_{dd} entre os desenvolvedores. A tarefa foi atribuída ao desenvolvedor *d14* e o desenvolvedor *d10* escreveu como a tarefa precisava ser feita. Esse caso mostra que o desenvolvedor *d10* tem conhecimento sobre essa tarefa, mas ele não produziu interação I_{da} sobre os artefatos relacionados com essa tarefa por

[16] www.eclipse.org/mylyn/ visitado em 24 de outubro de 2018.

[17] Por exemplo, para acessar o histórico de realização da tarefa 166406 use a URL https://bugs.eclipse.org/bugs/show_bug.cgi?id=166406

meio de interações I_{da}. Então, há indícios de que a percepção do conhecimento sobre tarefas também deve considerar as interações I_{dd}.

Nesse cenário, define-se a medida de conhecimento do desenvolvedor sobre uma tarefa com base na medida de seu conhecimento sobre os artefatos relacionados com essa tarefa por meio das interações I_{da}, mais a sua contribuição nas interações I_{dd} produzidas na realização da tarefa. A Equação 11 formaliza $K_s(d, s, t)$ como a soma do conhecimento do desenvolvedor *d* sobre todos os artefatos relacionados com a tarefa *s* ($\sum_{a \in s} K_a$), mais todas as interações I_{dd} produzidas pelo desenvolvedor *d* na execução da tarefa até o tempo *t*. O conhecimento do desenvolvedor sobre uma tarefa também sofre depreciação ao longo do tempo - a primeira depreciação foi herdada da Equação 10 (K_a), e a segunda parte de K_s usa a função de esquecimento R_d. Na função G_d, n é o número de dias, dentre os últimos sete dias, em que o desenvolvedor *d* expressou conhecimento sobre a tarefa *s* por meio de interação I_{dd}. Por exemplo, envio de uma mensagem a respeito da tarefa *s* para todos os desenvolvedores envolvidos na sua realização.

Figura 16 - Ilustração de interações Desenvolvedor-Desenvolvedor.

$$K_s(d,s,t) = \sum_{a \in s} K_a(d,a,t) + \sum_{f_i \leq t} II_{dd}(d_r, d_g, s, f_i) \cdot \frac{1}{(G_d(n) \cdot b \cdot \Delta t_i) + 1}, \quad (11)$$

$$onde\ d, d_r, d_g \in D\ e\ \Delta t_i = t - f_i$$

3.3 Medida de Conhecimento do Desenvolvedor sobre um Grupo de Tarefas Similares

O modelo $K_c(d,c,t)$ é definido como a medida de conhecimento do desenvolvedor *d* sobre o grupo de tarefas similares *c* na data-hora *t*. O K_c tem dois fatores: o primeiro é o $\frac{h}{m}$, onde h é o número de tarefas relacionadas ao grupo que tem a participação do desenvolvedor e m é o número de tarefas presentes no grupo. O fator $\frac{h}{m}$ quantifica o envolvimento do desenvolvedor com as tarefas do grupo (esse fator penaliza o desenvolvedor que tem conhecimento sobre poucas tarefas). No melhor caso, para h igual a m, o desenvolvedor tem conhecimento sobre todas as tarefas do grupo, $\frac{h}{m} = 1$. O segundo fator é a soma de $K_s(d,s,t)$, a medida de conhecimento do desenvolvedor sobre cada tarefa do grupo. A Equação 12 mostra $K_c(d,c,t)$ e a Tabela 6 mostra os valores de K_c, em grupos de até 10 tarefas, com K_s fixado em 1 para todas as tarefas.

$$K_c(d,c,t) = \frac{h}{m} \cdot \sum_{s \in c} K_s(d,s,t) \quad (12)$$

A Tabela 7 mostra um exemplo de como o fator $\frac{h}{m}$ pode influenciar no cálculo de K_c. Em um grupo com cinco tarefas similares, o desenvolvedor *d1* tem conhecimento sobre duas tarefas com $K_s = \{80,35\}$ e o desenvolvedor *d2* tem conhecimento sobre todas as cinco tarefas com $K_s = \{7,9,6,13,8\}$. Segundo o cálculo de K_c, considerando o fator, os desenvolvedores têm conhecimento com valores próximos: $K_c(d1) = 46$ e $K_c(d2) = 43$. No entanto, quando não se considera o fator $\frac{h}{m}$, o desenvolvedor *d1* apresenta um valor de K_c muito maior do que o de *d2*, mesmo não apresentando conhecimento sobre 3 tarefas em um grupo de 5 tarefas.

Tabela 6 - Os valores de K_c, em grupos de até 10 tarefas, com K_s fixado em 1 para todas as tarefas.

	m									
h	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	1	0,50	0,33	0,25	0,20	0,17	0,14	0,13	0,11	0,10
2		2	1,33	1	0,80	0,67	0,57	0,50	0,44	0,40
3			3	2,25	1,80	1,50	1,29	1,13	1	0,90
4				4	3,20	2,67	2,29	2	1,78	1,60
5					5	4,17	3,57	3,13	2,78	2,50
6						6	5,14	4,50	4	3,60
7							7	6,13	5,44	4,90
8								8	7,11	6,40
9									9	8,10
10										10

Tabela 7 – Exemplo da influência do fator $\frac{h}{m}$ no cálculo de K_c.

	K_s					K_c	K_c com $h = m = 1$
	1ª	2ª	3ª	4ª	5ª		
d1	80	35	0	0	0	46	115
d2	7	9	6	13	8	43	43

3.4 Medida de Conhecimento do Desenvolvedor sobre um Projeto

A contribuição dos desenvolvedores em projetos de desenvolvimento de software pode ser medida de diferentes formas: por produção de linhas de código, por pontos de função (SHEETZ *et al.*, 2009) e para projeto de software livre são utilizadas as medidas de tarefas executadas e número de *commits*[18]. Nesta tese, é definido o modelo K_p para medir o conhecimento dos desenvolvedores sobre um projeto de desenvolvimento de software, considerando a medida de conhecimento do desenvolvedor sobre os grupos de tarefas similares do projeto (K_c). De forma mais específica, o $K_p(d, p, t)$ representa a medida de

[18] https://projects.eclipse.org/projects/eclipse/who

conhecimento do desenvolvedor *d* sobre um projeto de software *p* em uma data-hora *t*, como visto na Equação 13.

$$K_p(d,p,t) = \frac{r}{q} \cdot \sum_{i=1}^{q} K_c(d,c_i,t) \quad (13)$$

A primeira parte da Equação 13, o fator $\frac{r}{q}$, determina a participação do desenvolvedor em grupos de tarefas similares, onde:

- r é o número de grupos que o desenvolvedor conhece do projeto de desenvolvimento de software e;
- q é o número total de grupos em que o projeto de desenvolvimento de software foi organizado.

Se o desenvolvedor participou de todos os grupos, $\frac{r}{q} = 1$. O fator penaliza o desenvolvedor que tem conhecimento sobre poucos grupos de tarefas similares. A influência do fator $\frac{h}{m}$ no cálculo de K_c, analisadas nas Tabelas 6 e 7, é igual à influência do fator $\frac{r}{q}$ no cálculo de K_p. A diferença é que no cálculo do K_c, o fator atua sobre K_s, enquanto que no cálculo do K_p, o fator atua sobre K_c.

A segunda parte da Equação 13 representa a soma do conhecimento do desenvolvedor sobre os grupos de tarefas similares. Na notação da Equação 13, a representação de quanto conhecimento o desenvolvedor "John" tem sobre o projeto "Mylyn Docs" em "2007-07-15" é $K_p(\text{"John"}, \text{"Mylyn Docs"}, \text{"2007} - 07 - 15\text{"})$.

3.5 Considerações Finais

Este capítulo estendeu o modelo de proveniência apresentado no Cap. 2 para definir explicitamente as interações Desenvolvedor-Artefato (I_{da}) e Desenvolvedor-Desenvolvedor (I_{dd}) presentes em projetos de desenvolvimento de software. Além dessas interações, também foi definida a similaridade de tarefas com base nas interações I_{da}. Em seguida, o capítulo formalizou matematicamente os quatro modelos para medir o conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software, levando em consideração explicitamente o esquecimento dos desenvolvedores.

O modelo para medir o conhecimento do desenvolvedor sobre um artefato (K_a) utiliza as interações I_{da}, considerando o esquecimento e a reaprendizagem dos

desenvolvedores. O esquecimento de uma interação I_{da} foi modelado considerando que um desenvolvedor esquece 50% de um arquivo de código fonte em 40 dias. Em consequência, a curva de esquecimento indicou que o desenvolvedor vai se lembrar de somente 10% da interação I_{da} depois de um ano.

O modelo para medir o conhecimento do desenvolvedor sobre uma tarefa (K_s) utiliza as interações I_{da} e I_{dd}. Esse modelo utiliza o modelo de conhecimento sobre um artefato (K_a) para calcular todo o conhecimento do desenvolvedor sobre os artefatos relacionados com a tarefa por meio de interação I_{da}. O modelo sobre uma tarefa também aplica esquecimento e reaprendizagem para todas as interações I_{dd} produzidas enquanto os desenvolvedores realizavam a tarefa.

O modelo para medir o conhecimento do desenvolvedor sobre um grupo de tarefas similares (K_c) calcula o somatório de todo o conhecimento do desenvolvedor sobre as tarefas do grupo utilizando o modelo de conhecimento sobre uma tarefa. A esse somatório de conhecimento, ainda é aplicado um fator para penalizar o desenvolvedor que tem conhecimento sobre poucas tarefas.

De forma análoga ao modelo que mede o conhecimento do desenvolvedor sobre um grupo de tarefas similares, o modelo para medir o conhecimento do desenvolvedor sobre todo o projeto de software (K_p) calcula o somatório de todo o conhecimento do desenvolvedor sobre os grupos de tarefas similares utilizando o modelo de conhecimento sobre um grupo de tarefas similares (K_c). A esse somatório de conhecimento, ainda é aplicado um fator para penalizar o desenvolvedor que tem conhecimento sobre poucos grupos.

Este capítulo, em resumo, atende ao objetivo específico desta tese para identificar especialistas em projetos de desenvolvimento de software: medir o conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software.

4- Método Tacin para Recomendar Contexto de Nova Tarefa

Este capítulo trata do objetivo desta tese: recomendar contexto para nova tarefa em projeto de desenvolvimento de software. O capítulo inicia com a análise visual dos dados de um projeto de desenvolvimento de software para mostrar o raciocínio utilizado na construção do método Tacin. Nas Seções 4.1, 4.2 e 4.3 são apresentados, em detalhe, os modelos para recomendar tarefas, artefatos e desenvolvedores para compor o contexto inicial de uma nova tarefa. O capítulo finaliza apresentando a proposta de uma representação do método Tacin.

O capítulo anterior apresentou os modelos do Tacin para perceber o conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software com base nas interações. Neste capítulo, é descrito o método Tacin para recomendar artefatos, tarefas similares e desenvolvedores para compor o contexto da nova tarefa, definido como:1- artefatos candidatos a serem editados pelos desenvolvedores na realização da nova tarefa; 2- desenvolvedores que têm potencial em apresentar conhecimento na nova tarefa; 3- tarefas realizadas similares à nova tarefa e; 4- especialistas nas tarefas similares à nova tarefa.

Primeiro será apresentado o raciocínio utilizado na concepção do método Tacin para recomendar o contexto inicial de novas tarefas em desenvolvimento de software. A Figura 17 mostra a visualização[19] do projeto Mylyn Docs[20], relacionando tarefas a artefatos, em função das interações de edição dos desenvolvedores sobre os artefatos enquanto realizavam tarefas (interação Desenvolvedor-Artefato definida no Capítulo 3). O projeto Mylyn Docs em 05/04/2017 apresentava 334 tarefas, 1538 artefatos e 49.906 interações Desenvolvedor-Artefato. As medidas do grau da interação Desenvolvedor-Artefato foram produzidas pelo plugin Mylyn no ambiente de desenvolvimento do Eclipse. Por exemplo, a relação tarefa$_X$-artefato$_Y$ é feita caso um desenvolvedor realize uma edição no artefato Y ao realizar a tarefa X, e será denominada visualização tarefa-artefato (*task-artifact view*). Para facilitar a análise, a Figura 17 não exibe todas as associações, ou seja, uma linha de relação tarefa-artefato pode estar representando muitas associações. Na parte de cima da Figura 17, identificamos tarefas isoladas relacionadas a poucos artefatos. No centro da Figura 17, na forma circular, identificamos uma tarefa

[19] Visualizações obtidas com o Cytoscape 3.6.0 (https://cytoscape.org/).

[20] https://projects.eclipse.org/projects/mylyn.docs visitado em 24 de outubro de 2018.

relacionada com muitos artefatos que não foram utilizados na realização de outras tarefas. E à esquerda, existe um conjunto com muitas tarefas que editaram artefatos criados por outras tarefas. Então, no projeto Mylyn Docs, novas tarefas após a sua realização irão se apresentar como tarefas isoladas ou irão se associar a outras tarefas segundo a visualização tarefa-artefato.

Por outro lado, a realização de uma nova tarefa produz artefatos novos. Por isso o software cresce em tamanho e complexidade ao longo do tempo. Para atividades intensivas em conhecimento, como é o caso do desenvolvimento de software (DI CICCIO *et al.*, 2015). A previsão de artefatos que serão criados ainda é um problema sem solução e não será mitigado nesta tese.

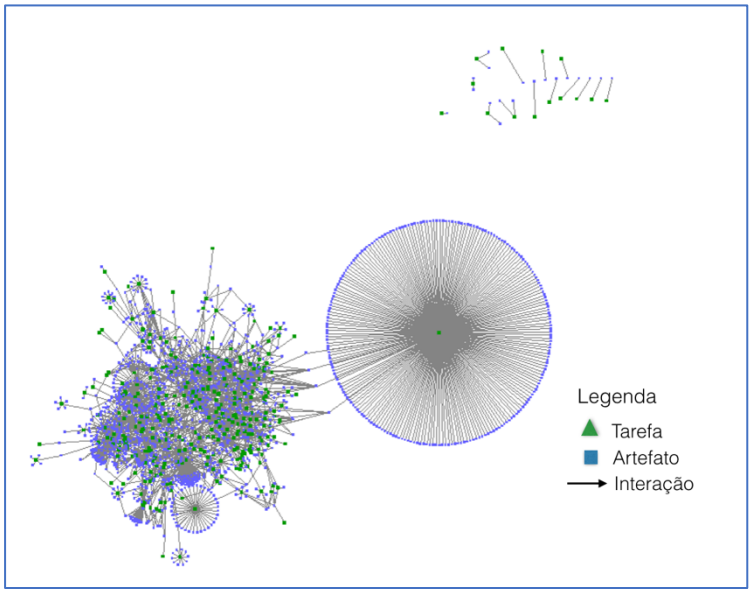

Figura 17 - Visualização do projeto Mylyn Docs em 05/04/2017.

Com o objetivo de entender melhor como as tarefas estão relacionadas por interação Desenvolvedor-Artefato no projeto Mylyn Docs, a Figura 18 exibe somente as primeiras 30 tarefas do projeto. Na parte de baixo da Figura 18, há a presença de tarefas isoladas: duas tarefas estão completamente isoladas e duas possuem um artefato em comum, mas estão isoladas das demais. Ao centro estão as tarefas que possuem artefatos em comum. As tarefas mais próximas possuem associações com os mesmos artefatos. Nesse cenário, para o Tacin, duas tarefas são semelhantes por interação se os desenvolvedores editaram pelo menos um artefato em comum enquanto realizavam as tarefas. Além disso, quanto mais artefatos em comum as tarefas tiverem, mais similares serão.

Uma nova tarefa, após o seu término, necessariamente fará parte do histórico de tarefas similares por interação, se apresentando, na maior parte das vezes, com alguma similaridade com as demais tarefas. O desafio é, portanto, antes do seu início, identificar, marcar, circular a provável região em que a nova tarefa irá se associar às demais tarefas após o seu término.

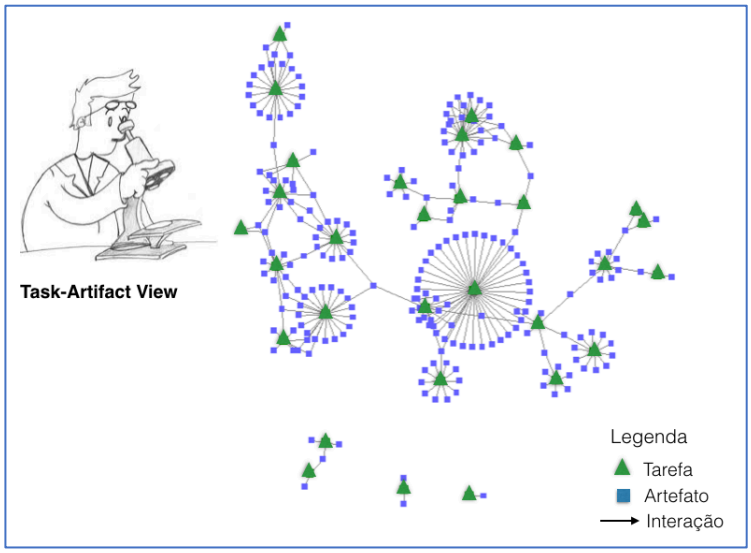

Figura 18 - Visualização das 30 primeiras tarefas do projeto Mylyn Docs.

O Tacin explora a possibilidade de circular as regiões similares por interação presentes no histórico de tarefas. Inicialmente, as regiões foram escolhidas visualmente levando em consideração a definição de similaridade de tarefas por interação. A Figura 19 mostra essas regiões, denominado de Cluster Objetivo. O Cluster Objetivo foi criado seguindo os passos: 1- Um grupo para cada tarefa isolada; 2 – Manter tarefas próximas visualmente no mesmo grupo; 3- Manter tarefas distantes em grupos diferentes.

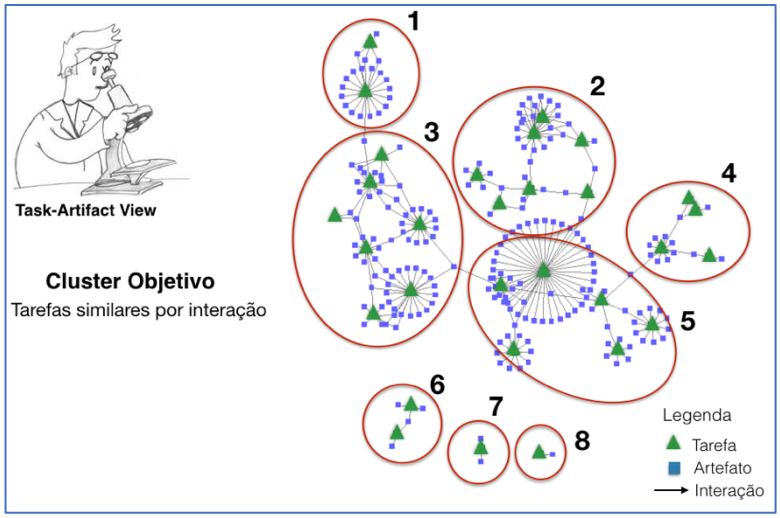

Figura 19 - Determinação do Cluster Objetivo de acordo com a análise visual das 30 primeiras tarefas do projeto Mylyn Docs.

Com a definição de Cluster Objetivo, a nova tarefa necessariamente fará parte de um grupo já existente ou irá, isoladamente, fazer parte de um novo grupo. Portanto, é aceitável pensar em recomendar um ou mais grupos para compor o contexto de uma nova tarefa. Para isso, é conveniente automatizar a construção do Cluster Objetivo e é preciso determinar como associar uma nova tarefa a um ou mais grupos do Cluster Objetivo.

A determinação de grupos similares automaticamente a partir de um conjunto de dados é amplamente estudada na literatura (CRUZ, 2010). No trabalho desta pesquisa de tese, foi utilizada uma heurística baseada na meta-heurística de busca em vizinhança grande para o problema de *clusterização* de módulos de software (MONÇORES *et al.*, 2018). Monçores agrupa os elementos de acordo com os pesos das associações com base em uma função que mede a qualidade da clusterização de módulos de software - do original em inglês, *modularization quality* (MQ). A função MQ está descrita na Seção 2.3. A Figura 20 mostra o resultado do agrupamento das tarefas similares por interação para as primeiras 30 tarefas do projeto Mylyn Docs. Tarefas com cores iguais estão nos mesmos grupos, a aresta entre duas tarefas indica que há similaridade entre elas, e o peso desta similaridade está indicado na aresta. Por exemplo, o Cluster 6, acima na Figura 20, expressa que a tarefas T245476 e 240743 têm similaridade por interação com peso 2, isto é, dois artefatos em comum foram editados pelos desenvolvedores enquanto realizavam essas tarefas. Esta visualização será denominada tarefa-tarefa (*task-task view*).

Agora podemos comparar o Cluster Objetivo exibido na

Figura 19 com os *clusters* gerados pelo LNS de Monçores *et al.* (2018), exibido na Figura 20. Na parte de baixo do Cluster Objetivo, as duas tarefas isoladas, grupos 7 e 8 não aparecem nos *clusters* gerados. As duas tarefas relacionadas por meio de um único artefato, Grupo 6 no Cluster Objetivo, aparecem no Grupo 11 da Figura 20. O Grupo 1, mais acima no Cluster Objetivo, corresponde ao Grupo 5, localizado mais à direita nos *clusters* gerados automaticamente. Os *clusters* também apresentam diferenças, os grupos 2, 3, 4 e 5 do Cluster Objetivo não possuem correspondência direta com os *clusters* gerados automaticamente. Cabe ressaltar que o Cluster Objetivo foi gerado por uma análise visual diretamente influenciada pela capacidade de observação do autor. Já os *clusters* gerados automaticamente fizeram uso da função MQ amplamente utilizada para medir a qualidade de clusterização de elementos.

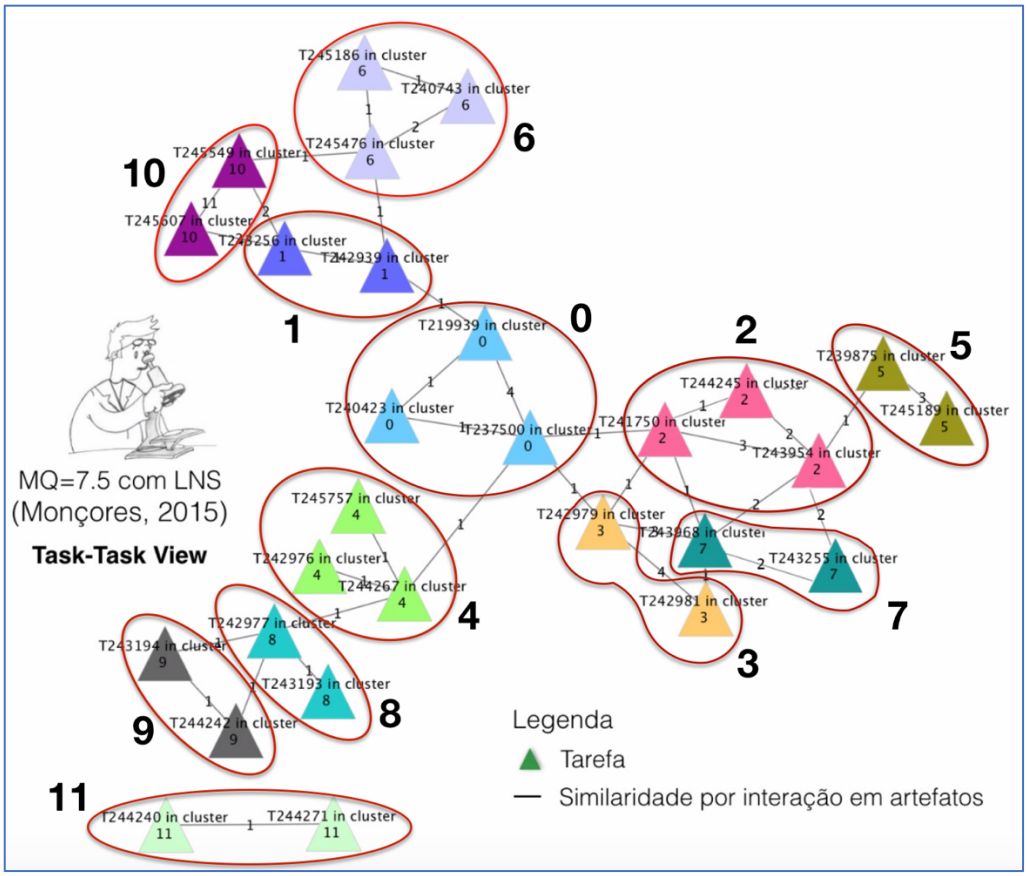

Figura 20 - Determinação do Cluster Objetivo de acordo com o algoritmo LNS (MONÇORES *et al.*, 2018) sobre as 30 primeiras tarefas do projeto Mylyn Docs (as 2 tarefas isoladas não aparecem na figura).

Uma vez alcançado o objetivo de organizar o histórico de elementos do projeto em grupos de tarefas similares por interação Desenvolvedor-Artefato, é preciso ainda determinar como associar uma nova tarefa a esses grupos. Uma nova tarefa, antes do seu início, não possui interação Desenvolvedor-Artefato. Logo, não é possível descobrir a similaridade por interação entre a nova tarefa e os grupos de tarefas similares diretamente. Cabe uma reflexão sobre a origem da informação. As interações Desenvolvedor-Artefato são registradas em função das ações de edição dos desenvolvedores sobre os artefatos com o objetivo de realizar tarefas. As tarefas, em sua grande maioria, são expressas de forma textual, caso do projeto Mylyn Docs. Por exemplo, na tarefa 245759, o item "Descrição Curta" expressa em linguagem natural (inglês) o erro detectado, e o item "Descrição" detalha o erro indicando o artefato que gerou o erro, como apresentado na Tabela 8. O desenvolvedor, guiado por essas informações textuais, produz edições nos artefatos para corrigir o erro. Então, por isso, o método Tacin associa a nova tarefa aos grupos através da similaridade textual.

Tabela 8 – Descrição curta e descrição da tarefa 245759 do projeto Mylyn Docs.

	TaskId 245759[21]
Descrição curta	cannot run the HtmlViewer or MarkupViewer in a stand-alone GUI
Descrição	the HtmlViewer and MarkupViewer should be usable outside of an Eclipse environment (ie: in a stand-alone SWT application) This code worked in Textile-J. Try running java class org.eclipse.mylyn.wikitext.textile.core.Main

A Figura 21, por sua vez, ilustra a associação de uma nova tarefa com os grupos presentes no Cluster Objetivo. Cada grupo pode ser visto como uma única tarefa onde a sua descrição curta é definida como a concatenação das descrições curtas de todas as tarefas pertencentes ao grupo. Da mesma forma, a descrição do grupo é definida como a concatenação de todas as descrições das tarefas do grupo. Assim, é possível calcular a similaridade textual entre a nova tarefa e um grupo de tarefas similares por interação.

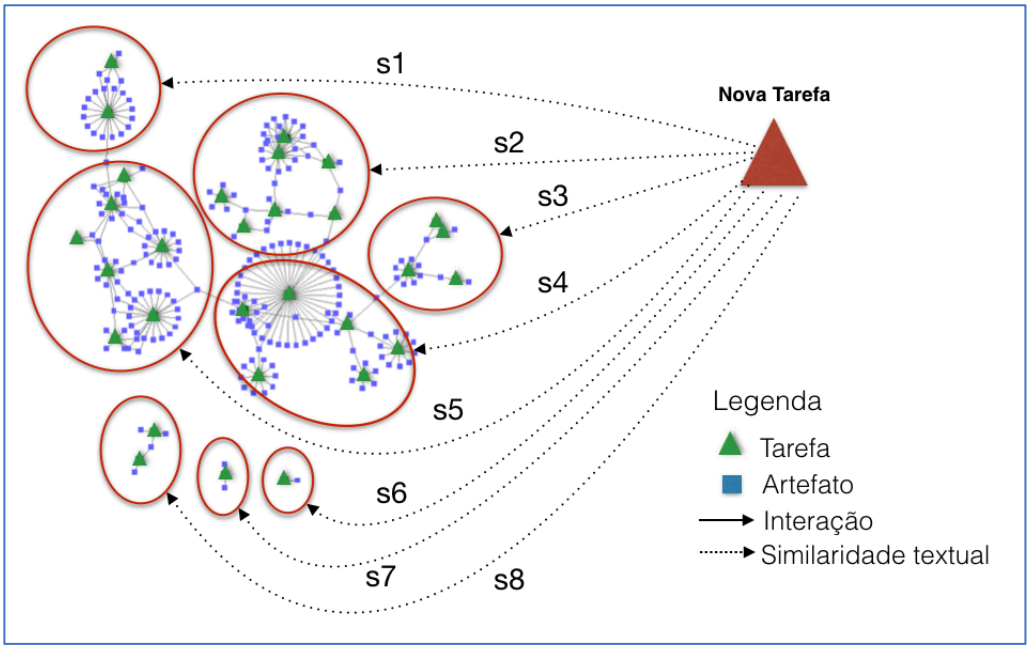

Figura 21 – Ilustração da similaridade textual entre a nova tarefa e o cluster de tarefas similares por interação.

