Tracy-SATD: Um módulo de identificação de dívidas técnicas auto-admitidas através de artefatos no código *

Emerson R. D. Silva[1], **Rodrigo R. de Almeida**[1]

[1]Departamento de Ciências Exatas (DCX) - Universidade Federal da Paraíba (UFPB)
Rua da Mangueira, s/n, Companhia de Tecidos - Rio Tinto - Paraíba - Brasil

Resumo. *Dívida técnica (Technical Debt - TD) é um fato comum durante o processo de desenvolvimento de software, elas podem surgir por diversos motivos e necessitam de atenção, pois como dívidas financeiras, elas também podem ter juros. As TDs podem surgir de forma voluntária e involuntária, sendo chamadas de TDs auto-admitidas as que foram identificadas de forma voluntária. Identificar e registrar uma dívida é uma atividade que pode gerar esforço para o desenvolvedor durante o processo de desenvolvimento. Sabendo disso, esse trabalho apresenta uma solução que identifica TDs através de comentários no código, e também associa de forma automática ao item de configuração afetado por ela. Esta solução foi implementada em uma empresa de tecnologia, onde foi avaliado que ela dá indícios de melhorar o processo de identificação de TDs.*

Abstract. *Technical debt (TD) is a common fact during the software development process, they can arise for several reasons and need attention, as financial debts, they can also bear interest. TDs can arise voluntarily and involuntary, being called self-admitted TDs those that were identified voluntarily. Identifying and registering a debt is an activity that can generate effort for the developer during the development process. Knowing this, this work presents a solution that identifies TDs through comments in the code, and also automatically associates it to the configuration item affected by it. This solution was implemented in a technology company, where it was evaluated that it shows signs of improving the TD identification process.*

1. Introdução

A área de pesquisa sobre dívidas técnicas tem sido explorada há cerca de 29 anos, sendo iniciada por Cunningham (1992), onde ele apresenta em um relato de experiência em que teve que deixar partes do sistema para serem melhoradas depois, para poder realizar entregas em prazos curtos e atender mudanças no planejamento.

No âmbito do desenvolvimento de software, é comum ter mudanças no planejamento, isso ocorre devido a diversos fatores, como novos requisitos e alterações de prazos (WEHAIBI; SHIHAB; GUERROUJ, 2016). Essas mudanças muitas vezes podem causar pressão nas equipes de desenvolvimento, forçando-as a escolher soluções mais "rápidas e inadequadas", postergando as soluções planejadas para serem implementadas no futuro (KRUCHTEN et al., 2013).

*Trabalho de conclusão de curso, sob orientação do professor Rodrigo Rebouças de Almeida, submetido ao Curso de Bacharelado de Sistemas de Informação do Centro de Ciências Aplicadas e Educação (CCAE) da Universidade Federal da Paraíba, como parte dos requisitos necessários para obtenção do grau de BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Essas escolhas possibilitam a entrega, entretanto criam uma dívida técnica, e como boa parte das dívidas, existem juros para ela. Os juros podem vir de diversas formas, como: dificuldade maior da implementação devido ao crescimento do sistema, falta de conhecimento de novos desenvolvedores sobre o código onde a dívida se encontra, dificuldade de atualizar as versões de dependências muito antigas, entre outros. Esses juros das dívidas técnicas podem impactar no planejamento da organização, seja em tempo de desenvolvimento ou até financeiramente.

Dívidas técnicas podem ser definidas como sendo "tarefas atrasadas e artefatos imaturos que constituem uma 'dívida' porque incorrem em custos extras no futuro na forma de aumento do custo de mudança durante evolução e manutenção" (AVGERIOU et al., 2016).

Dado o impacto das dívidas nas organizações, são necessários processos para gerenciar estas pendências que se mantêm nos sistemas, com objetivo de permitir a tomada de decisões sobre a necessidade de eliminar uma dívida e o momento mais adequado para fazer isso ((GRIFFITH et al., 2014) apud (GUO; SEAMAN, 2011)).

Segundo Kruchten et al. (2013), estes processos de gerência de dívidas técnicas envolvem diversas atividades como: identificação, monitoramento, priorização, controle do pagamento, entre outros.

A identificação de dívidas pode ocorrer fundamentalmente de duas formas: as dívidas podem ser identificadas conscientemente, ou seja, os profissionais sabem da existência ou da necessidade de criação das dívidas; ou podem ser inconscientes, quando dívidas são criadas sem os profissionais terem ciência disso. A primeira forma é o que chamamos de dívidas auto-admitidas ("*self-admited*") (POTDAR; SHIHAB, 2014). Este artigo atua no contexto da atividade de identificação de dívidas técnicas auto-admitidas.

Dívidas técnicas surgem por diferentes causas, Rios et al. (2020) apresentam 78 delas, e destas, doze estão relacionadas a planejamento e gestão. Almeida, Treude e Kulesza (2021) também identificaram doze causas que também estão relacionadas a negócios, elas foram classificadas em três categorias, que são: Puramente-negócio, Lacuna entre negócios e tecnologia da informação (TI), e Gestão.

