Abigail Sayury Nakashima

Avaliação de Técnicas de Aprendizado de Máquina na Classificação de Taxonomia em Dívida Técnica Auto-Admitida

Anteprojeto de pesquisa ser desenvolvido junto ao Curso de Bacharelado em Ciência da Computação do Departamento de Matemática e Computação da Faculdade de Ciências e Tecnologia - UNESP

Prof. Dr. Rogério Eduardo Garcia
(Orientador)

Presidente Prudente - SP
Agosto - 2025

Resumo

Na Engenharia de Software, a Dívida Técnica (DT) é uma metafóra que destaca a escolha por velocidade de entrega em detrimento da execução ampla das atividades técnicas de desenvolvimento de software. A metafóra representa uma troca entre a liberação rápida de um software para clientes e as implicações e custos futuros que o adiamento dessas atividades técnicas podem resultar. A falta de gestão implica no acúmulo de DT compromete a qualidade do software, tornando sua manutenção mais complexa e cara. A DT pode ser cometida intencionamente resultando na *Dívida Técnica Auto-Admitida* (DTAA), a sua gestão é satisfeita quanto a identificação de instâncias problematicas e investigada quando a sua categoriazação. A literatura apresenta uma Taxonomia para DTAA, contudo a automação desse processo ainda é pouco explorada. Neste contexto, este projeto tem como objetivo avaliar a eficácia de técnicas de Aprendizado de Máquina na Classificação de Taxonomia em DTAA.

Palavras-Chave: Dívida Técnica Auto-Admitida, Taxonomia de Dívida Técnica, Manutenção de Software, Aprendizado de Máquina.

1 Apresentação

O acúmulo de *Dívida Técnica* (DT) é um fator que contribui significativamente para a degradação da qualidade do software. Especificamente no contexto da *Dívida Técnica Auto-Admitida* (DTAA), sua presença torna a manutenção do código-fonte um processo mais caro e complexo [15]. Na era dos dados, o uso de Inteligência Artificial para automatizar tarefas na Engenharia de Software torna-se cada vez mais essencial. Entretanto, no contexto da classificação de DTAA segundo sua taxonomia, esse processo ainda é incipiente [7].

Neste âmbito, este projeto de pesquisa tem como foco avaliar técnicas clássicas de Aprendizado de Máquina para a classificação de DTAA com base na taxonomia apresentada por Fucci [7]. De modo complementar, será realizada uma ampliação do conjunto de dados disponível na literatura, por meio de novas rotulagens, com o intuito de subsidiar a comunidade científica no desenvolvimento de futuras pesquisas [8].

Este projeto de pesquisa está organizado da seguinte forma: a Seção 2 apresenta uma breve contextualização sobre DT e DTAA; a Seção 3 descreve a proposta de trabalho, contemplando a problemática, motivação, justificativa, objetivo e metodologia de trabalho; o cronograma proposto para o desenvolvimento do projeto e as perspectivas de contribuição.

2 Contextualização

DT é uma metáfora que descreve a relação custo-benefício entre retornos a curto prazo adiando atividades técnicas de desenvolvimento de software e consequências deste adiamento a longo prazo [2]. A metáfora de DT foi apresentada em um relato de experiência, descrevendo uma situação em que qualidade de código era trocada por ganhos gerais em curto prazo [5].

O termo foi originalmente usado com o foco em práticas de codificação, e depois a metáfora teve seu conceito ampliado, para incluir a arquitetura, teste, ou mesmo a documentação do software. Desse modo, uma DT pode estar relacionada a um tipo específico de artefato, por exemplo: um documento de requisitos incompleto, desatualizado ou inexistente caracteriza uma Dívida de Documentação; uma Dívida de Teste ocorre quando um caso de teste deixa de ser executado, ou mesmo a falta de um plano de teste para o software; Dívida de Projeto/Código é formada quando são encontradas deficiências no código-fonte causadas por violações das boas práticas de programação [9].

Considerando o ponto de vista econômico, DT é definida como o custo para reparar problemas de qualidade de software, com o objetivo de alcançar o nível de qualidade desejado pela organização, explica [13]. Em analogia com dívidas financeiras, DT também resulta em juros que precisam ser pagos futuramente no processo de desenvolvimento. Tais juros consistem em esforços extras que devem ser dedicados para sanar a DT acumulada [10].