A seguir, nas Seções 4.1, 4.2 e 4.3, são apresentados em detalhes o método Tacin para recomendar tarefas similares, artefatos e desenvolvedores para compor o contexto inicial de uma nova tarefa. E na Seção 4.4, o Método é representado em uma linguagem com notação visual.

[21] Disponível em https://bugs.eclipse.org/bugs/show_bug.cgi?id=245759

4.1 Recomendação de Tarefas Similares

No escopo desta tese, a descrição da nova tarefa indica o que deve ser feito utilizando linguagem natural (Alemão, Inglês, Português etc.). Já o histórico de elementos do projeto é organizado em grupos de tarefas semelhantes por interação Desenvolvedor-Artefato. Diferente de outros trabalhos (ČUBRANIĆ *et al.*, 2005a; ROCHA *et al.*, 2015), Tacin combina similaridade textual com similaridade por interação para inferir quais tarefas do histórico de tarefas do projeto são similares à nova tarefa. Essa combinação de texto com dados tem respaldo nos estudos de MAALEJ *et al.* (2017). Os estudos indicaram que as informações de texto e dos dados de contexto podem ser complementares para identificar relacionamentos entre tarefas.

Ao longo do projeto de desenvolvimento do software, tarefas são realizadas e as interações dos desenvolvedores sobre os artefatos são registradas. Por isso, a organização das tarefas do projeto em grupos similares por interação varia também no tempo. Por definição, cada tarefa do histórico de tarefas pertence a somente um grupo de tarefas semelhantes por vez, mas uma tarefa de um grupo pode apresentar similaridade por interação com tarefas de outros grupos.

A Figura 22 ilustra a recomendação de tarefas similares à nova tarefa. Os grupos 1, 2 e 3 representam os grupos com os maiores valores de similaridade textual com a nova tarefa. O método Tacin recomenda para o desenvolvedor que irá realizar a nova tarefa todas as tarefas dos grupos selecionados (um, dois ou três), informando os valores da similaridade textual dos grupos com a nova tarefa. Cabe ao desenvolvedor a opção por quais tarefas similares considerar.

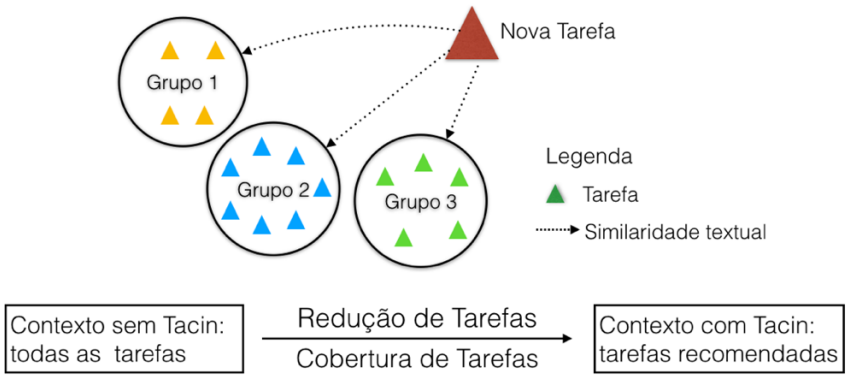

Figura 22 – Ilustração do método para recomendar tarefas para compor o contexto de uma nova tarefa em desenvolvimento de software.

A expectativa é que o número de tarefas recomendadas seja um número consideravelmente menor em relação ao número de tarefas do projeto. E também, ao

término da nova tarefa, que ela apresente similaridade por interação com os grupos de tarefas recomendados. A seleção de grupos para recomendação será detalhada a seguir.

4.2 Recomendação de Artefatos

Tacin recomenda os artefatos relacionados por interação Desenvolvedor-Artefato às tarefas dos grupos selecionados para compor o contexto inicial de uma nova tarefa. Em geral, analisando o histórico do projeto organizado em tarefas semelhantes por interação Desenvolvedor-Artefato, é esperado, no melhor caso, que a quanto mais tarefas um artefato estiver relacionado, mais chance terá de pertencer ao contexto de uma nova tarefa. Por outro lado, no pior caso, um artefato relacionado com uma tarefa isolada terá menos chance de ser editado por uma nova tarefa. Assim, a recomendação de um artefato específico pode acontecer através da seleção de muitos grupos, de acordo com o melhor caso. Por exemplo, na linguagem Java, sobre os artefatos responsáveis pela internacionalização do software, um artefato é criado para cada linguagem natural (MessagesBundle_en_US.properties, etc.). Nessa situação, todas as tarefas que precisarem incluir ou alterar uma informação para o usuário, precisarão editar esses artefatos. Já na análise do pior caso, o artefato relacionado com somente uma tarefa, necessariamente no Tacin, só será recomendado através de um grupo que tenha somente uma tarefa, configurando também o pior caso de recomendação.

Tacin é avaliado pela média harmônica entre Redução e Cobertura para a recomendação de artefatos (as métricas utilizadas na avaliação são apresentadas na Seção 2.2). Redução é o percentual de redução entre o total de artefatos disponíveis no projeto e os artefatos recomendados (Equação 14). Cobertura é largamente utilizada em sistemas de recomendação para indicar o percentual de acerto entre os artefatos recomendados (Equação 15), sendo que para a avaliação do Tacin, $Artefatos\ Relevantes$ são os artefatos que foram alterados pelos desenvolvedores na realização da nova tarefa. A opção por média harmônica foi inspirada na medida *F-measure*, média harmônica entre Precisão e Cobertura (AVAZPOUR *et al.*, 2014).

$$Redução = 1 - \frac{|Artefatos\ Recomendados|}{|ArtefatosDoProjeto|} \tag{14}$$

$$Cobertura = \frac{|Artefatos\ Recomendados \cap\ Artefatos\ Relevantes|}{|Artefatos\ Relevantes|} \tag{15}$$

A Figura 23 ilustra a recomendação de artefatos com o Tacin. Como na recomendação de tarefas similares, os três grupos que apresentarem mais semelhança textual com a nova tarefa são selecionados para a recomendação. O Tacin realiza avaliação da recomendação de artefatos para cada nova tarefa considerando três opções: 1 – somente o grupo mais similar textualmente à nova tarefa; 2 – os dois grupos mais similares textualmente à nova tarefa; 3 – e os três grupos mais similares textualmente à nova tarefa. Para cada nova tarefa, calcula-se a média harmônica entre Redução e Cobertura para as 3 opções, sendo que a opção que tiver a maior média recebe um ponto. O Tacin, por sua vez, recomenda contexto utilizando a opção com maior *Rc-measure*, e caso haja empate, recomenda a opção de menor número de grupos selecionados.

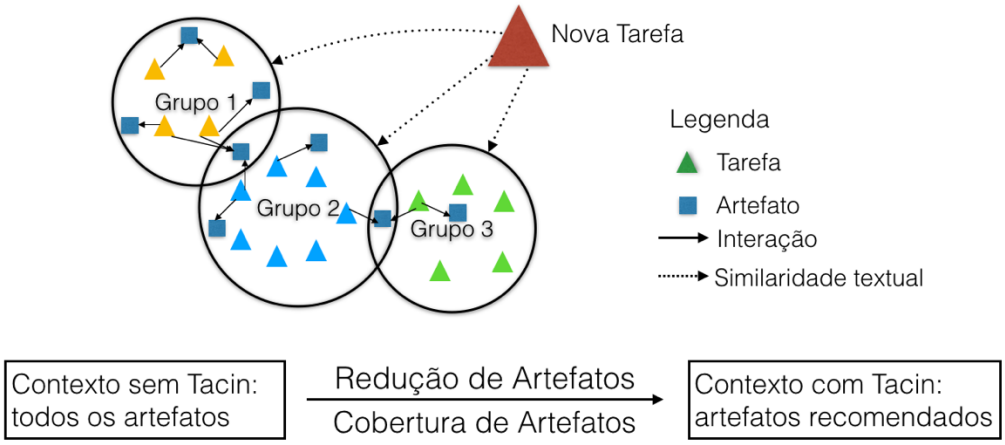

Figura 23 – Ilustração do método para recomendar artefatos para compor o contexto de uma nova tarefa em projeto de desenvolvimento de software.

4.3 Recomendação de Desenvolvedores

No Tacin, os desenvolvedores recomendados são aqueles que possuem conhecimento nos grupos selecionados com suas respectivas tarefas e artefatos. Os modelos de percepção do conhecimento apresentados no Capítulo 3 são aplicados nos grupos, tarefas e artefatos recomendados. A Figura 24 ilustra como o Tacin recomenda desenvolvedores no início de uma nova tarefa. Para todos os grupos de tarefas similares recomendados, como explicado na Seção 4.2, um grupo, os dois ou os três que apresentarem os maiores valores de similaridade textual com a nova tarefa, é gerada uma lista de desenvolvedores que possuem conhecimento de acordo com o modelo K_c apresentado na Seção 3.3. De forma parecida, para cada tarefa recomendada é gerada uma lista com os desenvolvedores que possuem conhecimento segundo o modelo K_s apresentado na Seção 3.2. O mesmo acontece para os artefatos recomendados, de acordo com o K_a (Seção 3.1).

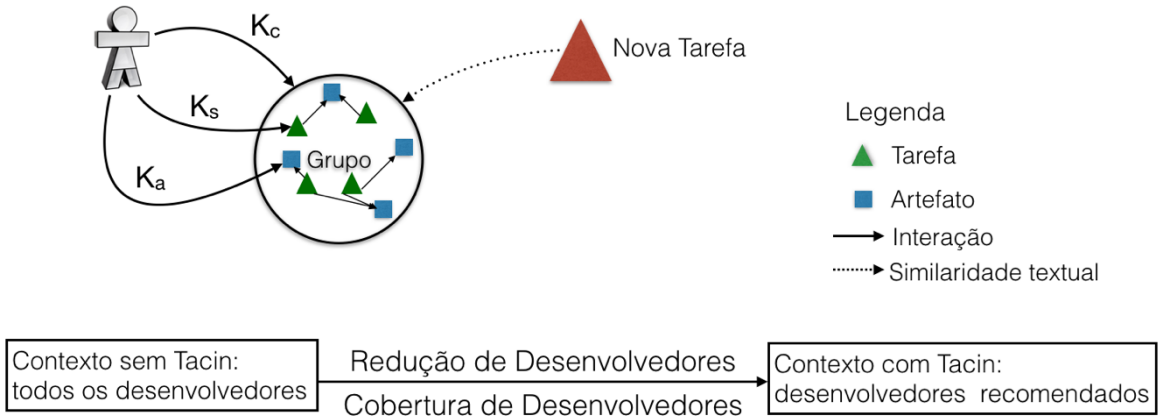

Figura 24 – Ilustração dos modelos para recomendar desenvolvedores para compor o contexto de uma nova tarefa em projeto de desenvolvimento de software.

4.4 Representação do Tacin

O método Tacin advoga que a recomendação de artefatos juntamente com a recomendação de tarefas similares e especialistas, com base nas interações dos desenvolvedores, pode contribuir para a formação do contexto inicial de novas tarefas. Com relação aos artefatos, o método busca recomendar um número de artefatos muito menor que o total de artefatos do projeto, mas sempre com a intenção de acertar os artefatos relevantes para a realização da nova tarefa.

A Figura 25 representa o Tacin utilizando a notação para representação de processos de negócio BPMN (Business Process Model and Notation) (BPMN, 2018). A representação do Tacin em BPMN descreve os subprocessos *Coletar Dados do Projeto*, *Construir Recomendação*, *Entregar Recomendação* e *Avaliar Recomendação* observando as atividades para a construção de um SRES (PROKSCH *et al.*, 2015). O Subprocesso *Coletar Dados do Projeto* é responsável por registrar as informações textuais (descrição curta e a descrição) das novas tarefas e das interações dos desenvolvedores (Desenvolvedor-Artefato e Desenvolvedor-Desenvolvedor).

O Subprocesso *Construir Recomendação* descreve as atividades para a produção da recomendação de artefatos, tarefas similares e seus respectivos desenvolvedores que apresentam mais conhecimento para compor o contexto de uma nova tarefa. A construção da recomendação inicia quando o desenvolvedor ativa uma nova tarefa, indicando que o

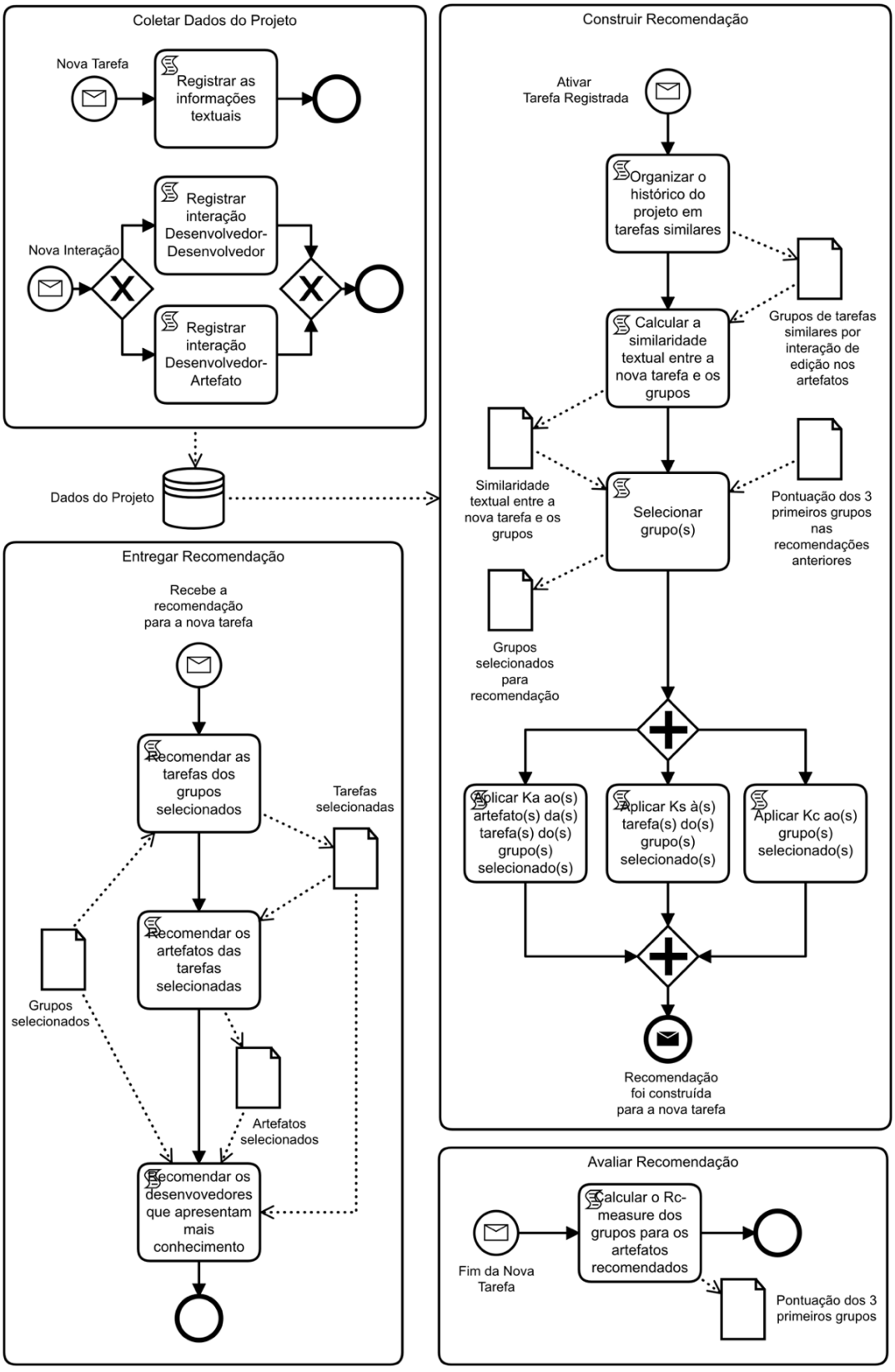

Figura 25 - Representação do Tacin em BPMN.

desenvolvedor tem a intenção de iniciar a realização da nova tarefa. A primeira atividade organiza o histórico do projeto em tarefas similares por interação em artefatos utilizando

a técnica de Clusterização. A segunda atividade calcula a similaridade textual entre a nova tarefa e os grupos de tarefas similares por interação em artefatos utilizando a técnica de Processamento de Linguagem Natural, sendo que a parte textual de um grupo é formada pela concatenação dos textos de todas as tarefas do grupo. A terceira atividade seleciona os grupos para recomendação, observando a atual pontuação dos grupos de acordo com as recomendações anteriores, resultado do processo *Avaliar Recomendação*. Em seguida, os modelos K_a, K_s e K_c são aplicados para identificar quais desenvolvedores apresentam mais conhecimento nos itens considerados como valiosos para a recomendação: artefato(s), tarefa(s) e grupo(s).

O Subprocesso *Entregar Recomendação* disponibiliza os itens selecionados como valiosos para compor o contexto inicial da nova tarefa informando também quem atualmente apresenta conhecimento sobre eles. Um grupo é disponibilizado com a informação do valor da similaridade textual com a nova tarefa. Uma tarefa é disponibilizada com a descrição curta e o valor da similaridade textual com a nova tarefa juntamente com os seus artefatos relacionados por interação Desenvolvedor-Artefato. Além disso, também é disponibilizada a informação das medidas de conhecimento dos desenvolvedores sobre os itens recomendados utilizando os modelos K_a, K_s e K_c.

E, finalmente, o Subprocesso *Avaliar Recomendação* calcula a média harmônica entre Redução e Cobertura (Rc-measure) para os artefatos recomendados nos três casos previstos: recomendação de um grupo, dois ou três grupos. O caso com o maior Rc-measure recebe 1 ponto. O Tacin recomendará de acordo com o caso que possuir maior pontuação nas próximas recomendações.

4.5 Considerações Finais

O presente capítulo apresentou o método Tacin para recomendar tarefas, artefatos e desenvolvedores para compor o contexto inicial de uma nova tarefa.

Primeiro, foi realizada uma análise visual das tarefas relacionadas pelas interações Desenvolvedor-Artefato de um projeto de desenvolvimento de software. O resultado dessa análise identificou: tarefas isoladas relacionadas a poucos artefatos; uma tarefa relacionada com muitos artefatos que não foram utilizados na realização de outras tarefas e; um conjunto com muitas tarefas que estavam relacionadas com outras tarefas. Esse resultado indicou a possibilidade de o Tacin organizar o histórico de elementos do projeto em grupos de tarefas similares por interação Desenvolvedor-Artefato.

Depois, foi realizada a associação da nova tarefa por similaridade textual com os grupos de tarefas similares utilizando a descrição textual da nova tarefa com a concatenação de todas as descrições das tarefas de cada grupo. O Tacin seleciona os grupos mais similares textualmente à nova tarefa de acordo com a pontuação das recomendações já realizadas: o grupo mais similar; ou os dois grupos mais similares; ou ainda, os três grupos mais similares textualmente à nova tarefa. O Tacin recomenda as tarefas e os artefatos desses grupos, listando os desenvolvedores que mais conhecimento têm sobre esses elementos de acordo com os modelos apresentados no Capítulo 3.

Portanto, este capítulo atende ao objetivo desta tese: recomendar contexto para uma nova tarefa em projeto de desenvolvimento de software.

5 – Avaliação

Este capítulo investiga a questão de pesquisa desta tese: ***QP-*** *A organização do histórico de elementos do projeto, em função da similaridade por interação dos desenvolvedores sobre artefatos, combinada com a similaridade textual, é suficiente para produzir recomendação de contexto de nova tarefa em projeto de desenvolvimento de software? O capítulo inicia descrevendo como foi realizada a coleta de dados. Segue com as avaliações dos Modelos de Percepção do Conhecimento (Seção 5.2), e da avaliação do Modelo para Recomendar Contexto no Início das Tarefas (Seção 5.3). Ambas as avaliações utilizam os dados coletados conforme descrito na Seção 5.1. Cada avaliação descreve os materiais e métodos utilizados, apresenta a análise dos resultados e descreve as ameaças à validade.*

5.1 Coleta de Dados

LETHBRIDGE *et al.* (2005) apresentaram uma taxonomia para as técnicas de coleta de dados com base no grau de contato humano. De acordo com essa taxonomia, a técnica de coleta de dados utilizada nesta tese é classificada em terceiro grau porque requer acesso apenas a artefatos de trabalho usando a técnica de Análise de Logs de Ferramentas. Os dados foram coletados do projeto Mylyn Docs, hospedado em https://www.eclipse.org/, e dos projetos Homebrew/homebrew-cask-drivers, Homebrew/homebrew-formula-analytics, brewsci/homebrew-science e Kerl/kerl, hospedados em https://github.com.

O projeto Mylyn Docs[22] é um subprojeto do projeto Mylyn[23], um plugin do Eclipse IDE, que reduz o número de artefatos disponíveis para facilitar a navegação em artefatos relevantes para a realização de uma determinada tarefa. Os dados coletados foram armazenados pelas ferramentas Git, Bugzilla e Mylyn. A Figura 26 ilustra as três perspectivas de observação: a primeira coleta os dados do projeto Mylyn Docs para gerar a linha de base do primeiro estudo; a segunda coleta os dados das interações do desenvolvedor sobre os artefatos registrados pelo plug-in Mylyn; e a terceira perspectiva foi capturada pela ferramenta Git, que registra as submissões de artefatos ao repositório do projeto (*commits*). Em seguida, a coleta de dados é descrita de acordo com cada perspectiva apresentada.

---

[22] http://projects.eclipse.org/projects/mylyn.docs

[23] http://www.eclipse.org/

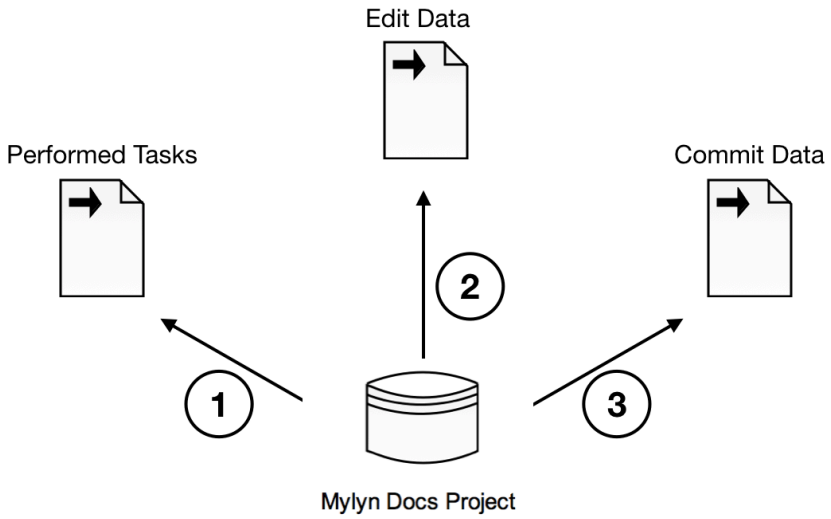

Figura 26 - Perspectivas de observação da coleta de dados.

A primeira perspectiva, Tarefas Realizadas (*Performed Tasks*), foi gerada a partir da consulta aos dados do Mylyn Docs gravados pelo Eclipse Bugzilla[24]. Os parâmetros para a consulta foram:

> Classification = Mylyn e Product = "Mylyn Docs" e Component = EPUB ou Framework ou HtmlText ou Wikitext e Status = RESOLVED e Resolution = FIXED.

A consulta expressa que as tarefas marcadas como resolvidas (RESOLVED) e testadas (FIXED) pelos desenvolvedores nos componentes EPUB, Framework, HtmlText e Wikitext do projeto Mylyn Docs pertencente ao Mylyn, serão retornadas na sua execução. A consulta foi executada em 18 de setembro de 2017 e gerou um arquivo XML a partir do qual foram extraídos os conteúdos de *bug_id*, *assigned_to* e *endtask* de 609 tarefas realizadas (correções de erros + novas funcionalidades).

A segunda perspectiva de observação, Dados de Edição (*Edit Data*), foi produzida a partir do log gerado pela ferramenta Mylyn sobre a realização das tarefas, disponíveis nos arquivos anexados no Eclipse Bugzilla. Os parâmetros para a consulta foram os parâmetros descritos na consulta da primeira perspectiva mais:

[24] https://bugs.eclipse.org/bugs/query.cgi

e Match ALL of the following separately→Attachment Description→contains the string→mylyn/context/zip.

A consulta foi executada em 05 de abril de 2017 e retornou 345 tarefas. Depois, todas as interações de edição (Kind = "edit") foram extraídas dos arquivos anexados (mylyn/context/zip) nas tarefas. Por exemplo, o Código 1 ilustra um log de eventos registrado pelo Mylyn para a tarefa 219939. A partir desse evento foi extraído que o desenvolvedor associado a essa tarefa executou uma interação de edição iniciada em "2008-07-16 19:01:36.15 -04" e terminada em "2008-07-16 19:34:54.312 -04" sobre o artefato /caminho/TaskEditorRichTextPart.java, com um grau de interesse igual a 82. A extração resultou em dados de 334 tarefas com 49906 edições de interação realizadas por 6 desenvolvedores em mais de 1538 artefatos. O fuso horário do PDT é igual a UTC-7. Portanto, o grau de interesse tipo edição, registrado pelo Mylyn, foi utilizado para ser o grau de interação Desenvolvedor-Artefato.

Código 1- O trecho de código XML registrado pelo Mylyn.

```
<InteractionEvent Delta="null"
  EndDate="2008−07−16 16:34:54.312 PDT"
  Interest="82.0" Kind="edit" Navigation="null"
  OriginId="org.eclipse.jdt.ui.CompilationUnitEditor"
  StartDate="2008−07−16 16:01:36.15 PDT"
  StructureHandle="/path/{TaskEditorRichTextPart.java [ TaskEditorRichTextPart\
texttildelowcreateControl \\texttildelowQComposite ; \texttildelowQFormToolkit ;"
StructureKind="java"/>
```

A terceira e última perspectiva de observação, Dados Submetidos ao Repositório de Código (*Commit Data*), coletou os dados registrados pelo Git no repositório de código do projeto Mylyn Docs. A Tabela 9 exibe os comandos 1-3 para extrair os *commits* até "05 de abril/2017" contendo *name*, *email*, *commit_date*, *commit_hash* e *commit_msg*. Os *commits* que identificaram a tarefa no início do *commit_msg* foram selecionados para registrar a interação de *commit*. O comando 4 foi usado para extrair os artefatos enviados pelo *commit*. O interesse de uma interação de consolidação foi definido como 1 e as datas de início e término da interação foram registradas com *commit_date*. Os nomes dos desenvolvedores cadastrados no Git, em alguns casos, eram diferentes daqueles cadastrados pelo Bugzilla. Adotou-se os nomes cadastrados no Bugzilla. A extração resultou em 918 *commits* realizados por 33 desenvolvedores sobre 5407 artefatos.

Tabela 9- Comandos para extrair *commits* de um repositório Git.

	Linha de Comando
1	git clone git://git.eclipse.org/gitroot/mylyn/org.eclipse.mylyn.docs.git
2	cd org.eclipse.mylyn.doc
3	git log --until=”05 Apr 2017” --format=”%an;%ae;%cl;%h;%s” > commits_mylynDocs.txt”
4	git diff-tree --no-commit-id --name-only -r *commit_hash*

Para os projetos hospedados no GitHub, os dados foram coletados de acordo com a primeira e terceira perspectivas de observação, utilizando uma biblioteca[25] de acesso aos projetos do GitHub. O Pseudocódigo 1 mostra os passos utilizados para realizar essa coleta. Dado que no GitHub, um *PullRequest* representa o conjunto de atualizações que foram necessárias para a realização de uma *Issue* (tarefa), a linha 2 retorna as informações sobre a realização de cada tarefa (*issue=closed*). E as linhas 4, 5 e 6 registram as informações de cada tarefa e das suas interações Desenvolvedor-Artefato e Desenvolvedor-Desenvolvedor. Para a interação Desenvolvedor-Artefato, o grau foi definido como o número de linhas inseridas e excluídas no artefato em cada *commit*.

```
1 -  for each gitHubProject{
2 -      pullRequestList = getPullRequests(GHIssueState.CLOSED);
3 -      for each pullRequest in pullRequestList{
4 -          insertTask(pullRequest.getIssue);
5 -          insertDeveloperArtifactInteraction(task, commits);
6 -          insertDeveloperDeveloperInteraction(task, comments);
7 -      }
8 -  }
```

Pseudocódigo 1 – Pseudocódigo para coletar dados de projetos hospedados no GitHub.