Casos ligados a negócios são os principais causadores de dívidas técnicas (AMPATZOGLOU et al., 2015). Por este motivo, Almeida et al. (2018) propõe que "[...] usar o gerenciamento de processos de negócios para capturar a perspectiva da empresa contribui para a priorização de dívida técnica, a fim de atender às expectativas do negócio". Isto posto, Almeida et al. (2018) apresenta Tracy, um *framework* para a gerência de dívidas técnicas orientadas a negócios, e no ano seguinte a ferramenta Tracy-TD (ALMEIDA; TREUDE; KULESZA, 2019), que implementa o *framework*.

Idealmente, o processo de identificação da dívida técnica deve gerar pouco esforço para a equipe de desenvolvimento. Sabendo disso, surge a problemática de criar uma forma com que as equipes de desenvolvimento cadastrem novas dívidas técnicas, de forma que se tenha pouco esforço.

O objetivo deste trabalho é apresentar Tracy-SATD (*Tracy Self Admitted Technical Debt*), um módulo de identificação de dívidas técnicas auto-admitidas que usa comentários no código para identificá-las. O módulo procura levar a identificação de dívidas

técnicas para próximo do desenvolvedor, visando reduzir o esforço necessário para essa atividade.

Tracy-SATD também associa automaticamente as dívidas técnicas a itens de configuração pré-configurados, ajudando a manter a consistência dos itens de configuração afetados por dívidas técnicas. Esta solução foi implementada em uma empresa de tecnologia, onde foi apresentada e feita uma observação e avaliação do seu uso.

Ademais, este trabalho está dividido nas seguintes seções: a Seção 2 trata da fundamentação teórica; a Seção 3 explica o que é Tracy-SATD; a Seção 4 explica o porquê de ter sido usado comentários para identificar as dívidas; a Seção 5 explica o fluxo de funcionamento da solução; a Seção 6 explica como funciona a identificação de itens de configuração no repositório e como as dívidas afetam eles; a Seção 7 explica como configurar Tracy-SATD; a Seção 8 apresenta os resultados obtidos neste trabalho; a Seção 9 apresenta trabalhos futuros; e por fim, a Seção 10 apresenta as considerações finais deste trabalho.

2. Fundamentação teórica

Essa sessão abordará alguns conceitos importantes para o bom entendimento deste artigo.

2.1. Gerência de dívidas técnicas orientada a negócios

Como foi dito por Kruchten et al. (2013), os processos de gerência de dívidas técnicas envolvem diversas atividades, como: identificação, monitoramento, priorização, controle do pagamento, entre outros.

Em seu trabalho, Almeida, Treude e Kulesza (2019) apresentam Tracy, um *framework* para gerência de dívidas técnicas orientadas a negócios, que busca apoiar os processos de negócios da organização utilizando de uma estrutura de priorização das dívidas técnicas com base nos ativos de TI da empresa e métricas de negócios.

O *framework* Tracy utiliza uma estrutura composta de 4 conceitos: ativos de TI, itens de configuração, fontes de valor, e as dívidas técnicas. Com isso é possível capturar os aspectos de negócios afetados pelas dívidas (ALMEIDA et al., 2021). Tais conceitos serão explicados na Seção 2.2.

Ademais, o *framework* foi usado como base para criação de uma ferramenta chamada Tracy-TD, a qual será descrita na Seção 2.3.

2.2. Itens de Configuração, Ativos de TI, Fontes de Valor e Dívidas Técnicas

A fim de exemplificar o que são ativos de TI, itens de configuração, valores de negócio e dívidas técnicas de uma determinada organização, segundo Almeida et al. (ALMEIDA; TREUDE; KULESZA, 2019), será utilizado o diagrama da Figura 1, que mostra de forma simplificada uma solução de e-commerce.

Os ativos de TI são sistemas de *softwares*, serviços ou aplicações que criam valor de negócio para a organização. No diagrama da Figura 1, destacado com a cor verde, temos a arquitetura de uma solução de *e-commerce*, que é o ativo de TI deste exemplo, pois seria o que geraria valor para uma determinada organização.