2.1 Dívida Técnica Auto-Admitida

A DT pode ser introduzida consciente ou inconscientemente. Gerentes de projetos podem optar por não realizar uma atividade técnica causando a introdução consciente da dívida, ou mesmo, programadores inexperientes podem produzir código de baixa qualidade e acabar inconscientemente introduzindo DT. Dívida Técnica (DT) é caracterizada pela sua visibilidade, que pode ser introduzida de forma consciente ou inconsciente [2]. Quando um programador inexperiente, por exemplo, produz um código de baixa qualidade sem intenção, essa DT se torna invisível para os gerentes do projeto. Nesses casos, a equipe necessita de mecanismos eficientes para identificar e priorizar sua resolução, um processo conhecido como pagamento da DT [2].

A situação em que atividades técnicas de desenvolvimento não são realizadas, ou mesmo correções rápidas ou temporárias são inseridas de modo intencional em um projeto de software (devido a corte de custos ou pressão do mercado), mas documentadas em comentários ao longo do código-fonte, caracteriza a formação da chamada *Dívida Técnica Auto-Admitida* (DTAA) [15]. Na Tabela 1 são exemplificadas alguns exemplos de instâncias de DTAA documentadas por meio de comentários no código-fonte.

Os comentários presentes ao longo do código-fonte são a principal forma dos gestores e desenvolvedores documentar e indicar que cometeram uma DT intencionalmente [15, 6]. Assim, nos últimos anos alguns estudos foram apresentados na literatura investigando e descrevendo estratégias para identificar, entender, priorizar e gerenciar DTAA [14, 1, 15, 6, 8].

3 Proposta de Trabalho

3.1 Formulação do Problema

Idealizado por Potdar [14], o conceito de *Dívida Técnica Auto-Admitida* (DTAA) consolidou-se como uma área de pesquisa promissora [15]. A frequência da DTAA em projetos de software tem aumentado, e sua presença torna o código mais difícil

Tabela 1: Exemplificação de comentários que indicam DTAA – Adaptado de [14, 11, 12, 8]

Exemplos de comentários DTAA
//todo should check that error has been logged
//Unsafe; should error
//FIXME: This is such a gross hack...
//Ugly, but what else?
//TODO: - This method is too complex, lets break it up
//TODO - need a lot more tests
//Bug in above method
//Subquery ! yuck !
//Fixme this correct
//Strictly speaking, this is a design error
//TODO: this isn't quite right but it's ok for now

de manter e evoluir [14, 16]. Dessa forma, a identificação e a classificação consistentes das instâncias de DTAA, visando à sua compreensão e melhor gestão futura, são tarefas fundamentais [6]. Nesse contexto, compreender a DTAA em um projeto configura-se como uma estratégia promissora para apoiar decisões e atividades de gestão. Entretanto, sua classificação e organização automática, conforme uma taxonomia estruturada, ainda demandam investigações mais aprofundadas [7, 4, 3].

3.2 Motivação e Justificativa

A comunidade ciêntifica ciente dos problemas e impactos que a presença e acúmulo de DTAA resultam na capacidade de manutenção do software, investigou e apresentou diferentes técnicas de identificação de DTAA, além de destacar a importância de sua priorização [15, 7]. Entretanto, o processo de classificação de DTAA segundo taxonomias existentes ainda é majoritariamente manual, demandando tempo e esforço consideráveis por parte de desenvolvedores e pesquisadores. Nesse cenário, a aplicação de técnicas de Aprendizado de Máquina surge como uma oportunidade para automatizar e padronizar essa classificação, reduzindo custos de análise, acelerando a tomada de decisão e contribuindo para a evolução do conhecimento científico na área.

Uma vez que, a analise das técnicas de classificação DTAA considerando sua Taxonomia esteja concluída, isso representará um avanço na automação ainda pouco explorada. Ao final, a pesquisa fornecerá dados sobre a eficácia dessas técnicas, o

que pode ajudar a indústria de software a identificar, de forma precisa, categorias da Taxonomia de Fucci em instâncias de DTAA. Com isso, será possível apoiar a gestão de DTAA e contribuir para a melhoria da capacidade de manutenção do código-fonte, isso justifica esse projeto de pesquisa.