5.2 Avaliação dos Modelos Tacin de Percepção do Conhecimento

A avaliação dos modelos de percepção investiga a subquestão de pesquisa desta tese: **QP$_1$**- Os dados do histórico das interações, considerando a modelagem do esquecimento humano, são suficientes para produzir medições de conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software?

[25] http://github-api.kohsuke.org/

Dessa forma, a avaliação pode ser assim resumida usando o guia do GQM (Goals Questions Metrics) (BASILI, 1992):

> **Analisar** as interações de edição e de *commit* dos desenvolvedores em artefatos de software **com a finalidade de** inferir o conhecimento dos desenvolvedores **em relação a** um projeto de desenvolvimento de software **sob o ponto de vista** do gerente de projeto.

A questão geral de pesquisa Q1 para esta avaliação é definida como: Quem sabe sobre o projeto de desenvolvimento de software? A Q1 pode ser respondida usando quatro medidas. A primeira - número de tarefas realizadas - é uma medida bem aceita para medir a produção ou participação dos desenvolvedores em projetos de desenvolvimento de software (ZHOU e MOCKUS, 2010), muito utilizada em projetos de código aberto, por exemplo, aqueles sob a administração do Eclipse[26]. A segunda observa o número de interações de edição realizadas pelos desenvolvedores para realizar tarefas (consideramos neste estudo as interações de edição e *commit*, sendo o *commit* uma indicação de edição). A terceira medida é o número de grupos de tarefas semelhantes por interação nas quais o desenvolvedor tem conhecimento, segundo o modelo K_c (Seção 3.3). A quarta, baseada no modelo K_p, conforme ilustrado na Seção 3.4, infere o conhecimento do desenvolvedor sobre o projeto de desenvolvimento de software, considerando as tarefas similares por interação Desenvolvedor-Artefato. O K_p também utiliza o grau das interações, levando em consideração o esquecimento dos desenvolvedores. Para comparar as semelhanças entre as classificações de desenvolvedores produzidas por essas quatro medidas, definimos três questões de pesquisa: RQ1: A classificação de desenvolvedores por *tarefas realizadas* (R_{pt}) é semelhante à classificação por suas interações com os dados? - RQ2: A classificação R_{pt} é semelhante à classificação gerada pelo modelo K_c? - RQ3: A classificação R_{pt} é semelhante à classificação gerada pelo modelo K_p?

Além dessas questões de pesquisa, também é realizada uma análise de sensibilidade de K_p com relação à influência dos parâmetros R_d, G_d, $\frac{h}{m}$ e $\frac{r}{q}$.

[26] http:www .eclipse.org

5.2.1 Materiais e Métodos

O Pseudocódigo 2 descreve os passos de planejamento do experimento com o objetivo de gerar listas de classificação de desenvolvedores com conhecimento sobre o projeto Mylyn Docs. As listas são geradas de quatro maneiras diferentes. A classificação dos desenvolvedores por número de tarefas realizadas usa os dados da perspectiva de observação "Tarefas Realizadas" (*Performed Tasks*). A classificação dos desenvolvedores por número de interações de edição realizadas é construída a partir da observação dos dados de edição em artefatos (*Edit Data*) e submissão de artefatos para o repositório de código do Projeto (*Commit Data)*. A classificação K_c ordena os desenvolvedores pelo número de *clusters* nos quais eles apresentam conhecimento de acordo com o modelo K_c. A classificação K_p usa o modelo K_c. Ambos os modelos, K_c e K_p, usam as perspectivas "Dados de Edição" e "Dados Submetidos" ao Repositório de Código, como visto em detalhe na Seção 5.1.

```
1− independent_runs := 13;
2− date := 2015−05−01;
3 – for each independent_run {
4−    computePerformedTasks_ranking_baseLine(date );
5−    computeInteractions_ranking(date);
6−    buildClusters_similarTasksByInteractions(date);
7−    computeKc_ranking(date);
8−    computeKp_ranking(date);
9−    date := nextMonth(date);
10 − }
```

Pseudocódigo 2- O Pseudocódigo para gerar listas de classificação de desenvolvedores com conhecimento sobre o projeto Mylyn Docs.

A linha 6 do Pseudocódigo 2 deve gerar grupos de tarefas semelhantes por interação de edição em artefatos. Duas tarefas são consideradas similares por interação quando elas interagem em pelo menos um artefato em comum. O peso da similaridade foi definido como o número de artefatos em comum (definição descrita no Capítulo 3). O experimento utilizou o algoritmo para gerar *clusters* LNS (MONÇORES *et al.*, 2018) com heurística "Modularization Quality" (MANCORIDIS *et al.*, 1999). O Apêndice B descreve a Implementação de Referência dos Modelos Tacin para esse experimento. Para cada rodada do experimento, 10 rodadas do algoritmo LNS foram realizadas e o *cluster*

com o maior MQ foi escolhido para ser usado nos cálculos das classificações usando K_c e K_p. LNS e MQ foram vistos em detalhe na Seção 2.

O experimento foi realizado com data de referência para cada primeiro dia de mês, de 01/05/2015 até 01/05/2016, totalizando 13 rodadas. Cada rodada produziu quatro listas de classificação por ordem decrescente de pontuação dos desenvolvedores por: 1 - tarefas realizadas; 2 - número de interações de edição em artefatos; 3- usando o modelo K_c; e 4- usando o modelo K_p. As classificações dos desenvolvedores por interações de edição em artefatos, K_c e K_p são avaliadas de acordo com o grau de correlação com a classificação por tarefas realizadas usando o modelo de correlação Spearman (BEST e ROBERTS, 1975) com nível de confiança definido em 95%.

5.2.2 Resultados e Análises

A Tabela 10 mostra o resultado da primeira rodada do experimento referente a 01/05/2015. A classificação por tarefas realizadas ordena os 10 desenvolvedores que realizaram mais tarefas até 01/05/2015. O primeiro lugar realizou 378 tarefas e o décimo realizou 3. O primeiro lugar nesta classificação também foi o primeiro nas outras classificações (Interação, K_c e K_p). O desenvolvedor dev1039 foi classificado em décimo na classificação por tarefas realizadas, 22º na classificação por interações, 13º no K_c e 13º em K_p. Espera-se que, para realizar uma tarefa, o desenvolvedor realize uma ou mais interações, mas não foi o que aconteceu com os desenvolvedores dev1512 e dev1039. O dev1512 realizou seis tarefas e produziu uma interação de edição. O dev1039 realizou três tarefas e também produziu uma interação de edição. Concluímos que esses desenvolvedores não registraram todas as suas interações enquanto realizavam suas tarefas. No entanto, optou-se por manter essas informações na avaliação das semelhanças entre as classificações.

A única interação do desenvolvedor dev1512 aconteceu em 26/05/2009 com grau 1 no artefato 6135 que é parte somente da tarefa 260483. Essa interação aplicada ao modelo K_a resulta em um valor já bem próximo de zero (0,02). Consequentemente, em função desses dados, os modelos K_c e K_p mostram que o desenvolvedor não apresenta conhecimento sobre o projeto nessa rodada. Por outro lado, a única interação do desenvolvedor dev1039 foi registrada pelo Mylyn em 10/03/2015 com grau de interesse 1 no artefato 382, que também faz parte de 29 tarefas. Segundo o K_a, o conhecimento do dev1039 sobre esse artefato no dia 01/05/2015 resulta em 0,44. Nesse caso, o modelo K_c inferiu o conhecimento do desenvolvedor em 13 *clusters* do total de 124 *clusters*, com o

K_p igual a 0,13. A medida K_p indica um valor mais próximo da realidade do que o valor K_c para o desenvolvedor dev1039.

Tabela 10 - Resultado da primeira rodada referente à 01/05/2015.

Tarefas Realizadas			Interação		K_c		K_p	
Ord.[27]	Desen.[28]	Tarefas	Ord.	Interações	Ord.	Clusters	Ord.	Medida
1º	dev327	378	1º	50984	1º	121	1º	1802,02
2º	dev3331	31	3º	1397	2º	108	2º	63,75
3º	dev36	13	8º	32	4º	23	4º	3,61
4º	dev919	8	4º	742	3º	41	3º	5,64
5º	dev890	7	7º	33	19º	5	16º	0,06
6º	dev1512	6	23º	1	24º	0	24º	0
7º	dev3974	5	2º	1459	6º	18	7º	1,02
8º	commiter71	4	15º	7	7º	17	12º	0,19
9º	dev84	4	12º	10	11º	15	8º	0,63
10º	dev1039	3	22º	1	13º	13	13º	0,13

A Tabela 11 mostra os resultados da análise de similaridade entre as classificações dos desenvolvedores por tarefas realizadas (linha de base) com as classificações com base no número de interações, e utilizando os modelos K_c e K_p. O estudo foi realizado em 13 rodadas, começando no mês 5 de 2015 e terminando no mês 5 de 2016, sempre no primeiro dia de cada mês. A 13ª rodada foi descartada porque apresentou *p.value* maior que 5% para o K_p. Todas as outras rodadas apresentaram valores de *p.value* menores que 5%, indicando confiança de 95% nos valores de *rho* dos testes de Spearman. Em geral, a classificação baseada no modelo K_p apresenta em média uma maior similaridade com a classificação por tarefas realizadas (72%). Então, com base nesses resultados, é possível responder às três questões de pesquisa deste experimento. SCHOBER *et al*. (2018) discutem as interpretações de resultados dos testes de Spearman. Neste estudo, para simplificar a interpretação dos valores de *rho,* será utilizada a seguinte escala: de 0,00 até 0,10 como correlação negligenciada; de 0,10 até 0,39 como correlação fraca; de 0,40 até

[27] Classificação dos desenvolvedores

[28] Identificação do desenvolvedor

0,69 como correlação moderada; de 0,70 até 0,89 como correlação forte; e de 0,90 até 1,00 como correlação muito forte.

Tabela 11 – Análise de similaridade entre a classificação por tarefas realizadas (linha de base) e as classificações por interação, e utilizando os modelos K_c e K_p.

.		Classificação					
		Interações		K_c		K_p	
rodada	data	rho	p.value	rho	p.value	rho	p.value
1ª	01/05/2015	0,66	<0,045	0,70	<0,032	0,68	<0,036
2ª	01/06/2015	0,66	<0,045	0,70	<0,032	0,68	<0,036
3ª	01/07/2015	0,66	<0,045	0,70	<0,032	0,68	<0,036
4ª	01/08/2015	0,69	<0,036	0,70	<0,032	0,70	<0,032
5ª	01/09/2015	0,69	<0,036	0,70	<0,032	0,66	<0,045
6ª	01/10/2015	0,69	<0,036	0,70	<0,032	0,75	<0,019
7ª	01/11/2015	0,69	<0,036	0,70	<0,032	0,76	<0,016
8ª	01/12/2015	0,69	<0,036	0,70	<0,032	0,76	<0,016
9ª	01/01/2016	0,69	<0,036	0,70	<0,032	0,76	<0,016
10ª	01/02/2016	0,69	<0,036	0,70	<0,032	0,76	<0,016
11ª	01/03/2016	0,69	<0,036	0,71	<0,028	0,75	<0,019
12ª	01/04/2016	0,69	<0,036	0,71	<0,028	0,75	<0,019
	Média	0,68		0,70		0,72	

A questão de pesquisa RQ1 foi definida para avaliar a semelhança entre a lista de desenvolvedores classificados por tarefas realizadas com a lista de desenvolvedores classificados por número de interações realizadas. A Tabela 11 mostra que o grau de semelhança tende a aumentar com o tempo: em 01/05/2015 foi de 66% (*rho*) e em 01/08/2015 tornou-se 69% na 4ª rodada. Essas classificações são semelhantes em 68%, em média, e isso significa que a classificação de tarefas realizadas e a classificação por interações têm correlação (semelhança) moderada. Concluímos que a resposta é sim para RQ1: A classificação de desenvolvedores por tarefas realizadas (R_{pt}) é semelhante à classificação por suas interações com os dados?

A questão de pesquisa RQ2 foi definida para avaliar a semelhança entre a lista de desenvolvedores classificados por tarefas realizadas com a lista de desenvolvedores classificados pelo modelo K_c. A Tabela 11 mostra que o grau de semelhança tende a aumentar com o tempo: em 01/05/2015 foi de 70% (*rho*) e em 01/03/2016 passou para 71%, na 11ª rodada. As classificações são, em média, semelhantes em 70%, um resultado melhor em comparação com a média do RQ1 (68%). Os valores de *p.value* também diminuíram em relação à RQ1, indicando uma tendência em aumentar o nível de

confiança nos valores de correlação (*rho*). De acordo com a escala *likert* para a avaliação de *rho* adotada no planejamento deste estudo, a classificação por tarefas realizadas e a classificação baseada no modelo K_c revelaram uma forte correlação, mas bem próximo à faixa considerada como correlação moderada (*rho* de 0,40 até 0,69). Concluímos que a resposta é também sim para a RQ2: A classificação do R_{pt} é semelhante à classificação gerada pelo modelo K_c?

A questão de pesquisa do RQ3 foi definida para avaliar a semelhança entre a lista de desenvolvedores classificados por tarefas realizadas com a lista de desenvolvedores classificados pelo modelo K_p. Essas classificações são, em média, semelhantes em 72%, um resultado melhor comparado aos resultados de RQ1 e RQ2 que apresentam em média 68% e 70% de similaridade, respectivamente. Os valores de *p.value* também diminuíram significativamente em relação à RQ1 e RQ2 a partir da sexta rodada, indicando um maior grau de confiança para os valores de correlação. De acordo com a escala *likert* para a avaliação de *rho*, a classificação por tarefas realizadas e a classificação baseada no modelo K_p revelaram uma forte correlação, mais acima que o limite superior de 0,69 da correlação classificada como moderada (*rho* de 0,40 até 0,69). Então a resposta também é sim para RQ3: A classificação R_{pt} é semelhante à classificação gerada pelo modelo K_p?

Com relação à análise de sensibilidade dos dados em K_p, a Tabela 12 mostra a influência dos parâmetros R_d, G_d, $\frac{h}{m}$ e $\frac{r}{q}$ em K_p calculados em 11/11/2008 sobre os dados do projeto Mylyn Docs. A melhor organização do projeto em grupos de tarefas semelhantes resultou em 22 grupos ($q = 22$). A segunda coluna apresenta as medidas de K_p conforme a Equação 13. As medidas de K_p para os três desenvolvedores da terceira coluna, quando calculados sem considerar o esquecimento ($R_d = 1$), têm valores maiores que o dobro da segunda coluna, mostrando que a modelagem do esquecimento tem uma influência significativa sobre as medidas de K_p. A quarta coluna com valores de K_p com $G_d = 1$, sem considerar a reaprendizagem, tem valores menores para os dois primeiros desenvolvedores e igual para o terceiro desenvolvedor. Isso indica que os dois primeiros desenvolvedores revisitaram artefatos nos últimos sete dias e o terceiro não. A quarta coluna, as medidas de K_p com valores de h, m, r e q iguais a 1, mostra um valor bem próximo da segunda coluna para o primeiro desenvolvedor, indicando que o primeiro desenvolvedor tem conhecimento sobre a maioria das tarefas e da maioria dos grupos de tarefas semelhantes do projeto (na verdade $r = 22$ para esse desenvolvedor). E o terceiro

desenvolvedor, apresenta um valor de $K_p = 36$ na segunda coluna e 121 na quarta coluna, mostrando uma influência desses parâmetros nas medidas de K_p. Esse desenvolvedor tem conhecimento de somente 7 ($r = 7$) grupos do projeto, no total de 22 ($r = 22$).

Tabela 12 – Análise de sensibilidade de K_p em 11/11/2008.

Ordem	K_p	K_p com $R_d = 1$	K_p com $G_d = 1$	K_p com $h = m = r = q = 1$
1º	2544	5135	2530	2545
2º	588	1643	578	603
3º	36	142	36	121

A Tabela 13 mostra a influência dos parâmetros R_d, G_d, $\frac{h}{m}$ e $\frac{r}{q}$ em K_p, calculados em 11/11/2011, sobre os dados do projeto Mylyn Docs. As medidas de K_p com $R_d = 1$ alteram a lista de classificação dos desenvolvedores em comparação a K_p, o 3º desenvolvedor, segundo o modelo K_p, aparece em 4º ($K_p\ com\ R_d = 1$), o 4º aparece em 5º e o 5º aparece em 3º. A terceira coluna, K_p sem considerar reaprendizagem, mostra que somente o primeiro desenvolvedor possui uma medida menor comparado à primeira coluna, isso acontece porque esse desenvolvedor interagiu com artefatos nos últimos sete dias para realizar tarefas. E finalmente a última coluna, K_p com os parâmetros *h*, *m*, *r* e *q* iguais a 1, mostra que todos os desenvolvedores são penalizados quando não possuem conhecimentos sobre todas as tarefas e grupos do projeto.

Tabela 13 - Análise de sensibilidade de K_p em 11/11/2011.

K_p		K_p com $R_d = 1$		K_p com $G_d = 1$		K_p com $h = n = r = q$	
Ord.	Medida	Ord.	Medida	Ord.	Medida	Ord.	Medida
1º	2751	1º	40561	1º	2698	1º	2814
2º	118	2º	2924	2º	118	2º	128
3º	15,2	4º	71	3º	15,2	3º	66
4º	2,96	5º	4	4º	2,96	4º	26
5º	2,79	3º	87	5º	2,79	5º	15,4

5.2.3 Ameaças à Validade

Segundo WOHLIN *et al.* (2000), ameaças à validade podem afetar um estudo experimental em conclusão, construção, ameaças internas e externas. As ameaças de

conclusão estão relacionadas com a relação entre tratamento e resultado. A linha de base deste estudo é o número de tarefas realizadas pelo desenvolvedor e as outras classificações têm como entrada principal as interações dos desenvolvedores sobre os artefatos durante a execução de tarefas. Alguns desenvolvedores não registraram as interações de edição, apenas as interações de submissão de artefatos para o repositório de código do projeto (*commits*). Portanto, atribuímos um grau igual a 1 para cada interação de submissão. Mesmo assim, dentre os 10 desenvolvedores que mais realizaram tarefas, descobrimos que dois deles realizaram mais tarefas do que interações em artefatos. Como resultado, as correlações entre as classificações podem estar subestimadas.

As ameaças internas avaliam se a relação entre tratamento e resultado é causal ou resulta de fatores que o pesquisador não pode controlar. A classificação baseada em interação não avaliou se uma interação contribuiu para a tarefa ou foi descartada por meio de comandos de desfazer alterações (*undo-type*). Além disso, pode haver uma distinção entre o número e o grau das interações de desenvolvedores com mais e menos experiência na execução de tarefas com o mesmo grau de dificuldade.

As ameaças de construção preocupam-se com a relação entre teoria e observação, assegurando que o tratamento reflete a construção da causa e o resultado reflete a construção do efeito. As classificações baseadas em K_c e K_p não usaram interações do tipo I_{dd}, aquelas observadas entre desenvolvedores e previsto no modelo K_s, porque tais interações não foram medidas no projeto Mylyn Docs.

Finalmente, as ameaças externas estão preocupadas com a generalização dos resultados do experimento em outros projetos de desenvolvimento de software. Nosso modelo é baseado nas interações do desenvolvedor sobre artefatos (I_{da}) e entre os desenvolvedores (I_{dd}). O experimento usou apenas dados de um projeto, o Mylyn Docs, considerando apenas as interações do tipo I_{da}.

5.3 Avaliação do Método Tacin para Recomendar Contexto de Nova Tarefa

O método Tacin recomenda artefatos, tarefas similares e desenvolvedores no início de uma nova tarefa em projeto de desenvolvimento de software. A avaliação do Método investiga a questão de pesquisa desta tese: **QP**- A organização do histórico de elementos do projeto, em função da similaridade por interação dos desenvolvedores sobre artefatos, combinada com a similaridade textual, é suficiente para produzir recomendação de contexto de nova tarefa em projeto de desenvolvimento de software? Como foi descrito no Capítulo 4, a recomendação de artefatos está diretamente relacionada com a

recomendação de tarefas similares à nova tarefa. Tacin recomenda os artefatos das tarefas dos grupos selecionados para recomendação e os desenvolvedores são listados de acordo com os modelos para perceber o conhecimento dos desenvolvedores sobre artefato, tarefa e grupo de tarefas, já avaliados na Seção 5.2.

Um estudo de caso automatizado foi realizado com dados coletados de um projeto de software para avaliar a eficácia do método de recomendação de artefatos e desenvolvedores do Tacin (Apêndice C). O planejamento desse estudo inclui objetivo, o caso, questões de pesquisa e método, de acordo com o guia de RUNESON e HÖST (2008) para conduzir e reportar caso de estudo na Engenharia de Software. O objetivo está descrito no formato indicado pelo GQM (BASILI, 1992):

> **Analisar** o método Tacin para recomendar artefatos e desenvolvedores **com o propósito de** avaliar a sua eficácia **sob o ponto de vista do** desenvolvedor **no contexto de** um projeto de desenvolvimento de software.

De acordo com o objetivo descrito acima, são formuladas duas questões de pesquisa:

> RQ4: O método Tacin é eficaz na recomendação de artefatos no início de uma nova tarefa de desenvolvimento de software?
>
> RQ5: O método Tacin é superior ao KNN ($k = 1$), na recomendação de artefatos e desenvolvedores, no início de uma nova tarefa de desenvolvimento de software?

O KNN é o algoritmo que seleciona os vizinhos mais próximos para produzir recomendações (ZHANG *et al.*, 2016). Nos estudos para a RQ5 iremos utilizar o KNN para encontrar a tarefa mais similar textualmente à nova tarefa ($k = 1$).

O método Tacin reduz o número de artefatos disponíveis no início de uma nova tarefa, procurando não omitir os artefatos que serão editados pela nova tarefa. Rc-measure combina estes dois objetivos de forma harmônica, conforme já apresentado na Seção 2.2. Por isso, a métrica Rc-measure foi escolhida para medir a eficácia do Método. A eficácia do Tacin depende da eficácia da clusterização de tarefas por interação e da correlação entre similaridade textual e similaridade por interação entre as tarefas. Então, foram definidas duas subquestões de pesquisa: RQ4.A: A clusterização de tarefas por interação é eficaz para a recomendação de artefatos? – RQ4.B: Há indícios sobre a existência de correlação entre similaridade textual e similaridade por interação entre tarefas?

5.3.1 Materiais e Métodos

O primeiro estudo foi realizado sobre os dados do projeto Mylyn. O Pseudocódigo 3 mostra o planejamento do estudo para gerar recomendações de artefatos para 20 tarefas do Projeto, simulando uma nova tarefa em cada primeiro dia de mês. A primeira rodada foi realizada em 01/09/2008 porque nessa data já existiam pelo menos 30 tarefas realizadas no projeto Mylyn Docs.

```
1 -  independent_runs := 20;
2 -  date := 2008-09-01;
3 -  for each independent_run {
4 -      clusterByInteraction := buildCluster_similarTasksByInteractions(date);
5 -      newTask := nextTaskAfter (date);
6 -      compute_TextualSimilarity (newTask, clusterByInteraction);
7 -      evaluate_select (FIRST);
8 -      evaluate_select (FIRST_SECOND);
9 -      evaluate_select (FIRST_SECOND_THIRD);
10 -     evaluate_select (ALL);
11 -     date := nextMonth(date);
12- }
```

Pseudocódigo 3- Passos para avaliar recomendações de artefatos com o método Tacin utilizando dados de um projeto de software.

A linha 4 do Pseudocódigo 3 gera grupos de tarefas similares por interação. Duas tarefas são ditas similares por interação quando elas estão relacionadas por interação de edição sobre pelo menos um artefato (Capítulo 3). Para cada rodada, foram realizadas 10 rodadas do LNS e o *cluster* de maior MQ foi escolhido para ser utilizado na avaliação do método Tacin. A clusterização LNS e a medida MQ são descritas na Seção 2.3.

A linha 5 descobre a primeira tarefa que tem interações somente após a data da rodada (variável *date*), ou seja, a nova tarefa para cada rodada. Os artefatos que foram relacionados com as tarefas até a data da rodada e também foram relacionados com a nova tarefa após a sua realização, formam o conjunto de artefatos alvo da recomendação do Tacin. A linha 6 calcula todas as similaridades entre a nova tarefa e os *clusters* calculados na linha 4. O campo texto utilizado foi o *short_description* da nova tarefa com a concatenação de todos os campos *short_description* das tarefas de cada *cluster*. A preparação dos textos utilizou as funções *Replace Tokens, Transform Cases, Tokenize, Filter Stopwords (English), Filter Tokens (by Length) e Stem (Porter)*, também descritas na Seção 2.4. Em seguida, os textos foram representados em vetores utilizando a

ocorrência de termos (*Term Occurrences*). E, finalmente, foram calculadas as similaridades entre os vetores (textos) utilizando a função de similaridade por cosseno.

As linhas 7, 8 e 9 avaliam a recomendação de artefatos do Tacin para responder à questão de pesquisa RQ4. A avaliação contempla três opções de recomendação de acordo com a similaridade textual da nova tarefa com os grupos (*clusters*) de tarefas similares por interação, resultado da linha 6. A primeira opção avalia a recomendação dos artefatos do grupo que possuir a maior similaridade textual com a nova tarefa. A segunda recomenda os artefatos das tarefas dos dois grupos com mais similaridade. E a terceira recomenda os artefatos das tarefas dos 3 primeiros grupos mais similares. Essas 3 opções são avaliadas de acordo com a Cobertura, Redução e Rc-measure para cada rodada. Finalmente, a linha 10 produz dados para responder às subquestões de pesquisa RQ4.A e RQ4.B. Para a subquestão RQ4.A, a maior Cobertura foi selecionada entre os grupos de cada rodada, e ao final, calculada a Cobertura média. E para a RQ4.B, os grupos foram ordenados em ordem decrescente de acordo com a similaridade textual com a nova tarefa. Para cada rodada, a ordem do grupo com maior Cobertura foi selecionada, e ao final, é determinada a moda, ou seja, identificar, caso exista, uma ordem com maior frequência entre as rodadas.

Os estudos, visando responder à subquestão RQ5, utilizaram os dados dos projetos do GitHub para avaliar a recomendação de artefatos e desenvolvedores. Esses estudos compararam os resultados do Tacin com os resultados do algoritmo da tarefa vizinha mais próxima ($KNN\ com\ k = 1$). A recomendação de artefatos comparou a Cobertura da recomendação do Tacin com a Cobertura da recomendação do KNN. A recomendação de desenvolvedores comparou o Tacin e o KNN de acordo com a Cobertura sobre a recomendação de três e cinco desenvolvedores (Top3 e Top5). O Tacin recomendou os desenvolvedores que apresentaram os maiores valores de K_p aplicado sobre o grupo mais similar textualmente à nova tarefa. O KNN recomendou os desenvolvedores que mais realizaram *commits* sobre a tarefa mais similar textualmente à nova tarefa.

5.3.2 Resultados e Análises

Os resultados do estudo sobre o projeto Mylyn Docs, para medir a eficácia do método Tacin na recomendação de artefatos no início de uma nova tarefa de desenvolvimento de software, foram divididos em três opções de avaliação (RQ4). A primeira opção avalia a recomendação dos artefatos das tarefas do grupo mais similar textualmente à nova tarefa (linha 7 do Pseudocódigo 3). A segunda opção avalia a recomendação dos artefatos das

tarefas dos dois grupos mais similares textualmente à nova tarefa (linha 8 do Pseudocódigo 3). E nesta mesma direção, a terceira opção recomenda os artefatos dos 3 grupos mais similares (linha 9 Pseudocódigo 3). Em resumo, a Cobertura média da primeira opção foi de 37%, com 96% de Redução e 41% de Rc-measure. A segunda opção apresentou 45% de Cobertura, 93% de Redução e 48% de Rc-measure. E a terceira obteve 52% de Cobertura, 90% de Redução e 57% de Rc-measure. Observamos, portanto, que a terceira opção é a melhor segundo os valores médios de Rc-measure. A avaliação da clusterização e da correlação por interação e textual entre tarefas, RQ4.A e RQ4.B, foi baseada nos resultados da linha 10 do Pseudocódigo 3. Os resultados indicam que a clusterização contribui mais para a eficácia do Método do que a correlação interação-textual entre tarefas. A clusterização obteve uma Cobertura de 77% escolhendo sempre o grupo com maior cobertura para as novas tarefas. A escolha do grupo por similaridade textual com a nova tarefa obteve uma Cobertura de 37%. Em seguida, os resultados são apresentados em detalhe.

