Um Benchmark de Ferramentas para Automatizar a Gestão de Dívida Técnica no Desenvolvimento de Software

Ana Melo[1,2], Laura Vidal[2], Ewerton Luna[2], Wylliams Santos[1], Roberta Fagundes[1]

[1]Universidade de Pernambuco (UPE)

[2]CESAR School, Recife - PE

{accm4, lbv2, eal}@cesar.school, {wbs, roberta.fagundes}@upe.br

Abstract. *Using support resources during Technical Debt (TD) management is essential to automate this process. However, there is a scarcity of discussions on the essential requirements for the effectiveness of these tools. Thus, this paper proposes a benchmark aiming to identify and compare characteristics of tools that can help automate TD management, which can be used by software development professionals. Features such as ease of use, plugin support, and impact on productivity were evaluated.*

Resumo. *Utilizar recursos de suporte durante o gerenciamento de Dívida Técnica (DT) é fundamental para automatizar esse processo. No entanto, existe uma escassez de discussões sobre os requisitos essenciais para a eficácia dessas ferramentas. Assim, esse trabalho propõe um benchmark com o objetivo de identificar e comparar características de ferramentas que possam auxiliar a automatizar o gerenciamento de DT, as quais possam ser utilizadas por profissionais no desenvolvimento de software. Características como facilidade de uso, suporte a plugins e impacto na produtividade foram avaliadas.*

1. Introdução

O termo Dívida Técnica (DT), inicialmente associado às atividades de codificação, expandiu-se para abranger todo o processo de desenvolvimento de software. Esse processo pode ser visto como uma cadeia de várias decisões que afetam a criação de um produto [Mandić et al. 2010] e, consequentemente, podem ocasionar em DT. Como consequência, DT é reconhecida como uma questão crucial para as organizações do desenvolvimento de software e desenvolvedores individuais [Candido De Melo et al. 2023].

Durante o desenvolvimento de software, mesmo com um planejamento prévio, desafios podem prejudicar a qualidade do produto final, os quais relacionam-se a complexidade de Sistemas de Informação e aos Grandes Desafios de Pesquisa em Sistemas de Informação no Brasil 2016 a 2026. Por exemplo, as datas de entrega de projetos não serem cumpridas ou critérios de qualidade não serem atingidos [Barros and Araujo 2016]. Diante desses obstáculos, a equipe de desenvolvimento muitas vezes recorre a soluções alternativas para alcançar os objetivos de curto prazo [Rios et al. 2018]. Dessa forma, gerenciar efetivamente uma DT torna-se necessário para auxiliar na qualidade do projeto.

Além disso, a gestão de DT está relacionada aos paradigmas, modelagem, design, engenharia e avaliação de sistemas de informação. Com relação ao paradigmas, por exemplo, a DT influencia na escolha de abordagens de desenvolvimento, considerando

o equilíbrio entre a entrega rápida e a qualidade do código. No design, DT impacta diretamente na arquitetura do sistema, influenciando a escalabilidade e manutenibilidade [Silva et al. 2023]. Assim, gerenciar DT em todas essas fases fortalece a sustentabilidade dos sistemas de informação ao longo do ciclo de vida do desenvolvimento.

Utilizar recursos de suporte durante o gerenciamento de DT é fundamental para automatizar esse processo. No entanto, segundo [Melo et al. 2022], ao analisar evidências presentes em 66 estudos primários, percebeu-se uma ausência de trabalhos que propõe ferramentas que possam automatizar o gerenciamento de DT. Dessa forma, a motivação para condução deste trabalho baseia-se em dois aspectos: (i) a ausência de ferramentas que possam automatizar o gerenciamento de DT; e (ii) a importância de fornecer evidências e soluções da academia para a indústria. Essa troca de conhecimento permite que a indústria se beneficie da expertise e da pesquisa realizada nas instituições acadêmicas.