3.3 Objetivo do Projeto

O objetivo geral deste projeto de pesquisa é avaliar e comparar o desempenho de técnicas de Aprendizado de Máquina na classificação de Dívida Técnica Auto-Admitida evidenciando técnicas apropriadas para a classificação seguindo a Taxonomia de Fucci [7].

Para isso, tem como objetivos específicos: A ampliação do do conjunto de dados de instâncias de DTAA manualmente classificadas, agora seguindo a categorias da Taxonomia de Fucci [7]; Implementação e análise do uso das técnicas de Aprendizado de Máquina para classificação; Indicar a perspectiva de estado da prática, a potencial vantagem de utilização na industria de desenvolvimento de software, no contexto de DTAA.

3.4 Metodologia

Neste trabalho incialmente será realizado uma Revisão Bibliográfica com a finalidade de obter uma maior fundamentação conceitual inicialmente sobre DT buscando uma maior compreensão e exploração dos detalhes que estão presentes nesta metáfora, como por exemplo: Gestão de DT e suas atividades.

Entretanto, o foco dessa Revisão da Bibliográfia é direcionando em estudos para ampliar conhecimentos em DTAA, técnicas de Identificação e sua gestão. Entendimento dos conjuntos de dados utilizados em pesquisas no ambito de DTAA e dominio da Taxonomia apresentada por Fucci [7] – Figura 1. Adicionalmente, ocorrerá um aprofundamento nos estudos de técnicas de Aprendizado de Máquina para o problema de classificação multiclasse e das Métricas de verificação de eficácia para analisar as técnicas de Aprendizado empregadas.

A etapa seguinte consiste na ampliação do conjunto de dados de instâncias de DTAA manualmente classificadas. Para isso, será utilizado o conjunto de dados [1] disponibilizado na literatura [8], que realiza o agrupamento dos conjuntos de dados existentes em DTAA, mas não faz a classificação manual por categorias seguindo a Taxonomia de Fucci [7]. Essa etapa será conduzida com apoio do grupo de pesquisa em Computação Aplicada, composto por membros do Laboratório de Pesquisa em

[1] `https://github.com/zscszndxdxs/2023-MT-BERT-SATD`

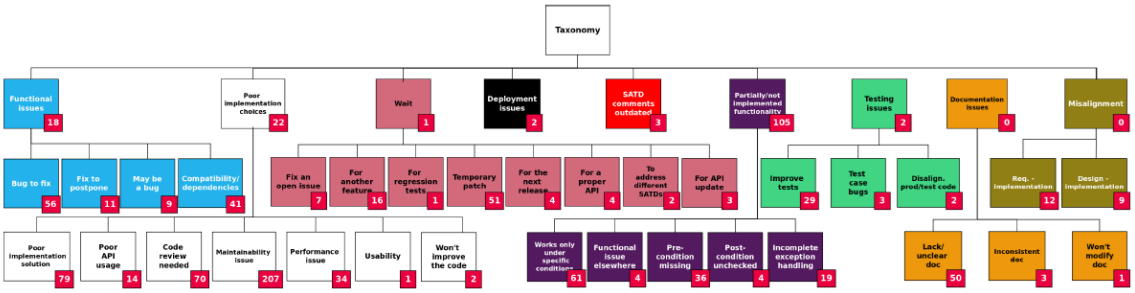

Figura 1: Taxonomia proposta por Fucci et al. [7].

Engenharia de Software Aplicada, incluindo alunos de Iniciação Científica, Mestrado e Doutorado da Faculdade de Ciências e Tecnologia da Universidade Estadual Paulista. Essa etapa será conduzida em conjunto com um projeto de pesquisa em nível de Doutorado.

A classificação de categorias realizada manualmente seguirá o processo definido por Fucci et al. [7]. Na Tabela 2 são resumidos os projetos de softwares [2] que fazem parte do conjunto de dados [8]. As características dos repositórios de Softwares são: possuem versionamento de código-fonte, taxa de comentários de código alta ou média e implementados em Linguagem JAVA, Python, Ruby, entre outras.