A Tabela 14 mostra os resultados da recomendação de artefatos no caso em que é selecionado o grupo de tarefas similares por interação que possuir maior similaridade textual com a nova tarefa. Para as colunas que identificam o projeto, leia-se, número de tarefas (N.º Δ) e artefatos (N.º □) que apresentam interações Desenvolvedor-Artefato registradas até a data da rodada (coluna 2). A tarefa escolhida como nova tarefa foi a primeira tarefa que apresentou interação após a data da rodada. Os artefatos da nova tarefa, alvo da recomendação, são aqueles que tiveram interações Desenvolvedor-Artefato na sua realização e também tiveram interações Desenvolvedor-Artefato até a data da rodada pelas tarefas do projeto Mylyn Docs. O *Cluster Recomendado* identifica o *id* do *cluster* com a maior similaridade textual com a nova tarefa, coluna *Similaridade*, informando o número de tarefas (N.º Δ) e o número de artefatos (N.º □) do *cluster*. E o sucesso da recomendação de artefatos para cada rodada é avaliada pela Cobertura, Redução e Rc-measure. Por exemplo, para a primeira rodada, a recomendação foi de 29 artefatos do total de 525 artefatos disponíveis no projeto, Redução de 94%, com acerto próximo aos 67% (Cobertura) e apresentando o Rc-measure com 78%.

A coluna Cobertura da Tabela 14 mostra que em 4 rodadas a recomendação conseguiu acertar 100% dos artefatos que foram editados na realização da nova tarefa (7ª, 10ª, 17ª e 18ª rodadas). Em 8 rodadas (40% das rodadas), a recomendação não conseguiu acertar artefatos da nova tarefa. E também em outras 8 rodadas, há sucesso de pelo menos 67% de cobertura na recomendação. O menor percentual de Redução foi alcançado na

terceira rodada, 84%, e nas outras rodadas os resultados foram superiores a 90%. A coluna Rc-measure, em 8 rodadas, as mesmas rodadas de Cobertura igual a 0, apresentam valores iguais a zero. Nas outras 12 rodadas (60%), Rc-measure apresenta valores maiores que zero, indicando grau de sucesso nas recomendações, isto é, cobertura e redução maiores que zero.

A Tabela 15 mostra os resultados de 20 rodadas para avaliar a recomendação dos artefatos das tarefas dos dois grupos mais semelhantes textualmente com a nova tarefa. As colunas "Projeto" e "Nova Tarefa" são um resumo das respectivas colunas na Tabela 14. A coluna *Clusters* informa os *ids* e seus respectivos graus de similaridade textual com a nova tarefa. Em uma rodada, a 16ª, o grupo 1 é o mais similar textualmente à nova tarefa, e por não haver outro grupo similar, foi recomendado o primeiro *cluster* rotulado (id zero). A coluna "Número de Artefatos Recomendados" (N.o □) apresenta os artefatos dos dois *clusters* selecionados, e como na Tabela 14, a coluna "Avaliação" apresenta os valores de Cobertura, Redução e Rc-measure. O número de Cobertura igual a zero (8 rodadas na Tabela 14) caiu para 6, enquanto o número com sucesso de pelo menos 67% de cobertura na recomendação subiu de 8 (Tabela 14) para 10. O percentual de Redução com valor inferior a 90% aconteceu em uma rodada na Tabela 14, subindo para 4 rodadas na Tabela 15. A medida Rc-measure apresentou grau de sucesso em 12 rodadas (60%) na Tabela 14 e agora, na Tabela 15, este número subiu para 14 (70%). Logo, o estudo mostrou que ao incluir o segundo grupo mais similar textualmente à nova tarefa, o Tacin melhora o sucesso na recomendação de artefatos. A recomendação com um grupo resultou em 0,41 de Rc-measure e com dois grupos o Rc-measure subiu para 0,48.

Tabela 14 – Avaliação da recomendação de artefatos para 20 tarefas de acordo com a Cobertura, Redução e Rc-measure. O grupo de tarefas similares por interação que possui maior similaridade textual com a nova tarefa é selecionado, os artefatos das tarefas do grupo são então recomendados.

	Projeto			Nova Tarefa		Cluster Recomendado				Avaliação		
	Data	N.º Δ	N.º □	id	N.º □	id	N.º Δ	Similaridade	N.º □	Cobertura	Redução	Rc-measure
1	01/09/08	32	525	245759	9	4	3	0,21	29	0,67	0,94	0,78
2	01/10/08	47	659	249265	3	8	2	0,71	31	0,67	0,95	0,78
3	01/11/08	63	790	244238	16	3	4	0,26	128	0,75	0,84	0,79
4	01/12/08	77	869	257009	6	1	3	0,45	22	0	0,97	0
5	01/01/09	81	933	258147	10	1	3	0,27	33	0,20	0,96	0,33
6	01/02/09	115	1258	263082	3	1	3	0,45	21	0,67	0,98	0,79
7	01/03/09	141	1354	266524	2	1	4	0,43	87	1	0,94	0,97
8	01/04/09	162	1387	271495	53	6	3	0,46	30	0,08	0,98	0,14
9	01/05/09	175	1534	274706	3	1	3	0,26	74	0	0,95	0
10	01/06/09	196	1583	279382	3	2	3	0,43	36	1	0,98	0,99
11	01/07/09	202	1599	283093	37	1	2	0,34	11	0	0,99	0
12	01/08/09	205	1603	283328	4	5	3	0,26	9	0	0,99	0
13	01/09/09	208	1616	288564	5	3	2	0,23	26	0	0,98	0
14	01/10/09	211	1643	284697	6	1	2	0,56	11	0,17	0,99	0,29
15	01/11/09	220	1656	293066	5	4	10	0,54	21	0,20	0,99	0,33
16	01/12/09	224	1675	296657	2	1	4	0,08	12	0	0,99	0
17	01/01/10	229	1703	298781	4	1	4	0,46	12	1	0,99	1
18	01/02/10	238	1751	302089	5	3	4	0,30	66	1	0,96	0,98
19	01/03/10	245	1827	304013	2	3	5	0,43	166	0	0,91	0
20	01/04/10	248	1852	308670	8	2	6	0,34	23	0	0,99	0
									Média	0,37	0,96	0,41

Tabela 15 - Avaliação da recomendação de artefatos para 20 tarefas de acordo com a Cobertura, Redução e Rc-measure. Os dois grupos de tarefas similares por interação que possuem mais similaridade textual com a nova tarefa são selecionados, os artefatos das tarefas desses grupos são então recomendados.

	Projeto	Nova	Clusters		N.º □	Avaliação		
	N.º □	Tarefa	Id	Sim.	Recomendados	Cob.	Red.	Rc-measure
1	525	245759	4 10	0,21 0,20	33	0,67	0,94	0,78
2	659	249265	8 9	0,71 0,16	59	0,67	0,91	0,77
3	790	244238	3 2	0,26 0,25	173	0,75	0,78	0,77
4	869	257009	12 3	0,45 0,40	131	0,83	0,85	0,84
5	933	258147	13 18	0,27 0,20	84	0,30	0,91	0,45
6	1258	263082	1 24	0,45 0,26	53	0,67	0,96	0,79
7	1354	266524	17 7	0,43 0,40	220	1	0,84	0,91
8	1387	271495	6 49	0,46 0,21	33	0,08	0,98	0,14
9	1534	274706	16 43	0,20 0,71	140	0	0,91	0
10	1583	279382	21 28	0,16 0,26	198	1	0,87	0,93
11	1599	283093	1 27	0,25 0,45	25	0,08	0,98	0,15
12	1603	283328	55 62	0,40 0,27	10	0	0,99	0
13	1616	288564	34 1	0,20 0,45	37	0	0,98	0
14	1643	284697	14 25	0,26 0,43	59	0,67	0,96	0,79
15	1656	293066	46 9	0,40 0,46	67	0,20	0,96	0,33

	Projeto	Nova	Clusters		N.º □	Avaliação		
	N.º □	Tarefa	Id	Sim.	Recomendados	Cob.	Red.	Rc-measure
16	1675	296657	1 0	0,08 0	71	0	0,96	0
17	1703	298781	1 27	0,46 0,26	32	1	0,98	0,99
18	1751	302089	30 24	0,30 0,08	94	1	0,95	0,97
19	1827	304013	33 34	0 0,46	189	0	0,90	0
20	1852	308670	25 30	0,26 0,30	45	0	0,98	0
					Média	0,45	0,93	0,48

A Tabela 16 é igual estruturalmente à Tabela 15, acrescentando a cada rodada, os três grupos mais similares à nova tarefa na coluna *Clusters*. Na rodada 16, por haver somente um grupo similar textualmente à nova tarefa, foram selecionados os dois grupos com menores Ids (0 e 2) que ainda não haviam sido selecionados. O número de Cobertura igual a zero (8 na Tabela 14 e 6 na Tabela 15) caiu para 4, enquanto o número com sucesso de pelo menos 67% na recomendação que ficou em 8 na Tabela 14, subiu para 10 na Tabela 15, permaneceu em 10 na Tabela 16. O percentual de Redução com valor inferior a 90% aconteceu em uma rodada na Tabela 14, subiu para 4 rodadas na Tabela 15, e foi de 7 na Tabela 16. A medida Rc-measure apresentou grau de sucesso em 12 rodadas na Tabela 14 (60% das rodadas) subiu para 14 (70%) na Tabela 15, e para 16 (80%) na Tabela 16. Logo, o estudo mostrou que ao incluir o terceiro grupo mais similar textualmente à nova tarefa, o Tacin melhora ainda mais o sucesso na recomendação de artefatos. A recomendação considerando somente o grupo mais similar textualmente à nova tarefa obteve 0,41 de Rc-measure, com dois grupos, o Rc-measure subiu para 0,48 e, finalmente com três grupos, o Rc-measure ficou em 0,57.

Tabela 16 - Avaliação da recomendação de artefatos para 20 tarefas de acordo com a Cobertura, Redução e Rc-measure. Os três grupos de tarefas similares por interação que possuem mais similaridade textual com a nova tarefa são selecionados, os artefatos das tarefas desses grupos são então recomendados.

	Projeto	Nova Tarefa	Clusters		N.º □ Recomendados	Avaliação		
	N.º □		Id	Sim.		Cob.	Red.	Rc-measure
1	525	245759	4 10 7	0,21 0,20 0,19	48	0,67	0,91	0,77
2	659	249265	8 9 15	0,71 0,16 0,12	64	0,67	0,90	0,77
3	790	244238	3 2 0	0,26 0,25 0,20	230	0,75	0,71	0,73
4	869	257009	12 3 7	0,45 0,40 0,35	159	0,83	0,82	0,83
5	933	258147	13 18 5	0,27 0,20 0,15	116	0,50	0,88	0,64
6	1258	263082	1 24 28	0,45 0,26 0,18	88	0,67	0,93	0,78
7	1354	266524	17 7 33	0,43 0,40 0,36	224	1	0,83	0,91
8	1387	271495	6 49 15	0,46 0,42 0,30	94	0,23	0,93	0,36
9	1534	274706	16 43 22	0,26 0,20 0,14	462	0	0,70	0
10	1583	279382	21 28	0,43 0,43	200	1	0,87	0,93

	Projeto	Nova Tarefa	Clusters		N.º □ Recomendados	Avaliação		
	N.º □		Id	Sim.		Cob.	Red.	Rc-measure
			38	0,35				
11	1599	283093	1 27 15	0,34 0,29 0,27	67	0,27	0,96	0,42
12	1603	283328	55 62 66	0,26 0,22 0,20	11	0	0,99	0
13	1616	288564	34 1 31	0,23 0,15 0,13	42	0,40	0,97	0,57
14	1643	284697	14 25 30	0,56 0,37 0,29	69	0,67	0,96	0,79
15	1656	293066	46 9 47	0,53 0,49 0,46	97	0,20	0,94	0,33
16	1675	296657	1 0 2	0,08 0 0	88	0	0,95	0
17	1703	298781	1 27 30	0,46 0,26 0,24	90	1	0,95	0,97
18	1751	302089	30 24 29	0,30 0,28 0,22	110	1	0,94	0,97
19	1827	304013	33 34 7	0,43 0,29 0,29	350	0	0,81	0
20	1852	308670	25 30 34	0,34 0,31 0,27	80	0,50	0,96	0,66
					Média	0,52	0,90	0,57

A eficácia do método Tacin depende dos seguintes resultados: A- da clusterização de tarefas por interação; B- e da correlação entre as similaridades textual e por interação entre tarefas. A clusterização pode ser avaliada pela Cobertura média sobre as 20 rodadas. A Tabela 17 mostra esses dados. Por exemplo, a primeira rodada gerou uma clusterização com 14 *clusters* (4 + 10), sendo que 4 apresentam similaridade textual maior que zero com a nova tarefa. Nessa rodada, o *cluster* que apresenta maior Cobertura é o que também apresenta maior similaridade textual com a nova tarefa. Outro exemplo, a quarta rodada, apresenta uma clusterização que em 4 *clusters* houve similaridade textual com a nova tarefa e também houve sucesso na recomendação (Cobertura = 0,83). A maior Cobertura para a quarta rodada foi de 83%, no segundo e quinto *clusters* com mais similaridade textual à nova tarefa. Em resumo, a Tabela 17 mostra que em média as 20 rodadas apresentaram uma clusterização com Cobertura média de 77%. A coluna Similaridade indica uma moda (1º *cluster* mais similar textualmente) na associação da nova tarefa ao *cluster* com maior Cobertura, mas indica também que não houve correlação entre as duas similaridades (interação e textual) em 4 das 20 rodadas (20%).

Tabela 17 - Resultado de 20 rodadas para avaliação da clusterização de tarefas semelhantes por interação e da correlação entre similaridades por interação e textual.

	N.º clusters		Recomendação		Melhor Cluster		
	Sim.		N.º clusters		Máximo		
	> 0	= 0	(Positivo, Positivo) Sim. > 0 & Cobertura > 0	(Falso, Positivo) Sim. = 0 & Cobertura > 0	Cob.	Red.[29]	Similaridade
1	4	10	1	1	0,67	0,95	1º *cluster*
2	8	11	2	2	0,67	0,95	1º e 4º *clusters*
3	15	7	5	3	0,75	0,84	1º *cluster*
4	20	7	4	1	0,83	0,87	2º e 5º *clusters*
5	12	15	9	3	0,5	0,66	-
6	10	28	3	3	0,67	0,98	1º e 5º *clusters*
7	40	4	5	1	1	0,94	1º, 2º e 6º *clusters*
8	42	12	21	2	0,36	0,98	6º *cluster*

[29] Redução referente ao primeiro *cluster* com Cobertura máxima.

9	6	53	0	3	1	0,95	-
10	44	21	1	0	1	0,98	1º *cluster*
11	26	41	14	12	0,38	0,97	7º *cluster*
12	7	60	2	6	0,75	0,97	-
13	12	55	3	6	0,8	0,97	9º *cluster*
14	22	46	8	7	0,67	0,97	2º *cluster*
15	51	18	13	1	0,6	0,94	5º *cluster*
16	1	69	0	1	1	0,99	-
17	13	59	2	4	1	0,99	1º *cluster*
18	27	47	10	2	1	0,96	1º *cluster*
19	56	18	2	0	1	0,96	46º *cluster*
20	26	50	9	7	0,75	0,92	6º *cluster*
				Média	0,77	0,94	**Moda** 1º *cluster*

A Tabela 18 mostra o comparativo entre os resultados das 3 opções de recomendação de artefatos para uma tarefa de desenvolvimento de software. Os resultados indicam que o Tacin é mais eficaz quando recomenda os três grupos mais similares à nova tarefa. A Tabela 16 apresenta resultados melhores em Cobertura e Rc-measure, e uma pequena desvantagem na medida de Redução. Por isso, com relação à RQ4 (O método Tacin é eficaz na recomendação de artefatos no início de uma nova tarefa de desenvolvimento de software?), considerando os resultados da Tabela 16, a resposta é que o método Tacin é eficaz com Cobertura média de 52%, Redução média de 90% e Rc-measure média de 57%.

Tabela 18 – Comparativo entre as três opções de recomendação, o primeiro (Tabela 14), o primeiro e segundo (Tabela 15) ou os três primeiros grupos (Tabela 16) de tarefas similares por interação que possuem mais similaridade textual com a nova tarefa.

	Tabela 14	Tabela 15	Tabela 16
Número de rodadas com Cobertura=0	8	6	4
Número de rodadas com Cobertura=1	4	4	4
Número de rodadas com Cobertura > 0	12	14	16
Cobertura Média	0,37	0,45	0,52
Redução Média	0,96	0,93	0,90
Rc-measure Média	0,41	0,48	0,57

A eficácia do método Tacin pode ser melhor investigada em função das subquestões RQ4.A (A clusterização de tarefas por interação é eficaz para a recomendação de artefatos?) e RQ4.B (Há indícios sobre a existência de correlação entre similaridade textual e similaridade por interação entre tarefas?). A RQ4.A pode ser respondida observando as médias de Cobertura e Redução da Tabela 17. Essas médias indicam que a clusterização de tarefas similares por interação irá produzir *clusters* que reunirão somente 6% (94% de Redução) dos artefatos disponíveis no projeto, mas contendo 77% dos artefatos (Cobertura) que serão editados por uma nova tarefa. Além disso, a clusterização apresenta melhores médias na Tabela 16 (52% de Cobertura e 90% de Redução). A conclusão é que a resposta é sim para RQ4.A quando combinado LNS com MQ.

Especificamente sobre RQ4.B, a última coluna da Tabela 17 mostra que a similaridade textual da nova tarefa com os *clusters* associou a nova tarefa a um dos *clusters* com Cobertura máxima em 8 rodadas (1ª, 2ª, 3ª, 6ª, 7ª, 10ª, 17ª e 18ª). Portanto, apresentando uma correlação ótima entre similaridade por interação e textual em 40% das rodadas. No entanto, em 10 rodadas (5ª, 8ª, 9ª, 11ª, 12ª, 13ª, 15ª, 16ª, 19ª e 20ª), os 3 *clusters* mais similares textualmente à nova tarefa não apresentaram Cobertura máxima, dentre esses, em 4 rodadas (5ª, 9ª, 12ª, 16ª), não houve correlação entre similaridade por interação e textual, ou seja, não houve similaridade textual entre a nova tarefa e o *cluster* com Cobertura máxima. A conclusão é que a resposta é sim também para a RQ4.B, existindo, portanto, indícios de correlação entre similaridade textual e similaridade por interação nas tarefas de desenvolvimento de software.

Como foi constatado acima, há necessidade de melhorar o grau de sucesso da associação textual da nova tarefa com o *cluster* de maior Cobertura. A Tabela 19 mostra que para textos pequenos, que incluem somente a descrição curta da tarefa, os resultados são melhores para OT (ocorrência de termos, do original em inglês *Term Occurrences*) e TF (frequência de temos, *Term Frequency*). A inserção do texto de descrição da tarefa fez com que a Cobertura média diminuísse mais para OT e TF, caiu para 14%, e para o TF-ID (*Term Frequency–Inverse Document Frequency*), caiu de 30% para 20%. Conclui-se que, em média, a inserção de mais texto não contribui para uma melhor associação entre a nova tarefa e o *Cluster* com melhor Cobertura.

Tabela 19 – Cobertura média de associação da nova tarefa ao Cluster com maior similaridade textual em 20 rodadas. Os textos de descrição curta e descrição das tarefas foram avaliadas de acordo com OT, TF e TF-ID.

	Cobertura Média		
Texto da Tarefa	OT	TF	TF-ID
Descrição curta	0,37 (Tabela 14)	0,37	0,30
Descrição curta mais Descrição	0,14	0,14	0,20

As Tabelas 20-23 apresentam os resultados dos estudos de comparação do Tacin com o KNN na recomendação de artefatos e desenvolvedores. Os estudos foram realizados com dados de quatro projetos do GitHub para responder à questão de pesquisa RQ5. A Tabela 20 mostra o resultado de 20 rodadas para o projeto Homebrew/homebrew-cask-drivers. A primeira rodada inicia com 26 tarefas, 50 artefatos e 16 desenvolvedores, com incremento de 30 dias entre as rodadas, até a vigésima rodada, que apresenta 557 tarefas, 233 artefatos e 183 desenvolvedores. As novas tarefas foram realizadas por um desenvolvedor em 16 rodadas e por dois desenvolvedores em quatro rodadas. As novas tarefas também apresentam uma relação de interação Desenvolvedor-Artefato com um artefato em 18 rodadas e zero artefato em duas rodadas.

Quanto à recomendação de artefatos para o projeto Homebrew/homebrew-cask-drivers, o Tacin recomendou o grupo mais similar textualmente à nova tarefa e obteve uma Cobertura média de 75%, enquanto que o KNN, recomendando a tarefa mais similar textualmente à nova tarefa, obteve uma Cobertura média de 65%. Cabe ressaltar que o Tacin recomendou muitos artefatos em três rodadas: 29 artefatos na 1ª rodada e 11 artefatos em duas rodadas (8ª e 10ª). Com relação à recomendação de desenvolvedores, o Tacin foi superior à recomendação do KNN, tanto na lista de três (Top3) quanto na lista de 5 (Top5) desenvolvedores recomendados. Em ambas a listas, Top3 e Top5, o Tacin obteve uma Cobertura média de 53% contra 30% do KNN.

A Tabela 21 apresenta os dados do projeto Homebrew/homebrew-formula-analytics para 20 rodadas, iniciando com 29 tarefas, 4 artefatos e 6 desenvolvedores e terminando com 88 tarefas, 9 artefatos e 8 desenvolvedores. Portanto, esse projeto representa um cenário onde poucos desenvolvedores produzem interações sobre poucos artefatos para realizar muitas tarefas. Nesse cenário, o Tacin e o KNN recomendam um ou dois artefatos e apresentam a mesma Cobertura média, 95%. E na recomendação de

desenvolvedores, o Tacin foi superior ao KNN em ambas as listas, Top3 e Top5, apresentando a mesma Cobertura média de 93% contra 83%.

A Tabela 22 apresenta os dados para o projeto brewsci/homebrew-science. Esse projeto representa um cenário onde poucos desenvolvedores produzem interações sobre muitos artefatos para realizar muitas tarefas. Nesse cenário, o Tacin também foi superior, apresentando uma Cobertura média de 80% contra 75% do KNN na recomendação de artefatos. O Tacin também foi superior ao KNN na recomendação de desenvolvedores: na recomendação Top3 obteve Cobertura média de 78% contra 63% do KNN, e na recomendação Top5 obteve Cobertura média de 88% contra 63% do KNN.

O último estudo comparou as recomendações do Tacin e do KNN sobre os dados do projeto Kerl/kerl. A Tabela 23 mostra que nesse projeto muitos desenvolvedores realizam muitas tarefas utilizando poucos artefatos. As rodadas apresentaram novas tarefas realizadas, em maioria por mais de um desenvolvedor, com interações Desenvolvedor-Artefato sobre um ou dois artefatos. O Tacin obteve Cobertura média de 93% contra 88% do KNN na recomendação de artefatos. O Tacin também superou o KNN na recomendação de desenvolvedores: na lista Top3 apresentou Cobertura média de 47% contra somente 19% do KNN, e na Top5 aumentou a Cobertura média para 63% enquanto o KNN permaneceu com 19%.

Em resposta à questão de pesquisa RQ5, a conclusão é que o Tacin é superior ao KNN ($k = 1$) na recomendação de artefatos e desenvolvedores, no início de uma nova tarefa de desenvolvimento de software, quando considera os valores médios de Cobertura.

Tabela 20 – Comparação da recomendação de artefatos e desenvolvedores entre o Tacin (recomendação do primeiro grupo) e o KNN ($k = 1$) sobre o projeto Homebrew/homebrew-cask-drivers.

	Projeto			Nova Tarefa			Recomendação de Artefatos				Recomendação de Desenvolvedores			
							Tacin		KNN		Cobertura Top3		Cobertura Top5	
	N.º △	N.º □	N.º	id	N.º □	N.º	N.º □	Cobertura	N.º □	Cobertura	K_p	KNN	K_p	KNN
1	26	50	16	37	1	1	29	1	0	0	0	0	0	0
2	46	99	25	58	1	2	1	1	0	0	0	0	0	0
3	69	100	28	82	1	1	1	0	1	0	0	0	0	0
4	89	108	39	106	1	1	1	1	1	1	1	1	1	1
5	112	117	52	130	0	2	1	0	1	0	0,5	0,5	0,5	0,5
6	137	121	59	159	1	1	1	1	1	1	1	1	1	1
7	170	136	76	202	0	1	1	0	1	0	0	0	0	0
8	213	185	87	248	1	2	11	1	1	1	1	0,5	1	0,5
9	249	192	98	286	1	1	1	1	1	1	1	1	1	1
10	275	199	107	315	1	2	11	0	1	0	0	0	0	0
11	296	201	112	340	1	1	1	1	1	1	1	0	1	0
12	316	203	121	364	1	1	1	1	1	1	1	0	1	0
13	344	205	129	396	1	1	1	1	1	1	1	1	1	1
14	370	211	137	428	1	1	2	0	1	0	0	0	0	0
15	390	216	141	452	1	1	1	1	1	1	1	0	1	0
16	412	219	147	478	1	1	1	1	1	1	0	0	0	0
17	427	222	155	495	1	1	3	1	1	1	1	1	1	1
18	472	224	163	547	1	1	1	1	1	1	1	0	1	0
19	524	228	176	606	1	1	3	1	1	1	0	0	0	0
20	557	233	183	648	1	1	1	1	1	1	0	0	0	0
						Média:		0,75		0,65	0,53	0,30	0,53	0,30

Tabela 21 - Comparação da recomendação de artefatos e desenvolvedores entre o Tacin (recomendação do primeiro grupo) e o KNN ($k = 1$) sobre o projeto Homebrew/homebrew-formula-analytics.

	Projeto			Nova Tarefa			Recomendação de Artefatos				Recomendação de Desenvolvedores			
							Tacin		KNN		Cobertura Top3		Cobertura Top5	
	N.º △	N.º □	N.º	id	N.º □	N.º	N.º □	Cobertura	N.º □	Cobertura	K_p	KNN	K_p	KNN
1	29	4	6	35	1	1	1	1	1	1	1	1	1	1
2	30	4	6	36	1	1	1	1	1	1	1	1	1	1
3	31	4	6	37	1	1	1	1	1	1	1	1	1	1
4	34	4	6	40	1	1	1	1	1	1	1	1	1	1
5	36	4	6	43	1	1	1	1	1	1	1	1	1	1
6	38	4	6	44	1	1	2	1	2	1	1	1	1	1
7	39	4	6	45	1	1	1	1	1	1	1	1	1	1
8	41	4	6	47	1	1	1	1	1	1	1	1	1	1
9	43	4	6	49	1	2	1	1	1	1	0,5	0,5	0,5	0,5
10	45	4	6	51	0	1	1	0	1	0	0	0	0	0
11	49	8	7	55	1	1	1	1	1	1	1	1	1	1
12	55	8	8	61	1	1	1	1	1	1	1	1	1	1
13	59	8	8	65	1	1	1	1	1	1	1	1	1	1
14	60	8	8	68	1	1	2	1	1	1	1	1	1	1
15	66	8	8	72	1	1	2	1	1	1	1	1	1	1
16	70	8	8	76	1	1	1	1	1	1	1	1	1	1
17	75	9	8	81	1	1	1	1	1	1	1	1	1	1
18	80	9	8	86	1	1	2	1	1	1	1	1	1	1
19	84	9	8	90	1	1	1	1	1	1	1	0	1	0
20	88	9	8	94	1	1	1	1	1	1	1	0	1	0
						Média:		0,95		0,95	0,93	0,83	0,93	0,83

Tabela 22 - Comparação da recomendação de artefatos e desenvolvedores entre o Tacin (recomendação do primeiro grupo) e o KNN ($k = 1$) sobre o projeto brewsci/homebrew-science.