Diante deste contexto, este projeto propõe um *benchmark* com o objetivo de identificar e comparar características de ferramentas que possam auxiliar a automatizar o gerenciamento de DT, as quais possam ser utilizadas por profissionais no desenvolvimento de software. Características como facilidade de uso e impacto na produtividade foram avaliadas. Ao final, foi possível concluir que parte das ferramentas impactam de maneira positiva no desenvolvimento de software e na correção de dívida técnica.

2. Trabalhos Relacionados

O trabalho de [Amanatidis et al. 2020] tinha como objetivo avaliar a concordância entre ferramentas de mensuração de DT, considerando o mesmo conjunto de código. Ao final, foi proposto um modelo que ajudará a identificar um pequeno conjunto de ”avaliações de referência”(perfis representativos de código). Esses perfis representarão trechos de código com níveis específicos de DT identificados por diferentes ferramentas.

O trabalho de [Candido De Melo et al. 2023] tinha como objetivo desenvolver um guia de apoio para auxiliar os profissionais e empresas do desenvolvimento de software na identificação e mensuração da DT de requisitos. O estudo apresenta um conjunto de diretrizes e informações que auxiliam a identificar dívida técnica, estratégias, métricas e ferramentas que facilitam a mensurar os dados necessários para a sua correção.

Após uma pesquisa informal na literatura, trabalhos relacionados foram analisados. No que se difere dos trabalhos supracitados, este estudo não objetiva apenas analisar ferramentas com foco em um tipo específico de DT. Mas realizar uma análise abrangente na área de estudo, em especial considerando todo o processo de gerenciamento de DT, ampliando a compreensão e apresentando informações técnicas sobre as ferramentas.

3. Materiais e Métodos

O *benchmark* é utilizado para comparar e definir melhores práticas de implementação de um produto. Nesta pesquisa, o *benchmark* foi utilizado como uma metodologia de avaliação qualitativa, possibilitando analisar as ferramentas de gestão de DT. O *benchmark* foi executado em três etapas: Planejamento, Execução e Análise [Perin et al. 2021].

3.1. Planejamento e Execução

Na primeira etapa, foram definidas quais estratégias iriam ser adotadas para atingir o objetivo do *benchmark* e quais características iriam ser analisadas nas ferramentas. As

estratégias adotadas foram: i) definição das palavras-chave para serem utilizadas na busca pelas ferramentas; e ii) realização de uma revisão da literatura não sistemática em busca de artigos que apresentem ferramentas para gestão de dívida técnica.

Na sequência, as características analisadas nas ferramentas foram definidas e seguem apresentadas. Estas particularidades foram escolhidas porque observou-se uma ausência desse tipo de análise na literatura [Melo et al. 2022], além de serem características que irão auxiliar na escolha de ferramentas ao problema abordado.

a) Se as ferramentas possuem funcionalidades que facilitam a execução de uma ou mais etapas do processo de gestão de dívida técnica proposto por [Li et al. 2014];

b) Se as ferramentas são utilizadas de maneira específica no âmbito de DT ou contribuem para a gestão de tarefas de forma abrangente;

c) Facilidade de uso na utilização e realização das tarefas;

d) Se as ferramentas impactam positivamente na produtividade do projeto.

Na segunda etapa do *benchmark*, referente a sua execução, baseando-se no trabalho de [Amanatidis et al. 2020], foi realizada uma pesquisa na literatura sobre ferramentas para gerenciamento de tarefas e dívida técnica no desenvolvimento de software. As palavras-chaves como ”*technical debt*”, ”*management*”, ”*tools*”e ”*software*”foram utilizadas na pesquisa. Além disso, para condução deste projeto, diferentes recursos de apoio foram utilizados, entre eles, o Google Scholar e as bases de busca IEEE Xplore e ACM Digital Library, os quais auxiliaram em uma pesquisa na literatura abrangente. E o Google Sheets para organização do *benchmark* através dos critérios pré-estabelecidos, bem como, para descrever às análises de semelhantes.