Em paralelo a ampliação do conjunto de dados, será realizado um Estudo de Tecnologias visando subsidiar a implementação de classificadores para identificação de DTAA. Tais técnologias constitem em: Estrutura de Arquivos, Ferramentas de Controle de Versão, Linguagem Python e seu ambiente de desenvolvimento, Bibliotecas direcionadas para Aprendizado de Máquina, tais como: Keras, TensorFlow, entre outras.

Após a ampliação do conjunto de dados e a conclusão do estudo de tecnologias, a próxima etapa será a implementação das técnicas de Aprendizado de Máquina para a classificação. Para isso, serão implementados classificadores: *Naive Bayes*, *Support Vector Machine*, *Random Forest* e *Gradient Boosting*.

O *F1-Score* será a medida de verificação da eficácia dos classificadores construídos. O *F1-Score* permite comparar a classificação fornecida por diversos classificadores que aplicam diferentes técnicas de Aprendizado. Deste modo, a análise da eficácia dos classificadores será dada pela *F1-Score*, que considera as medidas de Precisão e Revocação. Tais medidas são obtidas comparando a classificação automática e manual.

[2] `https://openhub.net/`

Tabela 2: Parte dos repositórios de software pertencentes aos conjuntos de dados a serem utilizados - Adaptado de [7, 6, 3, 8].

Repositório	**Desenvolvedores**	**LOC**	**Taxa de comentários**
Apache Ant	149	243K	Alta
Apache JMeter	121	303K	Média
Apache POI	53	511K	Média
Apache ZoooKeeper	283	177K	Média
ArgoUML	113	1.67M	Alta
Columba	10	155K	Alta
Dubbo	507	200K	Média
jEdit	56	316K	Média
Hibernate ORM	456	745K	Baixa
JFreeChart	20	313K	Alta
JRuby	528	916K	Baixa
SpringFramework	1112	813K	Média
SQuirrel	52	502K	Média

3.5 Plano de trabalho e cronograma de sua execução

Esse projeto de pesquisa, conforme a metodologia expressa na seção 3.4, foi dividido nas atividades descritas a seguir:

1. Revisão Bibliográfica: condução de uma Revisão da literatura no contexto de DTAA.

2. Ampliação do Conjunto de Dados: Ampliação do Conjunto de dados disponibilizado na literatura [8] por meio da efetuação de rotulagem manual considerando a Taxonomia de Fucci, conduzida em conjunto com o grupo de pesquisa em Computação Aplicada da FCT-UNESP.

3. Estudo de Tecnologias: Estudo de técnologias para subsidiar a implementação de classificadores utilizando técnicas de Aprendizado de Máquina.

4. Implementação dos Classificadores Multiclasse DTAA: construção de classificadores para categorias da Taxonomia de Fucci para DTAA.

5. Análise dos Classificação Multiclasses: os classificadores vão ser analisados e avaliados por meio de experimentos controlados e comparação com uso de medidas estatísticas, em principal *F1-Score*.

6. Escrita de relatórios e artigos: escrita de relatórios científicos acerca do andamento do trabalho e artigos com resultados parciais e finais.

Tabela 3: Cronograma de atividades

Atividades	**Bimestres**					
	1	**2**	**3**	**4**	**5**	**6**
1. Revisão Bibliográfica	x	x	x			
2. Ampliação do Conjunto de Dados		x	x	x		
3. Estudo de Técnologias			x	x		
4. Implementação dos Classificadores Multiclasse			x	x	x	
5. Análise dos Classificadores Multiclasse				x	x	x
6. Escrita de relatórios e artigos			x	x	x	x

As atividades estão organizadas em bimestres de acordo com o cronograma apresentado na Tabela 3, nas quais as atividades futuras são representadas por: 'x'.

3.6 Perspectivas de Contribuição do Trabalho

O projeto proposto está no contexto de outros trabalhos já desenvolvidos e em desenvolvimento pelo grupo de pesquisa em Computação Aplicada da FCT-UNESP. Essa análise irá contribuir para fomentar o uso de técnicas para classificação automatizada de categorias de Taxonomias em DTAA.