	Projeto			Nova Tarefa			Recomendação de Artefatos				Recomendação de Desenvolvedores			
							Tacin		KNN		Cobertura Top3		Cobertura Top5	
	N.º △	N.º □	N.º 🤖	id	N.º □	N.º 🤖	N.º □	Cobertura	N.º □	Cobertura	K_p	KNN	K_p	KNN
1	43	38	5	62	1	1	28	1	2	1	1	1	1	1
2	47	45	6	70	0	1	2	0	1	0	1	1	1	1
3	50	49	6	76	1	1	34	1	2	1	1	1	1	1
4	52	51	6	79	1	1	36	1	2	1	1	1	1	1
5	60	59	7	96	1	1	44	1	2	1	1	1	1	1
6	62	60	7	120	1	1	45	1	2	1	1	1	1	1
7	65	63	8	129	1	1	48	1	2	1	1	0	1	0
8	70	67	9	139	1	1	52	1	2	1	1	1	1	1
9	81	87	9	159	1	1	62	1	2	1	0	0	1	0
10	85	96	10	171	1	1	64	1	2	1	0	0	1	0
11	89	100	10	181	1	1	68	1	2	1	1	0	1	0
12	91	102	10	184	1	1	70	1	2	1	1	1	1	1
13	96	107	11	190	0	2	1	0	1	0	0,5	0,5	0,5	0,5
14	98	112	12	194	0	2	4	0	4	0	0	0	0	0
15	108	121	13	207	2	1	81	1	2	0,5	1	1	1	1
16	112	123	13	213	1	1	83	1	2	1	1	0	1	0
17	115	126	13	217	2	1	86	1	2	0,5	1	1	1	1
18	117	128	13	221	0	1	7	0	9	0	0	0	0	0
19	121	132	14	229	1	1	88	1	2	1	1	1	1	1
20	123	133	14	235	1	1	89	1	2	1	1	1	1	1
						Média:		0,80		0,75	0,78	0,63	0,88	0,63

Tabela 23 - Comparação da recomendação de artefatos e desenvolvedores entre o Tacin (recomendação do primeiro grupo) e o KNN ($k = 1$) sobre o projeto Kerl/kerl.

	Projeto			Nova Tarefa			Recomendação de Artefatos				Recomendação de Desenvolvedores			
							Tacin		KNN		Cobertura Top3		Cobertura Top5	
	N.º △	N.º □	N.º	id	N.º □	N.º	N.º □	Cobertura	N.º □	Cobertura	K_p	KNN	K_p	KNN
1	32	4	25	61	1	1	1	1	1	1	1	1	1	1
2	33	4	25	65	1	1	2	1	1	1	0	0	0	0
3	34	4	26	73	1	2	1	1	1	1	0	0,5	0,5	0,5
4	37	4	30	78	1	1	1	1	1	1	0	0	0	0
5	51	5	44	115	1	2	1	1	1	1	0,5	0	0,5	0
6	55	5	46	121	1	2	3	1	1	1	0,5	0	1	0
7	58	5	46	123	1	2	2	0	1	0	0,5	0	0,5	0
8	61	5	47	130	1	2	3	1	2	1	0,5	0,5	1	0,5
9	66	5	48	137	1	3	1	1	1	1	0,67	0	0,67	0
10	70	5	51	144	1	1	1	1	1	1	0	0	1	0
11	71	5	51	91	2	4	3	1	1	0,5	0,25	0	0,5	0
12	72	5	51	147	2	2	2	0,5	1	0,5	0,5	0	0,5	0
13	80	5	56	161	2	2	3	1	3	1	0,5	0,5	0,5	0,5
14	84	5	56	168	1	2	1	1	1	1	0,5	0,5	0,5	0,5
15	86	5	56	174	1	2	4	1	1	1	0,5	0	1	0
16	91	5	57	181	1	1	1	1	1	1	1	0	1	0
17	96	7	57	196	1	1	2	1	1	1	1	0	1	0
18	106	7	60	214	1	4	2	1	1	1	0,5	0,25	0,5	0,25
19	117	7	66	244	1	2	2	1	2	1	0,5	0,5	0,5	0,5
20	120	8	68	251	2	2	2	1	1	0,5	0,5	0	0,5	0
						Média:		0,93		0,88	0,47	0,19	0,63	0,19

5.3.3 Ameaças à Validade

As ameaças à validade destes estudos de caso são descritas de acordo com a validade de construção, validade interna, validade externa e confiabilidade (RUNESON e HÖST, 2008). O primeiro estudo avaliou se a associação de uma nova tarefa a grupos de tarefas similares por interação Desenvolvedor-Artefato, por meio de similaridade textual, produz bons resultados para a recomendação de artefatos, no início da nova tarefa. A clusterização por interação não considerou o grau de relevância que um artefato tem sobre uma tarefa, determinando peso igual a 1 para todos os artefatos relacionados com uma tarefa por interação Desenvolvedor-Artefato. O primeiro estudo também não comparou a eficiência de técnicas de similaridade textual, por exemplo, técnicas textuais para textos pequenos e com erros gramaticais (O'SHEA *et al.*, 2008). O primeiro estudo foi realizado somente sobre um projeto de software e considerou 20 rodadas. Consequentemente, o formalismo estatístico com inclusão de grau de confiança sobre a análise dos resultados também não foi realizado. Os estudos de comparação do Tacin com o KNN foram realizados considerando somente o primeiro grupo do Tacin (recomendação do primeiro grupo) com a primeira tarefa do KNN ($k = 1$).

A validade interna do estudo questiona as relações entre tratamento e saída. A clusterização das tarefas do projeto por similaridade de interação Desenvolvedor-Artefato mostrou-se satisfatória ao utilizar LNS e a heurística MQ. Embora o primeiro estudo tenha sido realizado em somente um projeto simulando a recomendação de 20 tarefas, há uma indicação de que o tratamento LNS+MQ irá produzir grupos adequados para a recomendação de contexto no início de uma nova tarefa. No entanto, parte das interações de edições foram simuladas a partir de *commits* ao repositório do projeto, possibilitando, portanto, uma ameaça a essa indicação de sucesso. Na parte de associação da nova tarefa ao grupo com a melhor Cobertura, o resultado pode ter sido melhor que o real já que muitas tarefas apresentam o nome do módulo no campo "Descrição curta". Os projetos do GitHub registram as interações Desenvolvedor-Artefato através de *commits*, não registrando muitas interações realizadas pelos desenvolvedores sobre os artefatos por meio de ferramentas de monitoramento como, por exemplo, o Mylyn.

A validade externa está relacionada à generalização do estudo. Os estudos foram planejados para utilizar dados de projetos de software com o objetivo de avaliar a recomendação de artefatos e desenvolvedores para novas tarefas. No contexto real de uma aplicação de recomendação, o usuário (desenvolvedor) pode ser afetado por inúmeros

fatores, tais como sociais, emocionais, recompensas, facilidade de utilização da ferramenta de recomendação etc. Então, a generalização dos resultados virá somente através da avaliação do Tacin em projetos acadêmicos e posteriormente em projetos reais.

A confiabilidade do estudo diz respeito à questão de se os dados e sua análise são independentes do pesquisador. A Seção 5.1 mostra como foi realizada a coleta de dados de cinco projetos de desenvolvimento de software de código aberto. O planejamento, execução e análise dos estudos foram descritos em detalhe nas Seções 5.3, 5.3.1 e 5.3.2, respectivamente.

5.4 Considerações Finais

O capítulo descreveu como foi realizada a coleta de dados do projeto Mylyn Docs, hospedado no Eclipse, e de quatro projetos hospedados no GitHub: Homebrew/homebrew-cask-drivers, Homebrew/homebrew-formula-analytics, brewsci/homebrew-science e Kerl/kerl. O projeto Mylyn Docs foi utilizado na avaliação dos Modelos de Percepção do Conhecimento e foram utilizados todos os cinco projetos na avaliação do método Tacin para recomendar o contexto para uma nova tarefa em projeto de desenvolvimento de software.

A avaliação verificou a similaridade entre as listas de classificação dos desenvolvedores do projeto Mylyn Docs. A lista de desenvolvedores por número de tarefas realizadas foi comparada com as listas de desenvolvedores por número de interações Desenvolvedor-Artefato, segundo o modelo K_c e segundo o modelo K_p. O resultado mostrou que a lista de classificação que utilizou o modelo K_p apresentou maior similaridade com a lista de tarefas realizadas.

A avaliação do método Tacin para recomendar o contexto para uma nova tarefa em projeto de desenvolvimento de software avaliou a recomendação de artefatos e desenvolvedores. A avaliação da recomendação de artefatos foi realizada sobre os cinco projetos e a avaliação de desenvolvedores foi realizada sobre quatro projetos do GitHub. Os resultados indicam que o Tacin supera o KNN $(k = 1)$ com base nos resultados de Cobertura média. Um teste mais rigoroso pode ser realizado para verificar a superioridade do Tacin sobre o KNN utilizando o teste de hipóteses (JURISTO e MORENO, 2010).

Este capítulo conclui que a organização do histórico de elementos do projeto, em função da similaridade por interação dos desenvolvedores sobre artefatos, combinada com a similaridade textual, é suficiente para produzir recomendação de contexto de nova tarefa em projeto de desenvolvimento de software.

6- Trabalhos Relacionados

Este capítulo inicia descrevendo como os trabalhos relacionados foram selecionados para esta tese, segue descrevendo-os, ressaltando o objetivo da recomendação, como recomendam e como foram avaliados. Compara os trabalhos de acordo com as características de utilização das técnicas de clusterização e Processamento de Linguagem Natural, se modelam similaridade por interação, esquecimento humano e reaprendizagem, informa também se o trabalho mede conhecimento sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software, e se recomendam artefatos, tarefas similares, especialistas em artefato, tarefa e tarefas similares. O capítulo finaliza indicando um caminho para a comparação dos trabalhos relacionados de acordo com a eficácia.

Na literatura existem muitos trabalhos sobre Sistemas de Recomendação, por exemplo, NAJAFABADI *et al.* (2017) realizaram uma análise crítica sobre 131 artigos. Na área da Engenharia de Software, os problemas mais mitigados são em recomendação de artefatos e especialistas com apoio de ferramentas para suporte de reutilização, depuração, implementação e manutenção (GASPARIC e JANES, 2016). Os trabalhos relacionados desta tese foram selecionados a partir de uma quasi-revisão sistemática (QRS) incluindo/partindo da revisão de GASPARIC e JANES (2016).

A QRS se orientou no protocolo de WOHLIN *et al.* (2012) utilizando apoio ferramental[30] (para mais informações, veja o Apêndice A). O objetivo foi identificar trabalhos que têm como objetivo recomendar artefatos, tarefas e/ou desenvolvedores em projetos de desenvolvimento de software. As questões de pesquisa foram: 1- Quais são as técnicas utilizadas para recomendar contexto em projetos de desenvolvimento de software? 2- O que recomendam? (pessoas; artefatos e/ou tarefas e/ou projeto); 3- Quais são os dados utilizados para recomendar contexto em projetos de software? 4- Quais as medidas utilizadas na avaliação da recomendação?

A *string* de busca da QRS foi executada na Scopus[31] e na ScienceDirect[32] . Além disso, os artigos referenciados pelas seguintes revisões da literatura foram adicionados a esta revisão: (INZUNZA et *al.*, 2016); (GASPARIC e JANES, 2016) e (SUN *et al.*, 2016).

[30] https://parsif.al/

[31] www.scopus.com

[32] www.sciencedirect.com

O resultado foi um conjunto de 2514 artigos. O autor desta tese, com base na leitura dos títulos e resumos, selecionou os estudos primários que apresentaram uma ou mais técnicas para recomendar contexto em projetos de desenvolvimento de software, resultando na seleção de 32 artigos. Em seguida os artigos foram lidos e classificados de acordo com os critérios de qualidade estabelecidos no planejamento da QRS. O resultado final foi um conjunto de 18 artigos, onde 16 deles são discutidos a seguir e apenas dois, ao final, serão mencionados, em função de serem trabalhos relacionados mais distantes em relação a esta tese.

A Tabela 24 lista os 16 trabalhos relacionados com esta tese. A primeira coluna da Tabela 24 informa a referência do trabalho; a segunda indica se o trabalho fez uso das técnicas de clusterização e Processamento de Linguagem Natural (NLP); a terceira coluna explicita se o trabalho modela ou não Similaridade por Interação (Sim. por Interação), Esquecimento humano (Esquec.) e Reaprendizagem (Reaprend.); a quarta coluna informa se o trabalho mede conhecimento sobre Artefato (□), Tarefa (Δ), Tarefas Similares (Δs Similares) e todo o Projeto de Desenvolvimento de Software; e a última coluna mostra se o trabalho recomenda artefatos (□), tarefas similares (Δs Similares), especialistas em artefatos, tarefas e tarefas similares. Os trabalhos foram ordenados de acordo com o objetivo da recomendação.

A seguir, cada trabalho relacionado será apresentado, identificando-se o objetivo da recomendação, como recomendam e informações sobre a avaliação.

6.1 Apresentação dos Trabalhos Relacionados

ALMHANA *et al.* (2016) observaram que a recomendação de artefatos relevantes para apoiar o desenvolvedor na correção de um erro pode ser modelada como um problema multiobjetivo, maximizando a relevância dos artefatos recomendados e minimizando o número de artefatos recomendados. O algoritmo faz uso das APIs de documentação para medir a similaridade entre os fragmentos de código e os erros de software registrados no histórico do projeto, e também da associação entre artefatos e erros já corrigidos. Presume-se também que artefatos alterados recentemente, por ocasião de correção de erros, têm maior probabilidade de serem alterados na correção de um novo erro. A avaliação mostrou uma forte evidência de que a proposta pode recomendar artefatos relevantes na maioria das vezes.

ANDRIC *et al.* (2004) relacionaram o histórico de artefatos através dos seus metadados para apoiar a busca por artefatos usando um texto base como entrada. Na

avaliação, 10 usuários utilizaram 10 termos aleatórios que já foram utilizados em buscas anteriores. Os termos são aplicados à proposta e a um sistema de busca que considera todo o texto. Cada usuário indicou a relevância dos itens retornados para cada par termo-sistema. O resultado indicou que o sistema proposto suplanta as técnicas tradicionais de procura textual que consideram todo o texto disponível.

ANTUNES *et al.* (2012) propuseram um sistema de recomendação para recomendar uma lista de artefatos relevantes que podem ajudar no trabalho do desenvolvedor. O sistema proposto utiliza as interações dos usuários em tempo real e tempo de acesso aos artefatos para listar os artefatos mais relevantes. Depois utiliza também as relações da estrutura da linguagem (Java) e associações textuais para ordenar os artefatos recomendados de forma decrescente de relevância. A avaliação mostrou uma Cobertura média de 42,7%.

CodeRAnts é um método de recomendação para recomendar artefatos. O Método se baseia na repetição das buscas textuais realizadas pelos programadores e na metáfora da colônia de formigas (CAICEDO e DUARTE, 2015). Primeiro, CodeRAnts constrói um grafo direcionado, onde os vértices representam as consultas realizadas no passado e as arestas como a combinação da similaridade textual, da correlação e da confiança entre os vértices. Depois, para uma nova consulta, é escolhido o vértice mais similar textualmente à nova consulta, ponto inicial da "caminhada da formiga" no grafo (para detalhes sobre o algoritmo da caminhada, veja CAICEDO e DUARTE (2015)). A avaliação foi realizada em ambiente simulado. Nesse ambiente CodeRAnts obteve, em média, uma Precisão de 0,53 e Cobertura de 0,71.

Hipikat é uma ferramenta que recomenda artefatos relevantes para a realização de uma nova tarefa de desenvolvimento de software (ČUBRANIĆ *et al.*, 2005b). Hipikat utiliza um grande número de documentos disponíveis no projeto, tais como, código fonte, documentação, comunicações entre os desenvolvedores (e-mail, fóruns de discussão), relatório de erros e planos de teste. A proposta faz recomendações a partir de similaridades textuais, por ligações e inferências de ligações entre os artefatos. Hipikat foi avaliado em um estudo e obteve uma precisão de 11%. THOMPSON e MURPHY (2014) fazem uso de similaridade de tarefas e de artefatos relacionados às tarefas para inferir um contexto inicial para uma nova tarefa de programação. O algoritmo proposto por eles para inferir o contexto inicial sobre os dados do Projeto Mylyn obteve uma

precisão de 21%. Este trabalho é uma evolução do Hipikat, mas ainda apresenta um indício de precisão muito baixo.

A proposta de YE *et al.* (2014) lista uma classificação de artefatos relevantes para a correção de um erro considerando as informações do histórico do projeto. A lista é construída com base em um modelo que soma as medidas entre o erro e o código considerando seis características, sendo que cada medida tem um peso calculado de acordo com o histórico do projeto. A primeira característica usa cosseno de similaridade textual entre o erro e o código; e entre o erro e os métodos (procedimentos do código). A segunda mede a similaridade do erro com as documentações do código e método. A terceira mede a similaridade entre o erro reportado e os erros corrigidos relacionados com um código (artefato). A quarta mede a similaridade considerando somente o nome do artefato (classe). A quinta considera que um código alterado recentemente por uma correção de erro tem maior probabilidade de estar envolvido nas próximas correções e, finalmente, a sexta mede a frequência com que um artefato é alterado em função da correção de erros. A avaliação indica que, em média, a proposta pode recomendar artefatos relevantes (*top* 10) em 70% das recomendações.

MORAES *et al.* (2010) propuseram um sistema para, dada uma mensagem textual, escolher o especialista em artefatos com base no histórico do código fonte, da documentação (JavaDoc) e das mensagens (e-mail) trocadas entre os desenvolvedores. O sistema aplica uma função de similaridade entre a mensagem textual informada e a documentação do projeto, permitindo que o pacote mais similar seja selecionado. Em seguida, verifica no histórico de mensagens os desenvolvedores que mais mencionam o pacote selecionado, criando uma pontuação sobre as mensagens. Depois, é gerada uma lista de classificação de desenvolvedores de acordo com a pontuação dos *commits* sobre as classes do pacote selecionado e de cada classe que depende dele. O sistema então envia a mensagem para o desenvolvedor que obtiver um maior valor na soma das pontuações. A avaliação mostrou que as mensagens trocadas entre desenvolvedores melhoram a eficácia do sistema para a escolha de especialista.

FRITZ *et al.* (2014) propuseram o DOK (Degree Of Knowledge), um modelo para inferir o conhecimento dos desenvolvedores sobre artefatos. Esse modelo é baseado na autoria dos artefatos combinado com o modelo DOI (KERSTEN, 2007). O DOK não considera explicitamente o esquecimento e a reaprendizagem dos desenvolvedores, mas herda do DOI a modelagem do interesse e desinteresse dos desenvolvedores por artefatos.

Então podemos dizer que o DOK modela implicitamente esquecimento e reaprendizagem. A avaliação do DOK mostrou que esse modelo produz bons resultados para recomendar especialistas e identificar interesses sobre mudanças no código, mas não se mostrou adequado para medir o conhecimento sobre APIs.

SUN *et al.* (2017) propõem o método EDR_SI para recomendar artefatos e desenvolvedores para novas tarefas de correção de erros ou alteração de software. O EDR_SI recomenda baseando-se nas especialidades e nos hábitos dos desenvolvedores. Primeiro, a nova tarefa e o histórico de *commits* são processados com técnicas de PLN. Depois, selecionam-se os *commits* mais similares à nova tarefa para extrair os seus autores e os artefatos relacionados. Em seguida, produz uma lista de artefatos e artefatos similares personalizada por desenvolvedor baseando-se também no histórico de edições de cada desenvolvedor. Além disso, EDR_SI recomenda informações adicionais para compor o contexto de realização de novas tarefas: histórico de edições dos artefatos, rede de desenvolvedores e uma lista de artefatos personalizados por cada desenvolvedor. A avaliação indica que EDR_SI pode melhorar a precisão da recomendação de desenvolvedores em comparação com trabalhos mais recentes.

Com relação ao problema de recomendar desenvolvedores que estão mais preparados para realizar uma determinada correção de erro, XIA *et al.* (2013) propuseram o método DevRec. Primeiro, DevRec realiza uma análise dos erros reportados para identificar os vizinhos mais próximos a uma nova correção de erro. Para isso, utiliza os termos das descrições dos erros (técnica PLN), os tópicos aos quais pertencem (técnica LDA), o produto e componente afetados por cada erro. Os desenvolvedores que participaram da correção dos vizinhos mais próximos têm probabilidades de também realizarem o novo erro. Segundo, realiza uma análise sobre o histórico de erros realizados para produzir a pontuação de cada desenvolvedor sobre os erros realizados de acordo com a sua afinidade nos termos, tópicos, componentes e produtos. DevRec então recomenda desenvolvedores para uma nova correção de erro combinando as probabilidades com as pontuações dos desenvolvedores. Na avaliação, DevRec demonstrou uma melhor performance comparado a outros métodos (XIA *et al.*, 2015).

XIE *et al.* (2012) propuseram o sistema de recomendação DRETOM para recomendar uma lista de desenvolvedores que, possivelmente, têm interesse e especialização para participar da resolução de um erro reportado. DRETOM aplica a técnica de classificação de documentos em tópicos sobre o histórico de erros corrigidos

para modelar o interesse e especialização dos desenvolvedores. Esse sistema associa um erro corrigido a somente um tópico, o de mais relevância para o erro. Então, os tópicos encontrados no histórico de erros formam grupos de erros pertencentes a cada tópico. Os desenvolvedores que participaram da correção de erros de cada grupo indicam, portanto, um interesse e especialização no tópico. DRETOM calcula a probabilidade de um desenvolvedor participar de um erro reportado multiplicando a probabilidade do erro pertencer a um tópico, pela probabilidade do desenvolvedor ser um candidato do mesmo tópico. A avaliação mostrou evidências de que o sistema recomenda desenvolvedores com uma Cobertura média de 82% em listas com 5 desenvolvedores.

O desenvolvimento de software com apoio da nuvem (*crowdsourcing*) faz surgir novos relacionamentos de trabalho. Por exemplo, na plataforma Topcoder[33], é possível publicar uma tarefa e oferecer um retorno financeiro pela sua realização. Nesse cenário, onde é necessário que o desenvolvedor tenha diversas habilidades para realizar determinada tarefa, recomendar tarefa para desenvolvedores capazes é um requisito para o sucesso desta forma de trabalho. Nessa direção, ZHU *et al.* (2016) recomendam uma lista de classificação de desenvolvedores para cada nova tarefa. O algoritmo primeiro extrai informações dos desenvolvedores e das tarefas, por exemplo, as habilidades técnicas (desenvolvedor tem conhecimento sobre HTML5), ou localização, quando há a informação de que um desenvolvedor de um local seja mais adequado para realizar a tarefa. Essas informações são modeladas em grupos de características que combinadas formam uma lista de classificação de desenvolvedores para cada tarefa.

O trabalho de ZHANG *et al.* (2016) está no escopo de manutenção de software, especificamente na classificação dos erros reportados e na indicação de desenvolvedores para corrigi-los. A seleção de erros similares ao novo erro é realizada utilizando uma medida de similaridade baseada nas características dos erros. Depois, com o novo erro e com os seus erros mais similares, a proposta recomenda uma lista de classificação de desenvolvedores baseando-se no cálculo dos índices social e de experiência de cada desenvolvedor. A avaliação mostrou que a proposta recomenda satisfatoriamente desenvolvedores para a correção de um novo erro reportado, superando a proposta de XIA *et al.* (2013).

[33] https://www.topcoder.com/

WANG *et al.* (2017) propuseram um sistema para recomendar desenvolvedores para plataformas *crowdsourcing* utilizando clusterização e modelo de aprendizagem. As tarefas são classificadas em *clusters* e para cada par desenvolvedor-*cluster* é criada uma curva de aprendizagem. Uma nova tarefa é relacionada ao cluster que for mais similar à nova tarefa considerando, por exemplo, informações técnicas. E para o *cluster* mais similar à nova tarefa é construída uma lista com os desenvolvedores mais capazes. Então, esses desenvolvedores são recomendados para realizar a nova tarefa.

MALHEIROS *et al.* (2012) construíram um sistema de recomendação, denominado Mentor, para ajudar os novatos na realização de tarefas que requisitam mudanças no software, por exemplo, correção de erros. Para uma nova requisição de mudança, Mentor recomenda solicitações de mudanças similares já realizadas anteriormente utilizando a técnica PPM (SALOMON e MOTTA, 2010). Além disso, Mentor recomenda os artefatos da solicitação de mudança mais similares à nova solicitação de mudança. A avaliação indica que a técnica PPM é superior à técnica LSI (ČUBRANIĆ *et al.*, 2005b) para recomendar tarefas similares.

E o último trabalho, DebugAdvisor é um sistema de recomendação para apoiar o desenvolvedor na solução de um erro reportado (ASHOK *et al.*, 2009). A primeira fase do sistema organiza as descrições dos erros em um conjunto de palavras (utilizando a técnica PLN) e o texto de procura informado pelo desenvolver também gera um conjunto de palavras. Então, DebugAdvisor retorna um conjunto de erros que estão próximos do texto de procura. Na segunda fase, DebugAdvisor produz *links* entre os erros retornados na primeira fase com o código (linhas, funções) e desenvolvedores relacionados, de acordo com as informações do sistema de controle de versão. A avaliação indica que a primeira fase do DebugAdvisor supera a procura utilizando todo o conteúdo do texto disponível. E na avaliação da segunda fase, na recomendação de desenvolvedores, constatou-se que os melhores resultados foram alcançados quando se utilizou os 5 erros mais relevantes resultantes da primeira fase.

Além desses trabalhos descritos acima, porém um pouco distante do trabalho desta tese, CANFORA *et al.* (2012) recomendam mentores para os novos membros de projetos de software, e GUO *et. al.* (2016) foca no aproveitamento de dados do histórico de outros projetos, no caso, o mais semelhante dentre eles, para serem utilizados como fonte de recomendação para o novo projeto.

Tabela 24- Comparação com os trabalhos relacionados.

	Técnica		Modelo			Mede Conhecimento				Recomenda Contexto				
												Especialistas		
	Cluster	PLN	Sim. por Interação	Esquec.	Reaprend.	Artefato □	Tarefa Δ	Δs Similares	Projeto	□	Δs Sim.	□	Δ	Δs Sim.
ALMHANA *et al.* (2016)		X								X				
ANDRIC *et al.* (2004)		X								X				
ANTUNES *et al.* (2012)		X								X				
CAICEDO e DUARTE (2015)		X								X				
ČUBRANIĆ *et al.* (2005)		X								X				
YE *et al.* (2014)		X								X				
MORAES *et al.* (2010)		X				X						X		
FRITZ *et al.* (2014)						**X**						X		
SUN *et al.* (2017)		X				X	X			X		X	X	
XIA *et al.* (2013)		X					**X**						X	
XIE *et al.* (2012)		X					X						X	
ZHU *et al.* (2016)		X					X						X	
ZHANG *et al.* (2016)		X					X	X	X				X	
WANG *et al.* (2017)	X						X	X					X	X
MALHEIROS *et al.* (2012)		X								X	X			
ASHOK *et al.* (2009)		X								X	X		X	X
Tacin: Capítulos 3 e 4	X	X	X	X	X	X	X	X	X	X	X	X	X	X

6.2 Análise dos Trabalhos Relacionados

Uma vez apresentados os trabalhos relacionados, é possível agora analisá-los de acordo com as características da Tabela 24. Somente um trabalho utilizou a técnica de clusterização. Por definição, como foi apresentado no Capítulo 2, clusterização organiza elementos mais semelhantes nos mesmos grupos, enquanto elementos menos semelhantes, ou não semelhantes, ficam em grupos diferentes. WANG *et al.* (2017) faz uso de clusterização utilizando o algoritmo *K-Means* para agrupar tarefas semelhantes, com o objetivo de medir o desempenho dos desenvolvedores através do modelo de aprendizado aplicado a cada grupo de tarefas resultante da clusterização. Tacin utiliza a clusterização para organizar em grupos as tarefas que estão relacionadas por interação Desenvolvedor-Artefato nos mesmos artefatos, como apresentado nos Capítulos 3 e 4. Então, é possível que uma nova tarefa tenha mais semelhança com determinados grupos, e por isso, há também uma expectativa de que os artefatos das tarefas do(s) grupo(s) mais semelhante(s) possam ser fonte de recomendação para serem editados pela nova tarefa. Além disso, Tacin pode medir o conhecimento dos desenvolvedores sobre cada grupo de tarefas. Conclui-se que Tacin faz uso de clusterização de forma diferente de WANG *et al.* (2017).