3.2. Análise e Discussão dos Resultados

Na etapa de análise, seis ferramentas foram analisadas individualmente em relação as características supracitadas. Na sequência, é fornecida uma breve descrição sobre cada uma delas. Vale ressaltar que, nem todas são utilizadas especificamente para gerenciar DT, mas na organização e controle de tarefas durante o desenvolvimento de um projeto.

(i) Jenkins: é um servidor de automação de código aberto utilizado para processos de integração e entrega contínua. Embora o Jenkins não forneça recursos diretos para gerenciamento de DT, ele pode ser integrado a outras ferramentas para dar suporte a esses aspectos. Ao executar regularmente análises de qualidade de código, o Jenkins pode fornecer informações sobre trechos do código que contribuem para o surgimento de DT.

(ii) SonarQube: é uma ferramenta de análise estática amplamente conhecida na manutenção de código-fonte. O SonarQube facilita o gerenciamento de DT ao mensurar diversos aspectos do código, como *code smells*, complexidade e manutenibilidade. Com base nesses fatores, o SonarQube calcula a quantidade de DT presente no código e fornece *insights* para que os desenvolvedores possam priorizar as ações de refatoração.

(iii) Jira: é uma ferramenta popular de gerenciamento de projetos. Embora o JIRA não seja uma ferramenta específica para lidar com qualidade de código ou DT, ele oferece recursos e integrações que podem ajudar nessas áreas. Por exemplo, é possível criar um campo personalizado para registrar e acompanhar uma dívida técnica.

(iv) FindBugs: é um analisador de código estático que detecta possíveis *bugs* e problemas

de qualidade no código em projetos Java, incluindo dívida técnica. Os bugs identificados pelo FindBugs são classificados de acordo com sua severidade, ajudando os desenvolvedores a priorizar a correção dos problemas mais críticos.

(v) Tracy-TD: é uma ferramenta que considera a perspectiva do negócio para priorizar DT. A Tracy-TD possui pontos de extensão que são próprios da organização que a utiliza, com estes elementos, são identificados os impactos potenciais de cada DT no negócio, ajudando na tomada de decisões das partes interessadas comerciais e técnicas.

(vi) CodeScene: ferramenta de análise do comportamento do código, que combina dados de controle de versão com algoritmos de aprendizado de máquina para identificar padrões sociais e riscos ocultos na base de código. A ferramenta pode identificar diversos tipos de DT, como código duplicado e design inadequado, bem como, auxiliar na priorização.

A seguir, a Figura 1 exibe as ferramentas e as informações analisadas para cada uma delas, levando em consideração, por exemplo, quais das cinco etapas do processo de gestão de DT podem ser automatizadas. Para facilitar a compreensão, os campos em verde indicam uma implementação satisfatória, os campos em amarelo indicam uma implementação insatisfatória, e o vermelho indica a ausência de implementação.