Adicionalmente, a ampliação do conjunto de dados de instâncias de DTAA classificadas manualmente quanto a sua categoria pertencente a Taxonomia de Fucci [7] deve subsidiar pesquisas realizadas pelo grupo de Computação Aplicada e comunidade ciêntifica para trabalhos futuros.

Aliás, ressalta-se que para o aluno, além da metodologia e critérios científicos, o envolvimento em um grupo de pesquisa que atua em uma linha importante da Ciência da Computação e participação colaborativa em outras pesquisas do grupo em diversos níveis, tem significativo potencial para enriquecer sua formação, além de motivar seu ingresso em um Programa de Pós-Graduação em Ciência da Computação.

Referências

[1] G. Bavota and B. Russo. A large-scale empirical study on self-admitted technical debt. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 315–326, May 2016.

[2] Nanette Brown, Yuanfang Cai, Yuepu Guo, Rick Kazman, Miryung Kim, Philippe Kruchten, Erin Lim, Alan MacCormack, Robert Nord, Ipek Ozkaya, Raghvinder Sangwan, Carolyn Seaman, Kevin Sullivan, and Nico Zazworka. Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10, pages 47–52, New York, NY, USA, 2010. ACM.

[3] Nathan Cassee, Neil Ernst, Nicole Novielli, and Alexander Serebrenik. Negativity in self-admitted technical debt: How sentiment influences prioritization. 2025.

[4] Nathan Cassee, Fiorella Zampetti, Nicole Novielli, Alexander Serebrenik, and Massimiliano Di Penta. Self-admitted technical debt and comments polarity: an empirical study. *Empirical Software Engineering*, 27(139), 2022.

[5] Ward Cunningham. The wycash portfolio management system. *SIGPLAN OOPS Mess.*, 4(2):29–30, December 1992.

[6] Bruno Santos de Lima, Rogerio Eduardo Garcia, and Danilo Medeiros Eler. Toward prioritization of self-admitted technical debt: An approach to support decision to payment. *Software Quality Journal*, 30(3):729–755, 2022.

[7] Gianmarco Fucci, Nathan Cassee, Fiorella Zampetti, Nicole Novielli, Alexander Serebrenik, and Massimiliano Di Penta. Waiting around or job half-done? sentiment in self-admitted technical debt. pages 403–414, 05 2021.

[8] Hao Gu, Shichao Zhang, Qiao Huang, Zhifang Liao, Jiakun Liu, and David Lo. Self-admitted technical debts identification: How far are we? pages 804–815, 2024.

[9] Y. Guo and C. Seaman. Measuring and monitoring technical debt. In *Advances in Computers*, volume 82, pages 25–46, 2011.

[10] Philippe Kruchten, Robert L. Nord, Ipek Ozkaya, and Davide Falessi. Technical debt: Towards a crisper definition report on the 4th international workshop on

managing technical debt. *SIGSOFT Softw. Eng. Notes*, 38(5):51–54, August 2013.

[11] E. D. S. Maldonado and E. Shihab. Detecting and quantifying different types of self-admitted technical debt. In *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, pages 9–15, Oct 2015.

[12] Solomon Mensah, Jacky Keung, Jeffery Svajlenko, Kwabena Ebo Bennin, and Qing Mi. On the value of a prioritization scheme for resolving self-admitted technical debt. *J. Syst. Softw.*, 135(C):37–54, January 2018.

[13] Ariadi Nugroho, Joost Visser, and Tobias Kuipers. An empirical model of technical debt and interest. In *Proceedings of the 2Nd Workshop on Managing Technical Debt*, MTD '11, pages 1–8, New York, NY, USA, 2011. ACM.

[14] A. Potdar and E. Shihab. An exploratory study on self-admitted technical debt. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 91–100, Sept 2014.

[15] Giancarlo Sierra, Emad Shihab, and Yasutaka Kamei. A survey of self-admitted technical debt. *Journal of Systems and Software*, 152:70 – 82, 2019.

[16] S. Wehaibi, E. Shihab, and L. Guerrouj. Examining the impact of self-admitted technical debt on software quality. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 179–188, March 2016.