A Linguagem Natural (LN) está presente ao longo de todas as fases do processo de desenvolvimento de software por ser ainda um trabalho realizado por seres humanos. Por exemplo, LN é utilizada no levantamento de requisitos, nos modelos, nas documentações, no código fonte, nos manuais, nas linguagens de programação genéricas (Java, Python etc) e específicas de domínio, tais como OCL[34] e ATL (JOUAULT e KURTEV, 2006). Então, nos históricos dos projetos de software existe uma enorme quantidade de informações expressas em LN, por isso as técnicas de PLN são largamente utilizadas por pesquisadores. As técnicas de PLN também são utilizadas na maioria dos trabalhos relacionados desta tese. Por exemplo, ALMHANA *et al.* (2016) e YE *et al.* (2014) utilizam similaridade textual calculada através de cosseno. LDA (*Latent Dirichlet Allocation*) é aplicada por XIA *et al.* (2013), XIE *et al*. (2012), ZHANG *et al.* (2016) e ZHU *et al.* (2016). LSI (*Latent Semantic Indexing*) é utilizada por ČUBRANIĆ *et al.* (2005a) e GUO *et al.* (2016). ANDRIC *et al.* (2004) e YE *et al.* (2014) utilizam TF-IDF. CAICEDO e DUARTE (2015) utilizam a técnica de medir a distância entre dois termos

[34] https://www.omg.org/spec/OCL/

de acordo com o número de edições (alteração, remoção, ou inclusão de caractere) que é preciso fazer em um termo para chegar no outro. E Tacin faz uso de cosseno de similaridade, analisando os resultados sobre TO, TF e TF-IDF.

A revisão da literatura (Apêndice A) não identificou nenhum trabalho relacionado que meça a similaridade entre as tarefas utilizando o histórico das interações de edição realizadas pelos desenvolvedores. Além disso, os trabalhos não consideraram explicitamente os modelos das características humanas de esquecimento e reaprendizagem nas recomendações. Tacin explora um modelo de esquecimento e reaprendizagem sobre artefatos e sobre tarefas similares por interação para medir o conhecimento dos desenvolvedores sobre artefatos, tarefas similares e todo o projeto de desenvolvimento de software, ao longo do tempo.

Com relação ao objetivo de medir conhecimento, o trabalho de MORAES *et al.* (2010) propõe um modelo para identificar especialista (*expert*) em parte do código. FRITZ *et al.* (2014) propõe um modelo para medir o conhecimento dos desenvolvedores sobre o código fonte. FRITZ *et al.* explicitamente utilizam a palavra *knowledge*. SUN *et al.* (2017) propõem um modelo para recomendar especialistas baseando-se também na identificação das especializações dos desenvolvedores registradas no histórico do projeto. XIA *et al.* (2013) modelam a afinidade (*affinity*) de um desenvolvedor com relação a um erro reportado. XIE *et al.* (2012) utilizam um modelo para inferir o interesse e a especialização (*expertise*) do desenvolvedor na resolução de um erro reportado. ZHU *et al.* (2016) medem a compatibilidade (*compatibility*) entre tarefas e desenvolvedores. ZHANG *et al.* (2016) propõem um modelo para medir a experiência (*experience*) do desenvolvedor no trabalho de corrigir erros. WANG *et al.* (2017) modelam a habilidade (*skill*) do desenvolvedor baseada em curva de aprendizado. Independente do termo utilizado, não foi encontrado nenhum trabalho com o objetivo de medir conhecimento sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software a partir das interações dos desenvolvedores e considerando a característica humana de esquecimento.

E a última característica de observação da Tabela 24, comparando os trabalhos relacionados com relação ao que recomendam, alguns recomendam exclusivamente artefatos (ALMHANA *et al.*, 2016, ANDRIC *et al*, 2004, ANTUNES *et al*, 2012, CAICEDO e DUARTE, 2015, ČUBRANIĆ *et al.*, 2005b, YE *et al*, 2014), outros recomendam exclusivamente especialistas em tarefa (XIA *et al.*, 2015, XIE *et al.*, 2012,

ZHANG *et al.*, 2016, ZHU *et al.*, 2016), dois trabalhos recomendam exclusivamente especialistas em artefatos (FRITZ *et al.*, 2014, MORAES *et al.*, 2010). SUN et al. (2017) recomendam artefatos e também especialistas em artefatos e tarefas. WANG *et al.* (2017) recomendam especialistas em tarefa e tarefas similares. MALHEIROS *et al.* (2012) recomendam artefatos e tarefas similares e ASHOK *et al.* (2009) recomendam artefatos, tarefas similares, especialista em tarefa e tarefas similares. Portanto, este último é o trabalho mais próximo do Tacin com relação ao objetivo de recomendação.

6.3 Considerações Finais

Este capítulo apresenta os trabalhos relacionados selecionados a partir de uma quasi-revisão sistemática apresentada com detalhes no Apêndice A. O resultado da revisão selecionou 16 artigos primários que apresentaram uma ou mais técnicas para recomendar contexto em projetos de desenvolvimento de software.

O Tacin foi comparado com os trabalhos relacionados evidenciando: 1- a utilização das técnicas de clusterização e processamento de linguagem natural; 2- a modelagem de similaridade por interação; 3- se considera esquecimento humano e reaprendizagem; 4- se propõe medir conhecimento sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software e; 5- se o trabalho recomenda artefatos, tarefas similares e também especialistas em artefatos, tarefas e tarefas similares.

O resultado da comparação do Tacin com os trabalhos relacionados apontou que: somente um trabalho utilizou a técnica de clusterização; as técnicas de processamento de linguagem natural são largamente utilizadas por pesquisadores; não foi identificado nenhum trabalho relacionado que meça a similaridade entre as tarefas utilizando o histórico das interações de edição realizadas pelos desenvolvedores; os trabalhos não consideraram explicitamente os modelos das características humanas de esquecimento e reaprendizagem nas recomendações e; somente um trabalho utiliza explicitamente a palavra conhecimento (*knowledge*). Além disso, nenhum dos trabalhos relacionados atendeu a todas as características de comparação.

Finalmente, a comparação dos trabalhos relacionados entre si e com o Tacin, com relação à eficácia de recomendação, é difícil porque cada proposta faz uso de diferentes dados, utiliza diferentes métricas, e não há um protocolo científico obrigatório a ser seguido para a avaliação de Sistemas de Recomendação na Engenharia de Software. Então, uma solução é buscar pesquisadores interessados em implementar os diversos modelos em uma ferramenta e realizar estudos experimentais em laboratório e em

empresas. Dessa forma, os resultados quantitativos resultantes de um mesmo conjunto de métricas poderão refletir realmente uma classificação dos trabalhos de acordo com a eficácia de recomendação.

7 - Discussão e Resultados

Este capítulo inicia evidenciando que o Tacin explora a similaridade de tarefas por interação e mostra que a avaliação da proposta não eliminou as imperfeições dos dados coletados. O capítulo segue com uma discussão sobre os modelos do Tacin para perceber conhecimento, Seção 7.1, e com o método Tacin para recomendar contexto de nova tarefa, 7.2. O capítulo finaliza respondendo à subquestão de pesquisa e também à questão de pesquisa desta tese.

As técnicas de similaridade textual são amplamente utilizadas para medir a similaridade de documentos, de modo que também são usadas para medir a similaridade entre tarefas de desenvolvimento de software (HOLMES *et al.*, 2005, YE e FISCHER, 2005). A unidade principal dessas técnicas depende dos padrões dos idiomas, tais como do Alemão, Inglês e Português e assim por diante. O Tacin utiliza similaridade de tarefa com base em interações de edição dos desenvolvedores sobre artefatos. Até onde se tem conhecimento, esta proposta é a primeira que faz uso da similaridade por interação de edição para atribuir grau de conhecimento aos desenvolvedores com o intuito de inferir o contexto de uma nova tarefa de desenvolvimento de software.

A avaliação do Tacin foi realizada em um conjunto de dados de cinco projetos de código aberto. Uma revisão da literatura foi realizada para verificar se o conjunto de dados coletados corresponde aos dados dos projetos. No entanto, não há garantia de que os dados de todas as tarefas realizadas foram realmente registrados no histórico do projeto e também não podemos avaliar o impacto dessas imperfeições. Portanto, não removemos os valores discrepantes e os erros de consistência, como no caso do desenvolvedor dev_1512 que executou 6 tarefas, mas registrou apenas uma interação, no projeto Mylyn Docs.

A seguir, apresenta-se a discussão sobre os modelos do Tacin e depois sobre o método do Tacin. O Capítulo finaliza mostrando os principais resultados da avaliação para embasar a resposta da subquestão de pesquisa **QP$_1$** e também da questão de pesquisa desta tese, **QP**.

7.1 Modelos Tacin

Os modelos para medir conhecimento do Tacin foram avaliados na perspectiva de identificar quem sabe mais sobre o projeto de desenvolvimento de software. O modelo K_p mede o conhecimento do desenvolvedor sobre todo o projeto com base no modelo K_c, que mede o conhecimento do desenvolvedor sobre um grupo de tarefas semelhantes por

interação Desenvolvedor-Artefato, e este o faz com base no modelo K_s, que mede o conhecimento do desenvolvedor sobre tarefa, que por sua vez utiliza o modelo K_a, que mede o conhecimento do desenvolvedor sobre artefato. Assim, os resultados da avaliação de K_p também representam uma avaliação dos modelos K_c, K_s e K_a.

A avaliação comparou K_p com a premissa de que um desenvolvedor que realizou uma tarefa, também adquiriu conhecimento sobre a mesma. Essa premissa está de acordo com FRITZ *et al.* (2014), que defende a autoria como uma informação relevante para medir conhecimento. A avaliação de K_p utilizou K_s com as interações entre Desenvolvedor-Artefato e não considerou as interações Desenvolvedor-Desenvolvedor por falta de dados. De acordo com o experimento de MORAES *et al.* (2010), as interações Desenvolvedor-Desenvolvedor também contribuem para a identificação de especialistas. Assim, é possível que a inclusão, futuramente, dos dados das interações Desenvolvedor-Desenvolvedor possa melhorar os resultados da avaliação dos modelos do Tacin.

Com relação à estrutura dos modelos, eles são baseados em informação com um grão muito fino, ou seja, no modelo K_a, uma interação de edição entre um desenvolvedor e um artefato cria um *link* entre esses elementos. Então, após a realização de uma tarefa, um desenvolvedor tende a ter várias associações com um mesmo artefato. E pelas dependências entre os modelos já citadas acima, essas associações também fazem parte dos outros modelos. Em um estudo anterior (LUCAS e OLIVEIRA, 2016) foi encontrada uma evidência de que essas micro associações podem contribuir para a melhoria da qualidade da informação. Naquele estudo, a colocação em 1º lugar do desenvolvedor que mais sabia sobre o projeto Mylyn foi atribuída à mesma pessoa considerando três diferentes perspectivas de observação, em março de 2016, segundo o número de erros corrigidos da lista oficial do projeto Mylyn, segundo o número de erros corrigidos dos dados coletados para uma avaliação preliminar do modelo K_p, e segundo o modelo K_p aplicado aos dados coletados. Nas três perspectivas, o desenvolvedor ocupou a primeira colocação com 2053 erros corrigidos na lista oficial, com 765 erros nos dados coletados e com o modelo K_p produzindo valor de 479285. E mais explicitamente, no estudo apresentado nesta tese, na primeira rodada do experimento, Tabela 10, o primeiro colocado também foi o mesmo, realizou 378 tarefas, com o total de 50984 interações (edição e *commit*s) e o valor final de K_p foi 1802,02.

No primeiro estudo desta tese, a linha de base (número de tarefas realizadas) representa o conhecimento do que foi feito porque ela indica quem fez qual tarefa,

atribuindo 1 ao desenvolvedor para cada tarefa realizada. O modelo K_p representa o conhecimento de como foi feito. Cada tarefa realizada está relacionada aos artefatos pelas interações Desenvolvedor-Artefato, e o valor do conhecimento atribuído ao desenvolvedor varia de acordo com o grau da interação nos artefatos ao longo do tempo. A linha base foi comparada com o número de interações realizadas no projeto. O grau de correlação médio foi de 68% com valores de *p.value* maiores que 3,5% e menores que 4,5% em 12 rodadas. No entanto, a classificação por interações também não leva em conta o esquecimento e o reaprendizado dos desenvolvedores. A comparação da linha de base com a classificação gerada pelo modelo K_c apresentou, em média, uma correlação de 70% com valores de *p.value* menores que 3,2% em 12 rodadas. A classificação pelo K_c apresentou uma correlação maior que a classificação por interações, além de herdar de K_s e K_a um modelo de esquecimento e reaprendizado dos desenvolvedores. A comparação da linha de base com K_p mostrou um melhor resultado em 12 rodadas, a média de correlação foi de 72%, com valores de *p.value* abaixo de 2% a partir da sexta rodada. Além disso, como o modelo K_p faz uso do modelo K_c, ele também herda a modelagem de esquecimento de K_s e K_a.

Não temos evidências para considerar as tarefas realizadas como uma medida precisa de conhecimento do desenvolvedor sobre todo o projeto de software. Como já dito anteriormente, é preciso ainda considerar outras informações, tal como a informação das interações Desenvolvedor-Desenvolvedor. No entanto, como já defendido, é uma informação que tende para a medida ideal. Diante do exposto, pode-se considerar que a correlação de K_p com tarefas realizadas é uma evidência de que os modelos de conhecimento do Tacin estão na direção de medir conhecimento de forma adequada.

O DOK, de FRITZ *et al.* (2014), é o trabalho relacionado mais próximo dos modelos de conhecimento do Tacin. Portanto, um candidato para ser uma base de comparação nas avaliações dos modelos. No entanto, a avaliação FRITZ *et al.* mostrou que o DOK não é adequado para medir conhecimento sobre APIs. Por isso, nesta tese, adotamos tarefas realizadas como a medida mais adequada para medir conhecimento sobre projeto de software. Além disso, atribuir conhecimento sobre uma tarefa em função da sua realização é bem aceito pela comunidade de software, e também, como já foi dito, é uma forma de autoria, informação que contribui para a medição de conhecimento.

Os modelos do Tacin não fazem uso da informação de autoria diretamente como faz o modelo DOK, mas o faz de uma forma indireta. O autor de um artefato, por definição

de autoria[35], é todo aquele que contribuiu com a sua criação. Os modelos do Tacin pontuam essas contribuições, indicando, portanto, também um grau de autoria do desenvolvedor sobre artefato, tarefa, tarefas similares, e todo o projeto. Por outro lado, o software cresce em tamanho ao longo do tempo, e para isso, os artefatos também crescem com a correção de erros e com a adição de novas funcionalidades. Então, pode-se entender que o autor de um artefato não é somente o desenvolvedor que o criou, mas todos aqueles que contribuíram para a sua atual versão. Nessa direção, os modelos do Tacin consideram a informação de autoria dos desenvolvedores e a utiliza de forma adequada para medir conhecimento.

A Percepção de Contexto, segundo OMORONYIA *et al.* (2010), baseia-se em informações sobre tarefas (história, tempo, duração e paralelismo). Nós defendemos a importância da identificação de quem sabe sobre o contexto. Então propomos a inclusão do elemento *Knowledge* na Percepção do Contexto com a descrição: "Quem sabe sobre o contexto?". Desta forma podemos dizer que os Modelos de Percepção apresentados neste artigo implementam o elemento *Knowledge* da Percepção de Contexto porque os Modelos podem informar quem sabe sobre os artefatos, tarefas realizadas, um conjunto de tarefas similares e sobre todo o projeto de software.

7.2 Método Tacin

Em muitas áreas do conhecimento humano, o que é planejado não é realizado na sua totalidade e em conformidade, ou simplesmente não é possível planejar tudo aquilo que será preciso realizar, por exemplo, no tratamento de um paciente (VAN DER AALST, 2009). O ato de descrever em texto uma nova tarefa para ser realizada é uma forma de planejar "o que" precisa ser realizado, mas não de planejar o "como" deve ser realizado. O contexto de uma tarefa está fortemente associado ao como foi feito e ao como deve ser feito. O como foi feito, nesta tese, fez a associação entre os artefatos e tarefas em função de quais artefatos realmente foram necessários alterar ou criar para realizar as tarefas. O como deve ser feito é um trabalho que faz uso de conhecimento intensivo, e no caso das

[35] Lei Brasileira 9610/98 sobre direitos autorais, mais especificamente: Art. 11, o autor é a pessoa física criadora de obra literária, artística ou científica. E , Art. 14, é titular de direitos de autor quem adapta, traduz, arranja ou orquestra obra caída no domínio público, não podendo opor-se a outra adaptação, arranjo, orquestração ou tradução, salvo se for cópia da sua.

tarefas de software, representa ainda um trabalho realizado por humanos, mas que pode e deve ser apoiado por ferramentas para diminuir erros e melhorar a produtividade.

O método Tacin para inferir o contexto de uma nova tarefa, ou seja, indicar o como a tarefa deve ser realizada, passa pelo desafio de associar a nova tarefa, que somente possui uma descrição textual de "o que" deve ser realizado, com as suas tarefas similares. Trabalhos relacionados descobrem as tarefas similares utilizando a descrição textual, ou seja, considerando informações do que precisa ser feito. O Tacin, primeiro, agrupa as tarefas considerando as informações de como elas foram realmente realizadas, isto é, tarefas que editaram os mesmos artefatos tendem a ser similares. Esse agrupamento, relaciona uma tarefa a um artefato quando o desenvolvedor realizou uma edição no artefato enquanto realizava a tarefa. Uma melhoria nesta associação pode considerar o grau de interação do desenvolvedor sobre o artefato.

A avaliação dos agrupamentos de tarefas por interação Desenvolvedor-Artefato mostrou a existência de pelo menos um grupo que continha artefatos que seriam editados pelas novas tarefas. Nessa avaliação, o grupo que tinha a melhor cobertura de artefatos para cada nova tarefa foi escolhido para recomendar artefatos. Em média, o grupo com a melhor cobertura continha somente 6% dos artefatos disponíveis no projeto, mas contendo uma cobertura média de 77%. Os números indicam que o agrupamento de tarefas por interação, do como as tarefas foram realizadas, produz contexto adequado para novas tarefas, que possuem somente a indicação do que precisa ser realizado.

O Tacin propõe a associação entre a nova tarefa e os grupos de tarefas similares por interação considerando a parte textual da descrição, isto é, da associação em função do que deve ser realizado. Essa associação explora a existência de correlação entre similaridade por interação e textual. Para a associação realizada com técnicas de processamento de texto, a avaliação constou que, em média, a recomendação de artefatos para novas tarefas, teve uma Cobertura de 52% e 90% de Redução, considerando os 3 grupos mais similares à nova tarefa.

Os números indicam que dentro de um projeto de desenvolvimento de software, as tarefas do passado podem indicar uma tendência de futuro. No entanto, não é verdade que uma tarefa só pode ser realizada de uma forma. No contexto do desenvolvimento de software, uma tarefa pode ser realizada reutilizando os artefatos passados ou criando novos artefatos. Assim, é verdade que os desenvolvedores podem realizar tarefas criando

artefatos de forma desnecessária, como observado por BEGEL e ZIMMERMANN (2014).

7.3 Resultados

Os resultados da avaliação dos modelos de percepção do conhecimento indicam que o modelo K_p possui forte correlação com a classificação por tarefas realizadas. O modelo K_p considera a complexidade da tarefa, a similaridade das tarefas por interação, a depreciação do conhecimento e a distribuição do conhecimento em um projeto de desenvolvimento de software. Além disso, o modelo K_p é um modelo que faz uso dos modelos K_c, K_s e K_a, o que também indica que esses modelos são apropriados para medir conhecimento sobre tarefas similares, tarefa e artefato, respectivamente. Logo, a avaliação dos modelos do Tacin mostrou evidências de que o histórico das interações dos desenvolvedores, de um projeto de desenvolvimento de software, considerando a modelagem do esquecimento humano, são suficientes para produzir medições de conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de software. Respondendo, portanto, de forma positiva à subquestão de pesquisa **QP$_1$**.

> *Os dados do histórico das interações, considerando a modelagem do esquecimento humano, são suficientes para produzir medições de conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software.*

Os resultados das avaliações do método Tacin para recomendar contexto de nova tarefa indicam que a organização das tarefas do histórico do projeto em grupos similares por interação em artefatos contribui para a recomendação de artefatos e desenvolvedores para novas tarefas. Na associação da nova tarefa com os grupos similares através de processamento de linguagem natural, em média obteve uma Cobertura de 52%, quando a Cobertura média máxima obtida foi de 77%. A avaliação do Tacin na recomendação de artefatos e desenvolvedores, em comparação ao algoritmo básico de recomendação (KNN com $k = 1$) mostrou a superioridade do Tacin em quatro projetos.

Conclui-se que ainda há espaço para melhorias. No entanto, a avaliação do método do Tacin mostrou evidências de que a organização do histórico de elementos do projeto, em função da similaridade por interação Desenvolvedor-Artefato, combinada com a similaridade textual, é suficiente para produzir recomendação de contexto de nova tarefa

em projeto de desenvolvimento de software. Portanto, a resposta também é positiva para a questão de pesquisa **QP**.

> *A organização do histórico de elementos do projeto, em função da similaridade por interação dos desenvolvedores sobre artefatos, combinada com a similaridade textual, é suficiente para produzir recomendação de contexto de nova tarefa em projeto de desenvolvimento de software.*

Em resumo, o resultado dos estudos apresentados nesta tese corrobora com a existência de correlação entre variáveis de contexto e textual (MAALEJ *et al.*, 2017).

7.4 Considerações Finais

Este capítulo apresentou a discussão sobre o Tacin e respondeu de forma positiva à subquestão de pesquisa QP1 e à questão de pesquisa QP.

O Tacin defende que o desenvolvedor que realizou uma tarefa, também adquiriu conhecimento sobre ela. Portanto, quem mais realizou tarefas em um projeto, mais sabe sobre ele. Logo, na avaliação dos modelos, é válido comparar a lista de desenvolvedores segundo o modelo K_p com a lista de desenvolvedores que mais realizaram tarefas.

A avaliação do método mostrou que a clusterização das tarefas por interações Desenvolvedor-Artefato produz grupos de tarefas similares adequados para o contexto de novas tarefas. No entanto, o processo de encontrar o melhor grupo para uma nova tarefa, com base na similaridade textual, ainda necessita de melhorias.

8 - Conclusão e Trabalhos Futuros

Este capítulo resume a solução desta tese, passando pela sua descrição, avaliação e comparação com os trabalhos relacionados. Lista alguns possíveis trabalhos futuros e mostra como a solução também pode ser aplicada em outros domínios. E finaliza concluindo que a solução apresentada, denominada de Tacin, atende ao objetivo desta tese.

A presente tese apresentou os modelos para perceber o conhecimento dos desenvolvedores de software e um método para recomendar artefatos, tarefas similares e especialistas para compor o contexto de uma nova tarefa. Os modelos e o método são baseados nas interações dos desenvolvedores sobre artefatos e entre eles, por isso denominados como modelos e método do Tacin (***Ta****sk* ***C****ontext based on* ***In****teractions,* Contexto de Tarefa baseado em Interações). Os modelos de conhecimento consideram a modelagem do esquecimento humano enquanto o Método utiliza a similaridade de tarefas, observando as interações Desenvolvedor-Artefato, e explora a correlação entre similaridade textual e similaridade por interação.

O Tacin foi avaliado com dados de cinco projetos de software livre. O modelo do Tacin, para listar quem mais sabe sobre o projeto de desenvolvimento de software, mostrou uma forte correlação com a classificação dos desenvolvedores por tarefas realizadas. Os modelos do Tacin atuam nas informações "do como" as tarefas foram realizadas enquanto a lista de tarefas realizadas considera somente "o que" foi realizado. Então, conclui-se que os modelos de conhecimento do Tacin avançam no conhecimento "do como" as tarefas foram realizadas.

A avaliação do método do Tacin para recomendar contexto de nova tarefa mostrou, que em média, uma recomendação consegue reduzir o conjunto de artefatos do projeto em 90%, mantendo 52% dos artefatos que precisam ser editados para a realização das novas tarefas. No entanto, a avaliação também mostrou que a organização do histórico de tarefas por interação produziu uma organização adequada de contexto, identificando 77% dos artefatos que serão editados. Isso indica que ainda é preciso investir esforços na melhoria da associação das novas tarefas com os grupos de tarefas similares por interação. A avaliação também mostrou que o Tacin é superior na recomendação de artefatos e desenvolvedores quando comparado ao algoritmo básico de recomendação.

O Tacin foi comparado com trabalhos que têm como objetivo recomendar artefatos, tarefas e/ou desenvolvedores em projetos de desenvolvimento de software. A comparação verificou se os trabalhos fazem uso das técnicas de clusterização,

processamento de linguagem natural, se modelam ou não similaridade por interação, esquecimento humano e reaprendizagem. Além de verificar se medem conhecimento sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software.

A comparação também mostra se os trabalhos recomendam artefatos, tarefas similares e especialistas para o contexto de nova tarefa. O resultado mostrou que o Tacin é o único trabalho que combina o uso das técnicas de clusterização, processamento de linguagem natural, similaridade por interação, modelagem do esquecimento humano e reaprendizagem com o propósito de medir o conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software; e também recomendar artefatos, tarefas similares e especialistas para compor o contexto de uma nova tarefa.

A avaliação com os dados de projetos de desenvolvimento de software e a comparação com os trabalhos relacionados mostram que o Tacin pode contribuir com informações relevantes para apoiar os desenvolvedores na realização de novas tarefas. No entanto, ainda existe espaço para melhorias e futuras avaliações. Nesse sentido, segue uma lista de possíveis trabalhos futuros:

1. Implementar o Tacin como um Sistema de Recomendação para a Engenharia de Software;
2. Avaliar o comportamento do Tacin dentro de um projeto de desenvolvimento de software em ambiente acadêmico;
3. Avaliar o comportamento do Tacin dentro de um projeto de desenvolvimento de software na indústria;
4. Confrontar os resultados das avaliações 2 e 3 acima com as avaliações realizadas nesta tese (*offline*). Cabe ressaltar que em uma aplicação de recomendação (*online*), o usuário pode ser afetado por inúmeros fatores, tais como sociais, emocionais, recompensas, facilidade de utilização da ferramenta de recomendação etc;
5. A avaliação do método Tacin mostrou que a medida de similaridade textual entre a nova tarefa e os grupos de tarefas semelhantes por interação precisa de melhorias. Uma alternativa é verificar o *dw-cosine* (LI e HAN, 2013), uma extensão do cosseno de similaridade, enquanto outra opção seria verificar as técnicas textuais para textos pequenos e com erros gramaticais (O'SHEA *et al.*, 2008);

6. O Tacin pode ser visto como uma ferramenta para apoiar um processo intensivo em conhecimento para produzir software. Processos intensivos em conhecimento são afetados por regras e restrições. Logo, é preciso investigar como regras e restrições podem melhorar a recomendação do contexto de tarefa;
7. Os dados do projeto de desenvolvimento de software utilizado para a avaliação do Tacin não contêm as medidas de interações entre desenvolvedores sobre a realização das tarefas. É preciso modelar essas interações e considerá-las nas avaliações 2 e 3 descritas acima;
8. As questões pessoais (perfil, emoções etc) do desenvolvedor podem afetá-lo nas suas interações com os artefatos e com os outros desenvolvedores. É preciso investigar a influência dessas questões nos modelos apresentados para perceber o conhecimento dos desenvolvedores;
9. A modelagem de reaprendizagem somente considera os últimos 7 dias. Nada podemos afirmar se o oitavo, nono e mais dias são relevantes. Estudos experimentais precisam avaliar esta questão para verificar a necessidade de considerar mais dias na modelagem de reaprendizagem;
10. O agrupamento de tarefas semelhantes por interação relaciona uma tarefa a um artefato quando o desenvolvedor realizou uma edição no artefato enquanto realizava a tarefa. Uma melhoria nessa relação pode considerar também o grau de interação do desenvolvedor sobre o artefato;
11. Os modelos de percepção do conhecimento do Tacin medem o conhecimento dos desenvolvedores até uma data-hora, e podem evoluir para aceitar também um intervalo de tempo (por exemplo, hoje, nesta semana ou nos últimos sete dias). Dessa forma, os modelos passariam a medir um grau de proximidade do desenvolvedor com o artefato, tarefa e tarefas similares.

O Tacin apresenta algumas limitações. Os modelos não produzem uma medida precisa para medir o conhecimento dos desenvolvedores. Isso ocorre, principalmente, por dois motivos. O primeiro é porque não há um consenso na literatura sobre a curva ideal para modelar conhecimento com esquecimento. O segundo motivo é porque o coeficiente de esquecimento foi definido de acordo com a média de esquecimento entre os desenvolvedores sobre arquivos de código fonte, resultado de um estudo da literatura.

Outra limitação do Tacin é que a recomendação de artefatos e desenvolvedores é sensível à descrição textual das tarefas. Nos estudos realizados, foram encontradas tarefas descritas com somente duas palavras, apresentando, portanto, baixo significado textual. O Tacin necessita que as tarefas tenham uma descrição textual indicando o que precisa ser feito e, quando possível, a indicação da solução.