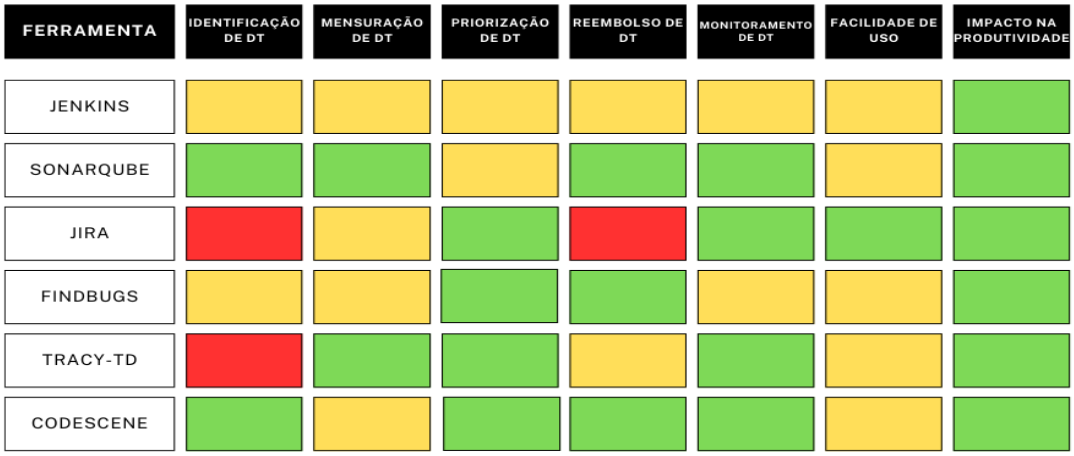

Figura 1. Análise comparativa das ferramentas.

Além da análise supracitada, esse projeto atualmente está realizando uma pesquisa técnica sobre as ferramentas. Essa análise apresenta características de implementação, a exemplo, se as ferramentas possuem suporte a *plugins* e extensibilidade, bem como, o modelo de licenciamento. Considerando ser uma análise em andamento, a Figura 2 apresenta essa comparação para quatro das seis ferramentas consideradas no estudo.

A comparação entre diversas ferramentas de gestão de DT revelou nuances essenciais para orientar a tomada de decisões estratégicas em ambientes de desenvolvimento de software. Ao analisar critérios como funcionalidades de monitoramento, priorização, e facilidade de implementação, destacaram-se diferenças entre as soluções disponíveis. Algumas ferramentas, como SonarQube e CodeScene demonstraram uma abordagem mais robusta no rastreamento e avaliação contínua de DT, enquanto outras se destacaram na capacidade de integração com os fluxos de trabalho. São análises preliminares e resultados emergentes, no entanto, essas conclusões enfatizam a necessidade de uma abordagem holística na escolha e implementação de ferramentas de gestão de dívida técnica.

FERRAMENTA	LINGUAGEM DE IMPLEMENTAÇÃO	ARQUITETURA DE SOFTWARE	SUPORTE A PLUGINS	INTEGRAÇÃO COM FERRAMENTAS	TIPO DE ANÁLISE	INDEPENDÊNCIA DE PLATAFORMA	MODELO DE LICENCIAMENTO
JENKINS	JAVA	MASTER-SLAVE	SIM	SIM	ESTÁTICA	SIM	OPEN-SOURCE
SONARQUBE	JAVA	CLIENTE-SERVIDOR	SIM	SIM	ESTÁTICA E DINÂMICA	SIM	OPEN-SOURCE
JIRA	JAVA	DESCONHECIDO	SIM	SIM	NÃO	SIM	PROPRIETÁRIO
FINDBUGS	JAVA	INDEPENDENTE	SIM	SIM	ESTÁTICA	SIM	OPEN-SOURCE

Figura 2. Análise técnica comparativa das ferramentas.

3.3. Implicações para Profissionais e Pesquisadores

Os resultados desse trabalho possuem importantes discussões para os pesquisadores e profissionais do desenvolvimento de software. No entanto, deve-se considerar que as implicações identificadas estão sujeitas a limitações de estudo, considerando que a pesquisa está em andamento. As discussões seguem apresentadas na sequência:

(i) Existe uma ausência de estudos que apresentem ferramentas que possam ser utilizadas para gerenciar DT. Encoraja-se aos pesquisadores realizarem pesquisas que dissertem informações, em especial empíricas, sobre a aplicabilidade de tais recursos;

(ii) Poucas ferramentas apresentadas neste trabalho tinham como foco específico DT, parte dos recursos são utilizados para gerenciamento de tarefas e projetos. No entanto, perante as análises realizadas, os profissionais da indústria de software podem de utilizá-los nesse escopo, adaptações deveram ser feitas de acordo com o cenário do projeto e da quantidade e severidade dos itens de dívida técnica;

(iii) As ferramentas apresentadas nesse estudo foram coletadas através de pesquisas na literatura, as quais foram realizadas de maneira inicial, até então. Além disso, os comparativos presentes nas Figuras 1 e 2 é o resultado da utilização das ferramentas pelas pesquisadoras desde trabalho durante um período de quatro meses. Dessa forma, o trabalho apresenta vieses perante a sua condução, no entanto, o protocolo do *benchmarking*, a condução das pesquisas, a utilização das ferramentas e as análises comparativas, foram realizadas por pesquisadores na área. Incluindo uma estudante do 3º período do curso de Ciência da Computação e duas professoras na área de qualidade de software e DT.