Os modelos de percepção do conhecimento apresentados também podem ser utilizados em outros domínios com algumas adaptações. Por exemplo, no ambiente Web, os usuários interagem com sites para atingir um objetivo. Nesse caso, as equações dos modelos de percepção poderiam ser adaptadas para estimar o interesse dos usuários em produtos, serviços ou assuntos. No cenário de cursos a distância, onde os alunos interagem com os sistemas e tutores para cumprir tarefas, a fim de adquirir conhecimento, há também a oportunidade de aplicar os modelos propostos. Uma possibilidade é estimar o grau de investimento dos alunos por disciplina e cruzar com o resultado das avaliações nas disciplinas.

O método para recomendar contexto para novas tarefas em projetos de desenvolvimento de software também pode ser utilizado em outros projetos. De forma geral, projetos são gerenciados com o apoio de sistemas informatizados, mesmo em casos que parte das tarefas ainda seja realizada produzindo artefatos ou serviços fora do ambiente virtual, casos do atendimento de emergência, na área médica, e da produção de peças, na indústria automotiva. Nesses cenários, as interações dos trabalhadores sobre os artefatos podem ajudar a estimar o quanto um trabalhador conhece um serviço. Por isso, também é possível recomendar contexto de uma nova tarefa de serviço utilizando o método de recomendar do Tacin.

Finalmente, os modelos de percepção do conhecimento do Tacin alcançam o objetivo específico desta tese: 1- medir o conhecimento dos desenvolvedores sobre artefato, tarefa, tarefas similares e todo o projeto de desenvolvimento de software. Enquanto o método do Tacin para recomendar contexto atende ao objetivo desta tese: 2- recomendar contexto para uma nova tarefa em projeto de desenvolvimento de software. Onde contexto de nova tarefa é formado por: 1- artefatos candidatos a serem editados pelos desenvolvedores na realização da nova tarefa; 2- desenvolvedores que têm potencial em apresentar conhecimento na nova tarefa; 3- tarefas realizadas similares à nova tarefa e; 4- por especialistas nas tarefas similares à nova tarefa.

Referências

ABRANTES, J. F., TRAVASSOS, G. H. "Common Agile Practices in Software Processes". In: **International Symposium on Empirical Software Engineering and Measurement**, pp. 355-358, Banff, AB, 2011.

ACTIVITI. Activiti. Disponível em: <http://www.activiti.org/>.

MYRÉN, A., **Evaluation of Machine Learning Algorithms in Recommender Systems**. Ph.D, University of Gothenburg, Gothenburg, Sweden, 2017.

ALMHANA, R. *et al.* "Recommending relevant classes for bug reports using multi-objective search". In: **Proceedings of The** *31st IEEE/ACM International Conference on Automated Software Engineering (ASE),* pp. 286-295, Singapore, 2016.

ANDRIC, M., HALL, W., CARR, L. "Assisting artifact retrieval in software engineering projects". In: **Proceedings of The 2004 ACM Symposium On Document Engineering**, pp. 48-50, New York, NY, USA, Oct. 2004.

ANTUNES, B., CORDEIRO, J., GOMES, P. "An Approach to Context-based Recommendation in Software Development". In: **Proceedings of the Sixth ACM Conference on Recommender Systems,** pp. 171-178, New York, NY, USA: ACM, 2012.

ARMOUR, P. G. "Beware of Counting LOC". **Commun. ACM**, v. 47, n. 3, pp. 21-24, Mar. 2004.

ASHOK, B. *et al.* "DebugAdvisor: A recommender system for debugging". In: *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering* **(ESEC/FSE '09),** pp. 373-382, ACM, New York, NY, USA, 2009.

AVAZPOUR, I. *et al.* "Dimensions and metrics for evaluating recommendation systems". In: **Robillard M., Maalej W., Walker R., Zimmermann T. (eds) Recommendation Systems in Software Engineering.** Springer, Berlin, Heidelberg. pp. 245-273, 2014.

BANI-SALAMEH, H., JEFFERY, C., AL-GHARAIBEH, J. "A Social Collaborative virtual environment for software development". **International Symposium on Collaborative Technologies and Systems,** Chicago, IL, Jun. 2010.

BARBOSA, C. E. *et al.* "Selecting Experts Using Data Quality Concepts". **International Journal of Database Management Systems**, v. 7, n. 6, pp. 01-15, Dec. 2015.

BASILI, V. R. **Software Modeling and Measurement: The Goal/Question/Metric Paradigm**. College Park, MD, USA: University of Maryland at College Park, 1992.

BELHAJJAME, K. *et al.*, (2013), **PROV-DM: The PROV Data Model**. W3C, In: Moreau, L., Missier, P. (eds), Disponível em: < http://www.w3.org/TR/2013/REC-prov-dm-20130430/ >. Acesso em: 1 jun. 2019.

BEST, D. J., ROBERTS, D. E. "Algorithm AS 89: The Upper Tail Probabilities of Spearman's Rho", **Journal of the Royal Statistical Society. Series C (Applied Statistics)**, v. 24, n. 3, pp. 377-379, 1975.

BJØRNER, D. "What is a method? - an essay on some aspects of domain engineering". In: *Programming methodology*, Annabelle McIver and Carroll Morgan (Eds.). Springer-Verlag New York, Inc., New York, NY, USA, pp. 175-203, 2003.

BOONGOEN, T., IAM-ON, N. "Cluster ensembles: A survey of approaches with recent extensions and applications", **Computer Science Review**, v. 28, pp. 1-25, 2018.

BPMN. **Business Process Model and Notation**. Disponível em: <http://www.bpmn.org/>.

CAICEDO, I., DUARTE, H. "CodeRAnts : A recommendation method based on collaborative searching and ant colonies , applied to reusing of open source code", **Ingenieria e Investigacion**, v. 34, n. 1, pp. 72-78, 2014.

CAMBRIA, E., WHITE, B. "Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]", **IEEE Computational Intelligence Magazine**, v. 9, n. 2, pp. 48–57, May 2014.

CANFORA, G. *et al.* "Who is Going to Mentor Newcomers in Open Source Projects? ". In: *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering* **(FSE '12)**, Article 44, 11 pages, ACM, New York, NY, USA, 2012.

CHESSA, A.; MURRE, J. "Spurious Power Laws of Learning and Forgetting: Mathematical and Computational Analyses of Averaging Artifacts", **Proceedings of the Annual Meeting of the Cognitive Science Society**, v. 31, n. 31, Jan. 2009.

CRUZ, M. D. **O problema de Clusterização Automática**. Tese de D.Sc., Rio de Janeiro, Brasil: COPPE/UFRJ, Jul. 2010.

ČUBRANIĆ, D. *et al.* "Hipikat: A project memory for software development", **IEEE Transactions on Software Engineering**, v. 31, n. 6, pp. 446-465, 2005.

DAHL, D. A. "Natural Language Processing: Past, Present and Future". In: **Mobile Speech and Advanced Natural Language Solutions,** Neustein A., Markowitz J. (eds), New York, NY: Springer, pp. 49-73, 2013.

DE O. BARROS, M. "An experimental evaluation of the importance of randomness in hill climbing searches applied to software engineering problems", **Empirical Software Engineering**, v. 19, n. 5, pp. 1423-1465, 2014.

DI CICCIO, C., MARRELLA, A., RUSSO, A. "Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches", **Journal on Data Semantics**, v. 4, n. 1, p. 29-57, Mar. 2015.

DOURISH, P., BELLOTTI, V. "Awareness and Coordination in Shared Workspaces". In **Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work. CSCW '92,** pp. 107-114, New York, NY, USA: ACM,1992.

EHRLICH, K., SHAMI, N. S. "Searching for Expertise". In **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**, pp. 1093-1096, New York, NY, USA: ACM, 2008.

ELLIS, C. A., GIBBS, S. J.; REIN, G. "Groupware: Some Issues and Experiences", **Commun. ACM**, v. 34, n. 1, pp. 39-58, Jan. 1991.

ERLIKH, L. "Leveraging legacy system dollars for e-business", **IT Professional**, v. 2, n. 3, pp. 17-23, May-June 2000.

FEILER, P. H., HUMPHREY, W. S. "Software process development and enactment: concepts and definitions". **Proceedings of the Second International Conference on the Software Process-Continuous Software Process Improvement**, pp. 28-40, Berlin, Germany, Feb. 1993.

FRAKES, W. B., KANG, K. "Software reuse research: status and future", **IEEE Transactions on Software Engineering**, v. 31, n. 7, pp. 529-536, July 2005.

FRITZ, T. *et al.* "A Degree-of-knowledge Model to Capture Source Code Familiarity". In: **Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering** - Volume 1. ICSE '10, pp. 385-394, New York, NY, USA: ACM, 2010.

FRITZ, T. *et al.* "Degree-of-knowledge: Modeling a developer's knowledge of code", **ACM Transactions on Software Engineering and Methodology**, v. 23, n. 2, March 2014.

FRITZ, T., MURPHY, G. C., HILL, E. "Does a Programmer's Activity Indicate Knowledge of Code?". In: *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering* **(ESEC-FSE '07)**, pp. 341-350, New York, NY, USA, Sep. 2007.

FUGGETTA, A. "Software Process: A Roadmap". In: *Proceedings of the Conference on The Future of Software Engineering* **(ICSE '00)**, pp. 25-34, ACM, New York, NY, USA, June 2000.

FUGGETTA, A., DI NITTO, E. "Software Process". In: **Proceedings of the on Future of Software Engineering**, pp. 1-12 , New York, NY, USA: ACM, May 2014.

FUKS, H. *et al.* "The 3C collaboration model". In: **Encyclopedia of E-Collaboration** , IGI Global, N. Kock (Ed.), pp. 637-644, 2007.

GASPARIC, M.; JANES, A. "What recommendation systems for software engineering recommend: A systematic literature review", **Journal of Systems and Software**, v. 113, pp. 101-113, 1 mar. 2016.

GOMES, T. L. *et al.* "Mining reuse processes". In: **Proceedings of the 17th Ibero-American Conference Software Engineering**, pp. 179-190, Pucon, Chile, April 2014.

GOSHTASBY, A. A. "Similarity and Dissimilarity Measures". In: **Image Registration. Advances in Computer Vision and Pattern Recognition**, Springer, London, pp. 7-66, 2012.

GREEN, S., MANNING, C. D. "Better Arabic Parsing: Baselines, Evaluations, and Analysis". **In:** *Proceedings of the 23rd International Conference on Computational*

Linguistics **(COLING '10). Association for Computational Linguistics**, pp. 394-402, Stroudsburg, PA, USA, Aug. 2010.

GUO, J. *et al.* "Cold-start Software Analytics". In: *Proceedings of the 13th International Conference on Mining Software Repositories* **(MSR '16)**, pp. 142-153, ACM, New York, NY, USA, May 2016.

GUTWIN, C., GREENBERG, S., ROSEMAN, M. "Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation". In: **Proceedings of HCI on People and Computers XI**, pp. 281-298
, London, UK, UK: Springer-Verlag, 1996.

HAN, J., KAMBER, M., PEI, J. "10 - Cluster Analysis: Basic Concepts and Methods". In: **Data Mining (Third Edition)**, **The Morgan Kaufmann Series in Data Management Systems**, Third Edition ed. HAN, J., KAMBER, M., PEI, J. (Eds.), Boston: Morgan Kaufmann, pp. 443-495, 2012.

HATTORI, L., LANZA, M. "Syde: A Tool for Collaborative Software Development". In: **Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering**, pp. 235-238, Cape Town, May 2010.

HATTORI, L. P., LANZA, M., ROBBES, R. "Refining Code Ownership with Synchronous Changes". **Empirical Softw. Engg.**, v. 17, n. 4-5, pp. 467-499, Ago. 2012.

HOLMES, R., WALKER, R. J., MURPHY, G. C. "Strathcona Example Recommendation Tool". In: *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering* **(ESEC/FSE-13)**, pp. 237-240, ACM, New York, NY, USA, Sept. 2005.

HUBBARD, D. W., **How to measure anything: Finding the value of "intangibles" in business**. Hoboken, NJ, US: John Wiley & Sons Inc, 2007.

HUNT, D. P. "The concept of knowledge and how to measure it", **Journal of Intellectual Capital**, v. 4, n. 1, pp. 100-113, 2003.

IEEE Standard for Software Project Management Plans. **IEEE Std 1058-1998**, pp. 1-28, Dec. 1998.

INZUNZA, S., JUÁREZ-RAMÍREZ, R., RAMÍREZ-NORIEGA, A. "User and Context Information in Context-Aware Recommender Systems: A Systematic Literature Review". In: **New Advances in Information Systems and Technologies**, vol. 444, **Advances in Intelligent Systems and Computing**, Springer International Publishing, pp. 649-658, 2016.

ISO/IEC/IEEE 12207:2017(E). ISO/IEC/IEEE International Standard - Systems and software engineering – Software life cycle processes. **ISO/IEC/IEEE 12207:2017(E) First edition 2017-11**, pp. 1-157, Nov. 2017.

JAIN, A. K., MURTY, M. N., FLYNN, P. J. "Data Clustering: A Review". **ACM Comput. Surv.**, v. 31, n. 3, pp. 264-323, Sep. 1999.

JALALI, S., WOHLIN, C. "Systematic literature studies: Database searches vs. backward snowballing". In: **Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement**, pp. 29-38, Lund, Sep. 2012.

JOUAULT, F., KURTEV, I. "Transforming Models with ATL". In: **Bruel JM. (eds) Satellite Events at the MoDELS 2005 Conference. MODELS 2005. Lecture Notes in Computer Science**, vol. 3844, pp 128-138, Springer Berlin Heidelberg, 2006.

JURISTO, N., MORENO, A. M., **Basics of Software Engineering Experimentation**. 1st. ed. Springer Publishing Company, Incorporated, 2010.

KERSTEN, M., 2007, **Focusing knowledge work with task context**. Ph.D. dissertation, University of British Columbia, Canada.

KOSKINEN, K. U., PIHLANTO, P., VANHARANTA, H. "Tacit knowledge acquisition and sharing in a project work context", **International Journal of Project Management**, v. 21, n. 4, pp. 281-290, May 2003.

KRÜGER, J. *et al.* "Do You Remember This Source Code? ". In: **Proceedings of the 40th International Conference on Software Engineering**. ICSE '18. New York, NY, USA: ACM, 2018.

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

LEE, M. D. "A Bayesian analysis of retention functions", **Journal of Mathematical Psychology**, v. 48, n. 5, pp. 310-321, Oct. 2004.

LETHBRIDGE, T. C., SIM, S. E., SINGER, J. "Studying Software Engineers: Data Collection Techniques for Software Field Studies", **Empirical Software Engineering**, v. 10, n. 3, pp. 311-341, July 2005.

LEVY, R., MANNING, C. "Is It Harder to Parse Chinese, or the Chinese Treebank?". In: **Proceedings of the 41st Annual Meeting on Association for Computational Linguistics**, v. 1, **Association for Computational Linguistics**, pp. 439-446, Stroudsburg, PA, USA, 2003.

LI, B.; HAN, L. "Distance Weighted Cosine Similarity Measure for Text Classification". In: **Intelligent Data Engineering and Automated Learning – IDEAL 2013**, **Lecture Notes in Computer Science**, vol. 8206, pp. 611-618, Springer, Berlin, Heidelberg, 2013.

LUCAS, E. *et al.* "A survey of languages to represent collaboration as a means of designing CSCW facilities in RDL". In: **Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design** *(CSCWD)*, pp. 55-60, Whistler, BC, Aug. 2013.

LUCAS, E. *et al.* "Investigating the collaborative support in CollabRDL: An analysis based on the 3C model". In: **Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2014**, pp. 529-534, Hsinchu, July 2014.

LUCAS, E. M. *et al.* "CollabRDL: A language to coordinate collaborative reuse", **Journal of Systems and Software**, v. 131, pp. 505-527, Sep. 2017.

LUCAS, E. M., OLIVEIRA, T. C., ALENCAR, P. S. C. "A Method to Recommend Artifacts to New Tasks in Software Projects". In: **Proceedings of the 31st International Conference on Software Engineering and Knowledge Engineering**, pp.489-492, Lisbon, Portugal, July 2019.

LUCAS, E. M., OLIVEIRA, T. C. "Um Modelo de Percepção para Indicar o Contexto de Novas Tarefas em Projetos de Software". In: **VII Congresso Brasileiro de Software: Teoria e Prática (CBSoft), VI Workshop de Teses e Dissertações do CBSoft**, pp. 10-18, Maringá, PR, Brasil, Set. 2016.

MAALEJ, W. **Intention-based Integration of Software Engineering Tools**. Verlag Dr. Hut, 2010.

MAALEJ, W., ELLMANN, M., ROBBES, R. "Using contexts similarity to predict relationships between tasks", **Journal of Systems and Software**, v. 128, pp. 267-284, 2017.

MALHEIROS, Y. *et al.* "A Source Code Recommender System to Support Newcomers". **2012 IEEE 36th Annual Computer Software and Applications Conference,** pp. 19-24, Izmir, Nov. 2012.

MALONE, T. W., CROWSTON, K. "The Interdisciplinary Study of Coordination". **ACM Comput. Surv.**, v. 26, n. 1, pp. 87-119, mar. 1994.

MANNING, C. D., RAGHAVAN, P., SCHÜTZE, H. "Scoring, term weighting, and the vector space model". In: **Introduction to Information Retrieval**. Cambridge: Cambridge University Press, pp. 100-123, 2008.

MANCORIDIS, S., MITCHELL, B.S., CHEN Y., GANSNER, E.R. "Bunch: A Clustering Tool for the Recovery and Maintenance of Software System Structures". In: **Proceedings of the IEEE International Conference on Software Maintenance**, pp. 50-59, Oxford, England,1999.

MCDONALD, D. W., ACKERMAN, M. S. "Expertise Recommender: A Flexible Recommendation System and Architecture". In: **Proceedings of the 2000 ACM conference on Computer supported cooperative work (CSCW '00)**, pp. 231-240,York, NY, USA: ACM, 2000.

MINELLI, R., MOCCI, A., LANZA, M. "I Know What You Did Last Summer - An Investigation of How Developers Spend Their Time". *2015 IEEE 23rd International Conference on Program Comprehension,* pp. 25-35*,* Florence, Sep. 2015.

MINER, G. *et al.* (EDS.), "Conceptual Foundations of Text Mining and Preprocessing Steps". In: **Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications,** Chapter 3, Boston: Academic Press, 2012.

MISTRÍK, I. *et al.* "Collaborative Software Engineering: Challenges and Prospects". In: MISTRÍK, I. *et al.* (Eds.). **Collaborative Software Engineering**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

MONÇORES, M. C., ALVIM, A. C. F., BARROS, M. O. "Large Neighborhood Search applied to the Software Module Clustering problem", **Computers & Operations Research**, v. 91, pp. 92-111, mar. 2018.

MORAES, A. *et al.* "Recommending Experts Using Communication History". In: **Proceedings of the 2Nd International Workshop on Recommendation Systems for Software Engineering**, pp. 41-45, New York, NY, USA: ACM, May 2010.

MPS.BR. **MPS.BR: Melhoria de Processo do Software Brasileiro, Guia Geral de Software**, 2012. Disponível em: <http://www.softex.br/>

MURRE, J. M. J., CHESSA, A. G. "Power laws from individual differences in learning and forgetting: mathematical analyses", **Psychonomic Bulletin & Review**, v. 18, n. 3, pp. 592–597, 2011.

MURRE, J. M. J., CHESSA, A. G., MEETER, M. "A mathematical model of forgetting and amnesia", **Frontiers in Psychology**, v. 4, p. 76, 2013.

MURRE, J. M. J., DROS, J. "Replication and Analysis of Ebbinghaus' Forgetting Curve", **PLOS ONE**, v. 10, n. 7, p. e0120644, Jul. 2015.

NAJAFABADI, M. K., MOHAMED, A. H., MAHRIN, M. N. "A survey on data mining techniques in recommender systems", **Soft Computing**, pp. 1-28, 2017.

NONAKA, I., TAKEUCHI, H. "The Theory of Organizational Knowledge Creation", **International Journal of Technology Management**, v. 11, n. 7/8, 1996.

OMG. **Object Management Group**. Disponível em: <http://www.omg.org/>.

OMORONYIA, I. 2008, **Enhancing awareness during distributed software development.** Ph.D. dissertation, Department of Computer and Information Sciences, University of Strathclyde, Glasgow, UK.

OMORONYIA, I. *et al.* "A Review of Awareness in Distributed Collaborative Software Engineering", **Softw. Pract. Exper.**, v. 40, n. 12, pp. 1107-1133, Nov. 2010.

O'SHEA, J. *et al.* "A Comparative Study of Two Short Text Semantic Similarity Measures". In: **Lecture Notes in Computer Science**, vol. 4953, **Agent and Multi-Agent Systems: Technologies and Applications**, pp. 172-181, Springer, Berlin, Heidelberg, 2008.

OVERBEEK, S. J. *et al.* "Characterizing Knowledge Intensive Tasks indicating Cognitive Requirements; Scenarios in Methods for Specific Tasks". In: **IFIP - The International Federation for Information Processing**, vol. 244, **Situational Method Engineering: Fundamentals and Experiences**, pp. 100-114, Springer, Boston, MA, 2007.

PEFFERS, K. *et al.* "A Design Science Research Methodology for Information Systems Research", **J. Manage. Inf. Syst.**, v. 24, n. 3, pp. 45-77, Dec. 2007.

PILLAT, R. M. *et al.* "BPMNt: A BPMN extension for specifying software process tailoring", **Information and Software Technology**, v. 57, pp. 95-115, Jan. 2015.

PROKSCH, S., BAUER, V., MURPHY, G. C. "How to Build a Recommendation System for Software Engineering". In: Meyer B., Nordio M. (eds) **Lecture Notes in**

Computer Science, vol. 8987, **Software Engineering: International Summer Schools, LASER 2013-2014, Elba, Italy, Revised Tutorial Lectures,** pp. 1-42, Springer, Cham, 2015.

QUMER, A., HENDERSON-SELLERS, B. "A framework to support the evaluation, adoption and improvement of agile methods in practice", **Journal of Systems and Software**, v. 81, n. 11, pp. 1899-1919, Nov. 2008.

ROBBES, R., LANZA, M. "SpyWare: A Change-aware Development Toolset". In: **Proceedings of the 30th International Conference on Software Engineering (ICSE '08)**, pp. 847-850, New York, NY, USA: ACM, May 2008.

ROBBES, R., RÖTHLISBERGER, D. "Using Developer Interaction Data to Compare Expertise Metrics". In: **Proceedings of the 10th Working Conference on Mining Software Repositories**, pp. 297-300, San Francisco, CA, Oct. 2013.

ROBILLARD, M. P. *et al.* (EDS.). **Recommendation Systems in Software Engineering**. Berlin Heidelberg: Springer-Verlag, 2014.

ROBILLARD, M. P., WALKER, R. J. "An Introduction to Recommendation Systems in Software Engineering". In: ROBILLARD, M. P. *et al.* (Eds.), **Recommendation Systems in Software Engineering**, chapter 1, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

ROCHA, A. R. C., MALDONADO, J. C., WEBER, K. C. **Qualidade de Software: Teoria e Prática**. São Paulo, Brasil: Prentice-Hall, 2001.

ROCHA, H. *et al.* "NextBug: a Bugzilla extension for recommending similar bugs", **Journal of Software Engineering Research and Development**, v. 3, n. 1, Dec. 2015.

ROEDIGER, I., "Henry L. Relativity of Remembering: Why the Laws of Memory Vanished", **Annual Review of Psychology**, v. 59, n. 1, pp. 225-254, Dec. 2007.

RUBIN, D. C., HINTON, S., WENZEL, A. "The precise time course of retention", **Journal of Experimental Psychology: Learning, Memory, and Cognition**, v. 25, n. 5, pp. 1161-1176, 1999.

RUBIN, D. C., WENZEL, A. E. "One hundred years of forgetting: A quantitative description of retention", **Psychological Review**, v. 103, n. 4, pp. 734-760, 1996.

RUNESON, P., HÖST, M. "Guidelines for conducting and reporting case study research in software engineering", **Empirical Software Engineering**, v. 14, n. 2, pp. 131, Dec. 2008.

RUS, I.; LINDVALL, M. "Knowledge management in software engineering", **IEEE Software**, v. 19, n. 3, pp. 26-38, May 2002.

SAKAI, T. "Metrics, statistics, tests". In: Ferro N. (eds), **Lecture Notes in Computer Science**, v. 8173, **Bridging Between Information Retrieval and Databases**, v. 8173, Springer, Berlin, Heidelberg, pp. 116–163, 2014.

SALOMON, D., MOTTA, G. **Handbook of Data Compression**. 5. ed. London: Springer-Verlag, 2010.

SANTOS, R. M. S., OLIVEIRA, T. C., E ABREU, F. B. "Mining Software Development Process Variations". In: **Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC '15)**, pp. 1657-1660, New York, NY, USA: ACM, April 2015.

SCHOBER, P., BOER, C., SCHWARTE, L. A. "Correlation Coefficients: Appropriate Use and Interpretation", **Anesthesia & Analgesia**, v. 126, n. 5, pp. 1763-1768, May 2018.

SHEETZ, S. D., HENDERSON, D., WALLACE, L. "Understanding developer and manager perceptions of function points and source lines of code", **Journal of Systems and Software**, v. 82, n. 9, pp. 1540-1549, Sep. 2009.

SOLIMAN, R., BRAUN, R., SIMOFF, S. "The essential ingredients of collaboration". In: **Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems**, pp. 366-373, St Louis, MO, May 2005.

SOUZA, C. R. B. DE, REDMILES, D. F. "The Awareness Network, To Whom Should I Display My Actions? And, Whose Actions Should I Monitor? ", **IEEE Transactions on Software Engineering**, v. 37, n. 3, pp. 325-340, May 2011.

SPEM. **Software & Systems Process Engineering Metamodel**. Disponível em: <https://www.omg.org/spec/SPEM>.

STOREY, M.-A. D., ČUBRANIĆ, D., GERMAN, D. M. "On the Use of Visualization to Support Awareness of Human Activities in Software Development: A Survey and a Framework". In: **Proceedings of the 2005 ACM Symposium on Software Visualization (SoftVis '05)**, pp. 193-202, New York, NY, USA: ACM, May 2005.

SUN, X. *et al.* "Exploring topic models in software engineering data analysis: A survey". **2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)**, pp. 357-362, Shanghai, July 2016.

SUN, X. *et al.* "Enhancing developer recommendation with supplementary information via mining historical commits", **Journal of Systems and Software**, v. 134, pp. 355-368, 2017.

THALHEIMER, W. **How Much Do People Forget?**, Abr. 2010. Disponível em: <http://www.work-learning.com/catalog.html>.

THOMPSON, C. A., MURPHY, G. C. "Recommending a Starting Point for a Programming Task: An Initial Investigation". In: **Proceedings of the 4th International Workshop on Recommendation Systems for Software Engineering**. **(**RSSE 2014), pp. 6-8, New York, NY, USA: ACM, June 2014.

TIBBLES, R., 2017, **Fractional Recall: Understanding Forgetting of Mathematics Learning from Computer Assisted Instruction Data and Implications for Personalized Learning in Low Resource Contexts**. Ph.D. dissertation, University of California, San Diego.

VAN DER AALST, W. M. P. "Process-Aware Information Systems: Lessons to Be Learned from Process Mining". In: **Lecture Notes in Computer Science**, vol. 5460, **Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on**

Concurrency in Process-Aware Information Systems, Springer, Berlin, Heidelberg, pp. 1-26, 2009.

WANG, Z. *et al.* "Recommending crowdsourced software developers in consideration of skill improvement". In: **2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)**, pp. 717-722, Urbana, IL, USA, Oct. 2017.

WHITEHEAD, J. *et al.* "Collaborative Software Engineering: Concepts and Techniques". In: MISTRÍK, I. *et al.* (Eds.), **Collaborative Software Engineering**, pp. 1-30, Springer, Berlin, Heidelberg, 2010.

WIXTED, J. T., EBBESEN, E. B. "On the Form of Forgetting", **Psychological Science**, v. 2, n. 6, pp. 409-415, Nov. 1991.

WOHLIN, C. *et al.* **Experimentation in Software Engineering: An Introduction**. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

WOHLIN, C. *et al.* **Experimentation in Software Engineering**. Berlin Heidelberg: Springer-Verlag, 2012.

XIA, X. *et al.* "Accurate developer recommendation for bug resolution". In**: 2013 20th Working Conference on Reverse Engineering (WCRE)**, pp. 72-81, Koblenz, Nov. 2013.

XIA, X. *et al.* "Dual analysis for recommending developers to resolve bugs", **Journal of Software: Evolution and Process**, v. 27, n. 3, pp. 195-220, 2015.