4. Considerações Finais

No cenário atual, a disponibilidade de informações sobre ferramentas de gerenciamento de DT torna-se limitada, bem como, existe uma escassez de discussões sobre os requisitos essenciais para a eficácia dessas ferramentas. Através de uma análise preliminar na literatura e um *benchmarking*, foi possível obter uma compreensão mais aprimorada das lacunas existentes em uma ferramenta que visa eficientemente gerenciar DT. Essa abordagem de pesquisa proporciona *insight*s valiosos na área de pesquisa.

A partir desse trabalho, quadros informativos que detalham as informações relacionadas à escolha de uma ferramenta para gestão de tarefas ou DT foram apresentadas.

A análise revela que as ferramentas atuais não são abrangentes o suficiente. Dessa forma, torna-se imperativo o desenvolvimento de uma ferramenta mais completa, capaz de atender aos critérios estabelecidos, visando simplificar e tornar mais prático o processo de gerenciamento de DT no mercado de desenvolvimento de software.

Este estudo encontra-se em fase inicial, os resultados são preliminares, mas é importante ressaltar que o trabalho está em evolução. O escopo futuro incluirá uma expansão das análises, explorando a eficácia e limitações de cada uma das ferramentas identificadas. Além disso, espera-se que novos softwares sejam incluídos no *benchmarking*, bem como, que estudos empíricos sejam realizados de forma a avaliar em projetos reais a eficácia das ferramentas. Ao final, espera-se apresentar a academia e indústria um corpo de conhecimento avaliado sobre recursos que possam automatizar a gestão de DT.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Referências

Amanatidis, T., Mittas, N., Moschou, A., Chatzigeorgiou, A., Ampatzoglou, A., and Angelis, L. (2020). Evaluating the agreement among technical debt measurement tools: building an empirical benchmark of technical debt liabilities. *Empirical Software Engineering*, 25:4161–4204.

Barros, M. and Araujo, R. (2016). Caminhos para o desenvolvimento de sistemas de informação resilientes.

Candido De Melo, A. C., Accioly, N., Fagundes, R., and Santos, W. (2023). Identifying and measuring technical debt in software requirements: a supporting guide. In *Proceedings of the XIX Brazilian Symposium on Information Systems*, pages 356–363.

Li, Z., Liang, P., and Avgeriou, P. (2014). Architectural debt management in value-oriented architecting. In *Economics-Driven Software Architecture*.

Mandić, V., Oivo, M., Rodríguez, P., Kuvaja, P., Kaikkonen, H., and Turhan, B. (2010). What is flowing in lean software development? In *International Conference on Lean Enterprise Software and Systems*, pages 72–84. Springer.

Melo, A., Fagundes, R., Lenarduzzi, V., and Santos, W. B. (2022). Identification and measurement of requirements technical debt in software development: A systematic literature review. *Journal of Systems and Software*, 194:111483.

Perin, A. P. J., Silva, D. E., and Valentim, N. M. C. (2021). Um benchmark de ferramentas de programação em blocos que podem ser utilizadas nas salas de aula do ensino médio. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*. SBC.

Rios, N., de Mendonça Neto, M. G., and Spínola, R. O. (2018). A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology*, 102:117–145.

Silva, R. K., Farias, K., Kunst, R., and Dalzochio, J. (2023). An approach based on machine learning for predicting software design problems. In *Proceedings of the XIX Brazilian Symposium on Information Systems*, pages 53–60.