XIE, X. *et al.* "DRETOM: Developer Recommendation Based on Topic Models for Bug Resolution". In: **Proceedings of the 8th International Conference on Predictive Models in Software Engineering (PROMISE '12)**, pp. 19-28, New York, NY, USA: ACM, Sep. 2012.

YE, X., BUNESCU, R., LIU, C. "Learning to rank relevant files for bug reports using domain knowledge". In: **Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014)**, pp. 689-699, ACM, New York, NY, USA, Nov. 2014.

YE, Y., FISCHER, G. "Reuse-Conducive Development Environments", **Automated Software Engineering**, v. 12, n. 2, pp. 199-235, April 2005.

ZHANG, T. *et al.* "Towards more accurate severity prediction and fixer recommendation of software bugs", **Journal of Systems and Software**, v. 117, pp. 166-184, 2016.

ZHANG, Y., ZHOU, Y. "Knowledge forgetting: Properties and applications", **Artificial Intelligence**, v. 173, n. 16, pp. 1525-1537, Nov. 2009.

ZHOU, M., MOCKUS, A. "Developer Fluency: Achieving True Mastery in Software Projects". In: **Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE '10)**, pp. 137-146, New York, NY, USA: ACM, 2010.

ZHU, J., SHEN, B., HU, F. "A learning to rank framework for developer recommendation in software crowdsourcing". In: **2015 Asia-Pacific Software Engineering Conference (APSEC)**, pp. 285-292, New Delhi, 2015.

Apêndice A: Revisão (quasi-sistemática) da Literatura

Esta revisão da literatura foi realizada com o apoio da ferramenta Parsifal[36]. O Capítulo 6 apresenta uma análise dos artigos selecionados.

Recomendação de Contexto na Engenharia de Software

Edson Lucas

A presente revisão da literatura (quasi-sistemática) pretende identificar o estado da arte na recomendação de contexto para o desenvolvedor de software no início de uma nova tarefa em projetos de desenvolvimento software. Contexto é definido como o conjunto que contém os elementos desenvolvedores, artefatos e tarefas em um projeto de desenvolvimento de software.

Planning

Identificar trabalhos que têm como objetivo recomendar artefatos, tarefas e/ou desenvolvedores em projetos de desenvolvimento de software.

PICOC

- **Population:** software development, application development, application project, software engineering, software project, software project development, system development, system project
- **Intervention:** model, recommender system, recommendation system, recommender system, technique, method, tool
- **Comparison:** -
- **Outcome:** artifact, artefact, asset, context, developer, collaborator, expert, mentor, programmer, task
- **Context:** prediction, recommendation, recommending

Research Questions

1. Quais são as técnicas utilizadas para recomendar contexto em projetos de desenvolvimento de software?
2. O que recomendam? (pessoas; artefatos e/ou tarefas e/ou projeto)
3. Quais são os dados utilizados para recomendar contexto em projetos de desenvolvimento de software?
4. Quais as medidas utilizadas na avaliação da recomendação?

[36] https://parsif.al/ visitado em 25 de outubro de 2018.

Keywords and Synonyms

Keyword	Synonyms
artifact	artefact, asset
context	
developer	collaborator, expert, mentor, programmer
model	
prediction	
recommendation	recommending
recommender system	recommendation system, recommender system
software development	application development, application project, software engineering, software project, software project development, system development, system project
task	
technique	method
tool	

Search String

("software development" OR "application development" OR "application project" OR "software engineering" OR "software project" OR "software project development" OR "system development" OR "system project") AND ("model" OR "recommender system" OR "recommendation system" OR "recommender system" OR "technique" OR "method" OR "tool") AND ("artifact" OR "artefact" OR "asset" OR "context" OR "developer" OR "collaborator" OR "expert" OR "mentor" OR "programmer" OR "task") AND ("prediction" OR "recommendation" OR "recommending")

Sources

- Literature Review (http://10.1016/j.jss.2015.11.036)
- Science@Direct (http://www.sciencedirect.com)
- Scopus (http://www.scopus.com)

Selection Criteria

Inclusion Criteria:

- Apresentar uma ou mais técnicas para recomendar contexto em projetos de desenvolvimento de software

Exclusion Criteria:

- Não é estudo primário.

Quality Assessment Checklist

Questions:

- Qual é o tipo de estudo utilizado para avaliar a recomendação? (In Vivo=Muito Bom In Vitro=Bom; Simulação/Estudo de Caso=Regular; Descrição/Sem Avaliação=Ruim)
- Qual a linha de base da avaliação? (Dados Reais=Muito Bom; Opinião de Pessoas=Regular; Sem linha de base=Ruim)
- Qual o resultado da avaliação? (Avaliação Estatística=Muito Bom; Avaliação de Pessoas=Regular)

Answers:

- Muito Bom
- Bom
- Regular
- Ruim

Data Extraction Form

- Técnicas utilizadas para recomendar
- Escopo da recomendação
- Dados utilizados para recomendar
- O que recomendam
- Mede conhecimento?
- Considera o esquecimento humano?
- Considera a reaprendizagem?
- Aplica similaridade de tarefas por interação?
- Linha de base da avaliação da recomendação
- Medidas utilizadas na avaliação da recomendação
- O resultado da avaliação da recomendação

Conducting

Digital Libraries Search Strings

Literature Review:

- What recommendation systems for software engineering recommend: A systematic literature review - Quarenta e seis (46) artigos selecionados de 2003 a 2013 (10.1016/j.jss.2015.11.036) (ARTIGOS IMPORTADOS)

- User and context information in context-aware recommender systems: A systematic literature review - 2012 to 2015 - (https://link.springer.com/chapter/10.1007/978-3-319-

31232-3_61) (SEM ACESSO AO ARTIGO) (IMPORTADAS AS REFS DO ARTIGO)

- Recommender system application developments: A survey () (REJEITADO)

- Exploring topic models in software engineering data analysis: A survey (ARTIGOS IMPORTADOS)

Science@Direct:

(title-abstr-key("software development" OR "application development" OR "application project" OR "software engineering" OR "software project" OR "software project development" OR "system development" OR "system project") AND ("model" OR "recommender system" OR "recommendation system" OR "recommender system" OR "technique" OR "method" OR "tool") AND ("artifact" OR "artefact" OR "asset" OR "context" OR "developer" OR "collaborator" OR "expert" OR "mentor" OR "programmer" OR "task") AND ("prediction" OR "recommendation" OR "recommending")) AND (pub-date AFT 20121231) [All Sources(Computer Science)]

Scopus:

(TITLE-ABS-KEY(("software development" OR "application development" OR "application project" OR "software engineering" OR "software project" OR "software project development" OR "system development" OR "system project") AND ("model" OR "recommender system" OR "recommendation system" OR "recommender system" OR "technique" OR "method" OR "tool") AND ("artifact" OR "artefact" OR "asset" OR "context" OR "developer" OR "collaborator" OR "expert" OR "mentor" OR "programmer" OR "task") AND ("prediction" OR "recommendation" OR "recommending"))) AND (PUBYEAR > 2012)

Imported Studies - July 4, 2018

- **Literature Review:** 97
- **Science@Direct:** 1288
- **Scopus:** 1129

Data Extraction

Tabela 25 e Tabela 26.

Tabela 25 – Extração de dados dos artigos selecionados.

Article	Técnicas utilizadas para recomendar	Escopo da recomendação	Dados utilizados para recomendar	O que recomendam	Mede conhecimento	Considera o esquecimento humano
(ANDRIC *et al.*, 2004)	zzstructure, TF-IDF	Projeto	700 documents in 1000 files and 2000 issues. 50 users who generated around 5000 attributes.	Artefatos		False
(ČUBRANIĆ *et al.*, 2005b)	Latent Semantic Indexing (LSI) for determining text similarity. Project artifacts themselves and also of links between those artifacts indicating relationships.	Projeto	CVS source repository. Issue-tracking system (Bugzilla). Communication channels ((email messages or discussion forum). Online documentation (reference manuals, web site).	Artefatos		False
(ASHOK *et al.*, 2009)	Typed Documents and Similarity. Relationship graph.	Projeto	Version control, bug database, logs of debugger sessions,	Artefatos, Especialistas em tarefas		False
(MORAES *et al.*, 2010)	Fuzzy similarity.	Projeto	Conscius uses the source code, its history, the project documentation (javadoc) and the developer's mailing list archives to recommend source code experts.	Especialistas em artefatos	Artefato	False
(ANTUNES *et al.*, 2012)	knowledge base that uses an ontology to represent of the source code structure that is stored in the workspace of a developer.	Projeto	The source code elements that are more relevant for developers during their work, providing an insight to what is their focus of attention at each moment.	Artefatos		False

(CANFORA *et al.*, 2012)	Mineração de repositório de código e lista de e-mails.	Projeto	submissão de código e lista de e-mails.	Especialistas no projeto		False
(MALHEIROS *et al.*, 2012)	Prediction by Partial Matching (PPM)	Projeto	change requests (reported bugs)	Tarefas Similares		False
(XIE *et al.*, 2012)	LDA (Latent Dirichlet Allocation.	Projeto	bug report (title and the full description contents of bug reports to build topic models.)	Especialistas em tarefas		False
(XIA *et al.*, 2013)	Euclidean Distance Metric, Multi-Label Classification, Latent Dirichlet Allocation (LDA).	Projeto	Bug reports: summary; description; product, platform component, developers.	Especialistas em tarefas		False
(FRITZ *et al.*, 2014)	Mathematical model.	Projeto	Authorship data from the source revision system and interaction data from monitoring the developer's activity in the development environment (degree-of-interest- DOI).	Especialistas em artefatos	Artefato	False
(YE *et al.*, 2014)	Collaborative filtering. Learning-to-rank technique. Vector Space Model (VSM). Cosine similarity.	Projeto	Bug report (Summary, Description)	Artefatos		False
(CAICEDO e DUARTE, 2015)	Similar to Collaborative Searching	Projeto	simulation	Artefatos		False
(ALMHANA *et al.*, 2016)	Cosine similarity [24] between the description of a bug report and the source code (LS). Similaridade baseada no histórico de correções de erro (HS): number of times that a class was fixed to eliminate bugs;	Projeto	Histórico de bugs (texto); Código fonte (parte do texto).	Artefatos		False

	recommended class was recently changed or fixed; classes that are recommended together for similar previous bug reports.					
(GUO *et al.*, 2016)	Genetic Algorithm (GA)	Projeto	Source artifacts (such as requirements), target artifacts (such as source code). Find the Expert was constructed from five sets of feature requests (containing requirements and users) and fif- teen sets of bug assignments (consisting of bug reports with associated fixes as well as the names of developers assigned to fix the bugs).	Especialistas em tarefas		False
(ZHANG *et al.*, 2016)	Latent Dirichlet Allocation (LDA); BM25ext ; K-Nearest Neighbours.	Projeto	Histórico de bugs reportados: texto do sumário e da descrição.	Especialistas em tarefas		False
(ZHU *et al.*, 2016)	Latent Dirichlet Allocation (LDA).	Projeto	Software crowdsourcing platform.	Especialistas em tarefas		False
(SUN *et al.*, 2017)	Collaborative topic modeling (CTM) technique to analyze the historical commit repositories. Natural language processing (NLP) techniques	Projeto	Source code files, commits.	Artefatos, Especialistas em tarefas	Artefato	False
(WANG *et al.*, 2017)	Learning curve models. K-Means to cluster the tasks in the historical data.	Projeto	We crawled a dataset from Topcoder involving 32,565 challenges, 7,620 developers who made 59,230 submissions from 2006 to 2016.	Especialistas em tarefas		False

Tabela 26 - Extração de dados dos artigos selecionados, continuação da Tabela 25.

Article	Mede conhecimento	Aplica similaridade de tarefas por interação	Linha de base da avaliação da recomendação	Medidas utilizadas na avaliação da recomendação	O resultado da avaliação da recomendação
(ANDRIC *et al.*, 2004)		False	the full text search (FTS)	Precision	The results indicate better accuracy for the “A LA” system.
(ČUBRANIĆ *et al.*, 2005b)		False	Source files were relevant to each of the bugs in the sample, based on the fix that was eventually implemented.	Precision e Recall	Hipikat was able to provide a useful pointer to the files involved in the solution of the task.
(ASHOK *et al.*, 2009)		False	User feedback logged in the tool, and anecdotal feedback through emails from users, and interviews with users. Full-text search.	Precision, Recall, Percentage	User feedback (useful results): 78% in study one; 75% in study two. DebugAdvisor’s first phase and full-text search: DebugAdvisor’s improves recall by 33%.
(MORAES *et al.*, 2010)	Artefato	False	source code history.	Precision, Recall e Accuracy.	Our experiment showed evidence of the potential value of communication content as a source to expert location. The results even overcome results relying only on the source code history.
(ANTUNES *et al.*, 2012)		False	The source code elements opened for the first time were already being recommended by the system.	Percentage, average.	In average, considering all query sizes, 42.7% of the source code elements opened for the first time were already being recommended by the system.
(CANFORA *et al.*, 2012)		False	Número de *commits* enviados pelos desenvolvedores.	Precisão	O autor: Yoda has been evaluated on data from five

					open source projects, Apache httpd, the FreeBSD kernel, PostgreSQL, Python, and Samba. Results of the study indicate that, except for FreeBSD, Yoda is able to identify candidate pairs of mentor-newcomer with a precision in most cases higher than 80%, and recommend them with a precision greater than 70%.
(MALHEIROS *et al.*, 2012)		False	LSI	Precision, recall and recall rate	All results using PPM to find similar change requests were better than the results using LSI.
(XIE *et al.*, 2012)		False	DREX. Developers participate in each bug resolution .	Precision, recall and F1-measure.	In our experiment for Mozilla Firefox, how- ever, the counterpart threshold is 46. Besides, the size of data set they used is 5195 (from 2002/02 to 2009/08). Considering these adjustable differences, we think our approach is competitive with DREX. DRETOM can achieve high recall up to 82% and 50% with top 5 and top 7 recommendations for Eclipse JDT and Mozilla Firefox respectively.
(XIA *et al.*, 2013)		False	Bugzie and DREX	Recall@K	DevRec improves the average recall@5 and

					recall@10 scores of Bugzie by 57.55% and 39.39%, respectively. DevRec also outperforms DREX by improving the average recall@5 and recall@10 scores by 165.38% and 89.36%, respectively.
(FRITZ *et al.*, 2014)	Artefato	False	We asked each developer.	Percentage	The case studies we conducted supply initial evidence that our model can provide value in scenarios, such as expert finding and identifying changes of interest.The case studies also show that our model does not adequately reflect a developer's knowledge for API elements.
(YE *et al.*, 2014)		False	The standard VSM method that ranks source files based on their textual similarity with the bug report. The Usual Suspects method that recommends only the top k most frequently fixed files. BugLocator. BugScout.	Accuracy@k Mean Average Precision (MAP) Precision@k Mean Reciprocal Rank (MRR)	Experimental evaluations on six Java projects show that our approach can locate the relevant files within the top 10 recommendations for over 70% of the bug reports in Eclipse Platform and Tomcat. Our ranking model outperforms BugLocator [45] and BugScout [34], two recent state-of-the-art approaches.
(CAICEDO e		False	simulation	Precision and Recall	CodeRAnts was not evaluated with the same experimental method and

DUARTE, 2015)					setting carried out in the other systems by other researchers.
(ALMHANA *et al.*, 2016)		False	Lexical measure (LS); History measure (HS).	Precision@K, Recall@K, Accuracy@K.	Autores: The results on the before fix versions show that our system outperforms, on average, three state-of-the-art approaches not based on search techniques [16, 26, 28]. In particular, our search-based approach is able to successfully locate the true buggy methods within the top 10 recommendations for over 87% of the bug reports.
(GUO *et al.*, 2016)		False	best-of-breed	PercentOfAchieved-Maximum (PAM)	The cold-start solution proposed in this paper is relatively effective for Find the Expert.
(ZHANG *et al.*, 2016)		False	DRETOM, DREX e DevRec.	Precision, Recall, F-measure e MRR.	Autores: Our approach for semi-automatic fixer recommendation outperforms the cutting-edge approaches such as DRETOM, DREX, and DevRec when we consider the appropriate number of the nearest neighbours of the new bug report.
(ZHU *et al.*, 2016)		False	Pointwise, pairwise and listwise algorithms.	Precision, Recall e F1-Measure.	The experiments on real-world dataset show the feasibility and effectiveness of our approach.

(SUN *et al.*, 2017)	Artefato	False	The iMacPro, Location and ABA-Time-tf-idf. BR-Tracer.	Recall@K ATNF (accuracy of top N files) STNF (skillful for top N files) ratio.	we can conclude that the accuracy of EDR_SI recommending developers is improved over the state-of-art ap- proaches, i.e., iMacPro, Location and ABA-Time-tf-idf. In conclusion, EDR_SI can recommend more useful and personalized files over the traditional bug location technique, i.e., BR- Tracer.
(WANG *et al.*, 2017)		False	Future work	Future work	Future work

Apêndice B: Implementação de Referência dos Modelos Tacin

A construção dos modelos para medir conhecimento utilizaram os dados do projeto Mylyn Dcos coletados através da consulta do Bugzilla do Eclipse[37], com os parâmetros: Resolution=Fixed; Status=Resolved; Product=Mylyn; Attachment description=mylyn/context/zip. A consulta foi realizada em 15 de janeiro de 2016 retornando os Ids das tarefas desde o início do Projeto. Em seguida, *scripts* foram criados para baixar os dados de cada tarefa. A coleta resultou em: 1804 tarefas contendo a *descrição curta* textual e as suas datas de início e término; 5515 artefatos que tiveram interação de edição capturados pelo modelo DOI; e 197649 interações de edição realizadas pelos desenvolvedores enquanto corrigiram erros no Projeto Mylyn.

Inicialmente, a implementação de referência para a construção dos modelos do Tacin foi realizada com a linguagem SQL, utilizando a ferramenta pgAdmin para administrar banco de dados Postgres. Depois da construção e calibração dos modelos, a implementação foi migrada para a linguagem Java com o objetivo de facilitar a construção dos experimentos descritos no Capítulo 5. A classe *TacinMeasureTest* foi desenvolvida para garantir que os modelos expressos em SQL e Java produziam os mesmos resultados. Além disso, um revisor verificou a correção da implementação dos modelos nas duas linguagens.

A Figura 27 mostra o modelo de classes da implementação de referência dos modelos Tacin. A classe *TacinMeasure* possui os métodos *computeKa*, *computeKs*, *computeKc* e *computeKp* para indicar a implementação dos modelos em Java. Esses métodos têm acesso aos dados através da classe *InteractionDataManager*. O pacote *PPGI.UNIRIO* representa a implementação de referência do algoritmo de clusterização LNS, utilizado para organizar o histórico de tarefas do projeto em tarefas similares por interação em artefatos. A clusterização de tarefas é necessária para a implementação dos métodos *computeKc* e *computeKp*.

Especificamente para atender às necessidades do experimento, a classe *TacinMeasureManager* gera listas de classificação dos desenvolvedores do projeto considerando as informações de *número de tarefas realizadas*, *número de interações realizadas* e os resultados dos modelos K_c e K_p. Além disso, o método *computeSpearman*

[37] https://bugs.eclipse.org/bugs/query.cgi

foi implementado com a ferramenta RStudio Desktop para medir a similaridade entre as listas de classificação dos desenvolvedores por *número de tarefas realizadas* com as outras listas já citadas acima. Finalmente, a classe *TacinExperiment* representa os passos necessários para realizar o planejamento do estudo conforme apresentado no Capítulo 5.

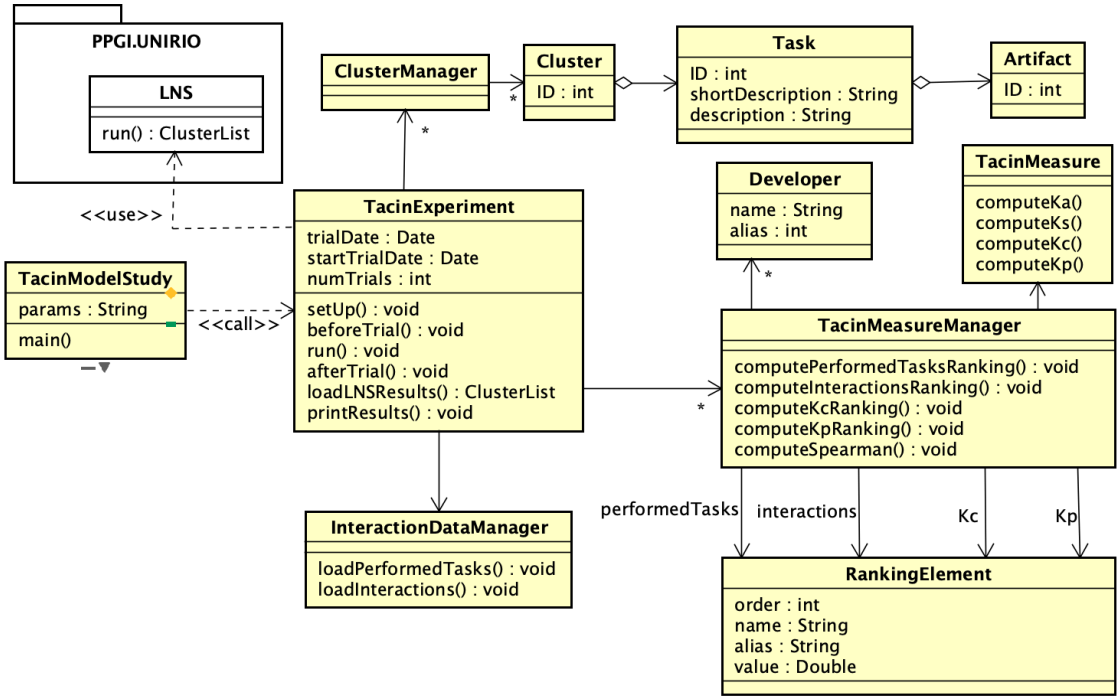

Figura 27 – Modelo de classes da implementação de referência dos modelos Tacin expresso em UML[38] (Unified Modeling Language).

[38] https://www.uml.org/

Apêndice C: Implementação de Referência do Método Tacin

A implementação de referência do método Tacin foi utilizada para realizar a avaliação da recomendação de artefatos e desenvolvedores, descrita no Capítulo 5. Essa implementação possibilita a comparação do Tacin com o algoritmo que recomenda a tarefa vizinha mais próxima da nova tarefa ($KNN\ com\ k = 1$). A seguir serão apresentados o modelo de classes expresso em UML e o processo para calcular a similaridade textual entre uma nova tarefa com grupos de tarefas similares por interação em artefatos.

A **Error! Reference source not found.** mostra o modelo de classes dessa i mplementação. A classe *TacinNewTaskContextStudy* representa os parâmetros iniciais do estudo. A classe *TacinNewTaskContextStudyExperiment* herda a estrutura da classe *TacinExperiment* e acrescenta o método *computeTextualSimilarityNewTaskToCluster* para calcular a similaridade textual entre uma nova tarefa e os clusters gerados pelo algoritmo de clusterização LNS. A classe *TacinNewTaskContextManager* gerencia a avaliação para cada rodada de recomendação de contexto de nova tarefa realizada pelo Tacin. A classe *ContextEvaluate* avalia se os artefatos recomendados realmente foram alterados pelos desenvolvedores na realização da nova tarefa. A avaliação é realizada pelos métodos *computeRecallArtifacts*, *computeReductionArtifacts*, *computeRc-MeasureArtifacts*, *computeRecallDevelopersTop3* e *computeRecallDevelopersTop5* para cada recomendação de contexto de nova tarefa.

A classe *NewTaskContext* representa o contexto recomendado pelo Tacin para uma nova tarefa. A classe *Cluster* representa os grupos de tarefas similares por interação em artefatos, por isso representado como a composição de várias tarefas, classe *Task*, que

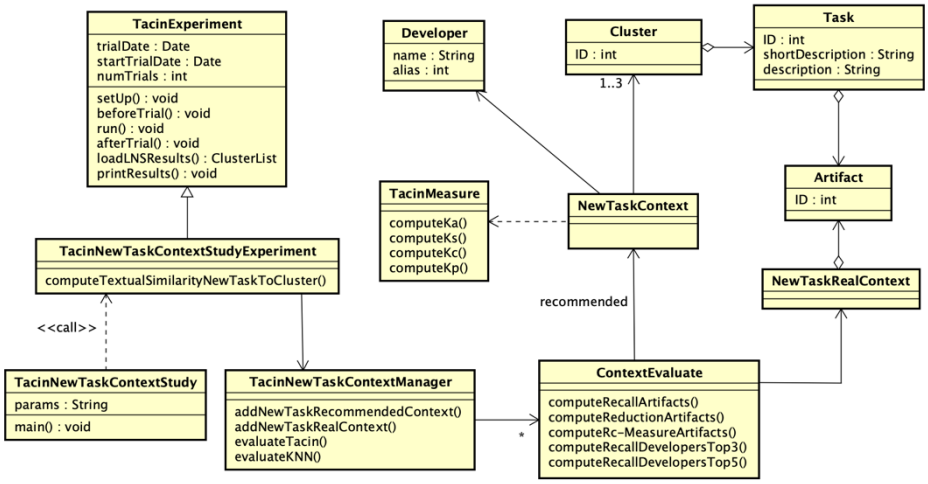

Figura 28 - Modelo de classes da implementação de referência do método Tacin expresso em UML.

por sua vez pode estar relacionada a vários artefatos. O contexto recomendado ainda indica os desenvolvedores com mais conhecimento nos grupos recomendados com as suas tarefas e artefatos. Especificamente para o experimento, o contexto real da nova tarefa está representado pela classe *NewTaskRealContext* que representa os artefatos alterados pelos desenvolvedores na realização da nova tarefa.

O método *computeTextualSimilarityNewTaskToCluster* da classe *TacinNewTaskContextStudyExperiment* foi implementado no RapidMiner Studio. Um processo é implementado no RapidMiner através da ligação de operadores. Um operador encapsula funcionalidades e pode apresentar várias entradas e saídas. A Figura 29 mostra o processo para calcular a similaridade textual entre uma nova tarefa e grupos de tarefas similares por interação em artefatos. O processo Calcular Similaridade Textual recebe como entrada a nova tarefa e a organização do histórico de tarefas organizados em clusters de tarefas similares por interação em artefatos. A primeira linha do arquivo de entrada contém as informações da nova tarefa e as demais linhas contêm as informações dos clusters. O operador *Process Documents* representa o Subprocesso responsável por realizar o pré-processamento do texto, em seguida os operadores *Data to Similarity*, *Similarity to Data* e *Filter Examples* produzem a similaridade textual entre a nova tarefa e os clusters utilizando a medida de cosseno por similaridade entre os vetores que representam os textos. E, finalmente, o operador *Write CSV* grava os resultados em um arquivo.

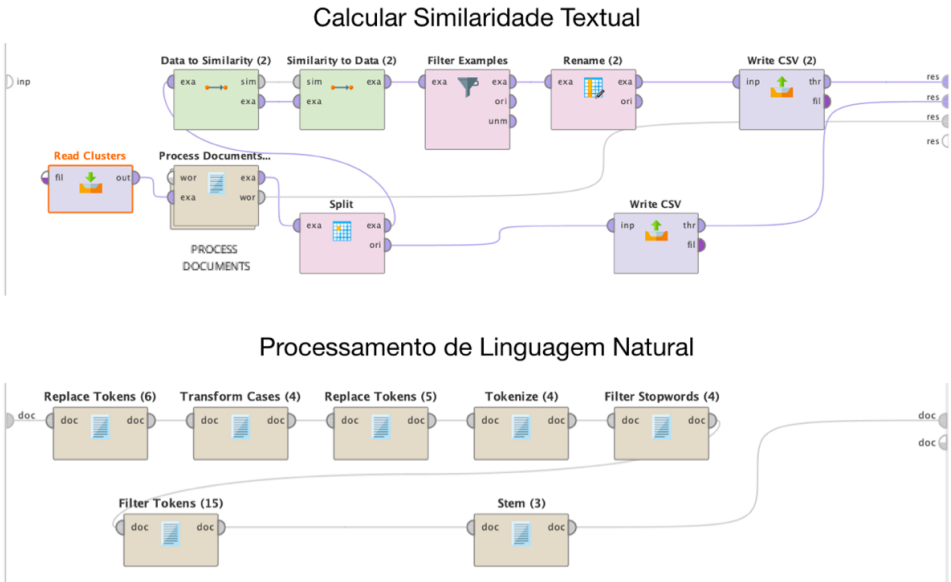

Figura 29 -Processo para calcular a similaridade textual entre uma nova tarefa e grupos de tarefas similares por interação em artefatos.