Já os itens de configuração são partes operantes ou não operantes do sistema, como: módulos, banco de dados, infraestrutura de *software*, entre outras coisas, que

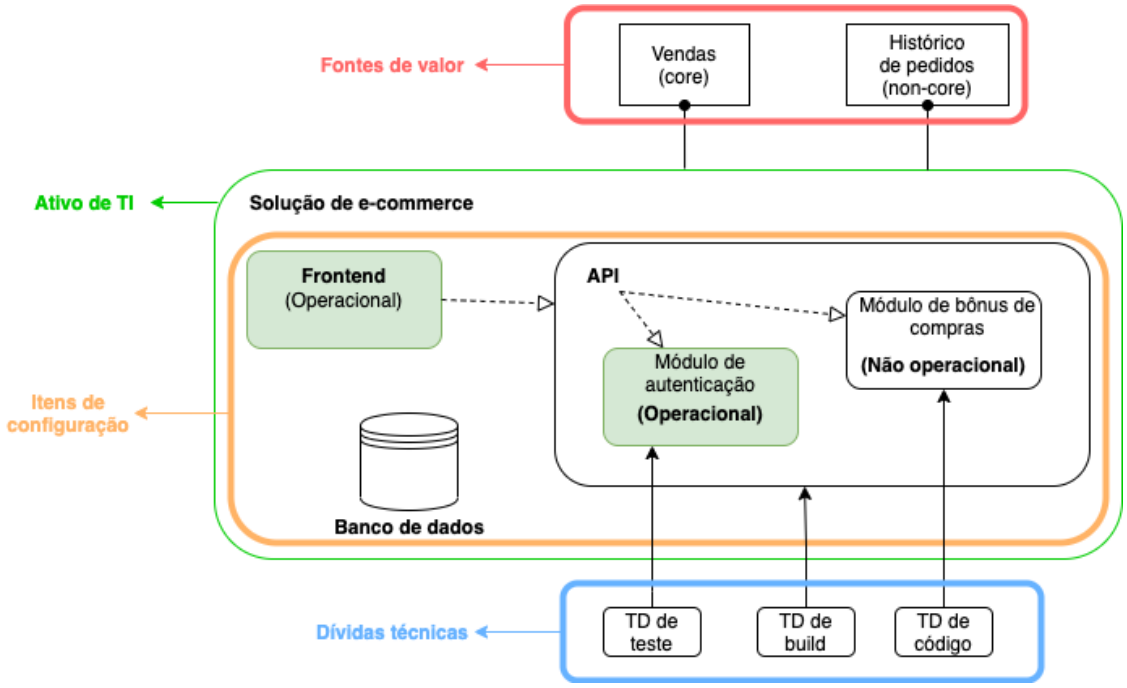


Figura 1. Exemplo de uma solução de *e-commerce* para explicar valores de negócio, ativos de TI, itens de configuração e dívidas técnicas.

podem ser afetadas por uma ou mais dívida(s) técnica(s). Entende-se que um item de configuração é operante quando ele está em pleno funcionamento, e sendo usado pelo público alvo pelo qual ele foi criado, e não operante o contrário dessa afirmação.

Ademais, itens de configuração apoiam os ativos de TI direta ou indiretamente, como por exemplo: O *front-end* da solução do *e-commerce*, ele foi criado diretamente para apoiá-lo, mas suponhamos que exista uma biblioteca externa de ícones que esteja sendo usada, essa biblioteca apoia indiretamente a solução do *e-commerce*, pois não foi feita especificamente para essa solução.

Na Figura 1, destacado em laranja, estão representados: um *front-end*, uma API(*Application Programming Interface*) e um banco de dados, ambos são itens de configuração, pois são partes operantes ou não operantes do *e-commerce*. Também pode-se observar que um item de configuração pode conter outros itens de configuração, como no caso da Figura 1, onde a API contém outros dois itens de configuração: o módulo de autenticação e o módulo de bônus de compras, sendo um operacional e o outro não, respectivamente.

Já o conceito de fontes de valor refere-se às funcionalidades do sistema, que podem ser core (principais) ou outras. Destacado de vermelho na Figura 1 temos duas fontes de valor do *e-commerce*, que são duas funcionalidades: vendas - que é *core*, principal ou que é muito importante para que o *e-commerce* gere valor - e histórico de pedidos - que também é importante, mas não é a principal fonte de valor da organização.

E por último, o conceito de dívida técnica, que é uma pendência criada para ser resolvida em algum tempo futuro, tendo sido criada para conseguir atingir alguma meta requerida no momento, como por exemplo uma entrega de software com prazo apertado.

A dívida técnica pode ser criada conscientemente, ou seja, quando a pessoa responsável sabe da existência dela, e de que ela terá que ser paga futuramente, ou pode ser criada inconscientemente, quando o desenvolvedor acaba criando-a sem saber disso.

Destacado em azul na Figura 1, tem-se 3 dívidas técnicas. A primeira dívida técnica é sobre testes - afetando o módulo de autenticação. Essa dívida pode ter surgido por falta de tempo do desenvolvedor para criar os testes, devido a prazos apertados, por falta de conhecimento na criação de testes ou outros motivos. A segunda dívida técnica do exemplo se trata do *build* da API - afetando a API como um todo, e pode ter surgido pelos mesmos motivos da primeira dívida. A terceira dívida técnica se trata de código - afetando o módulo de compras, que pode ser, por exemplo, uma classe com métodos muito grandes e demorados, com *code smells* ou outros problemas, que também podem ter surgido pelos mesmos motivos das dívidas anteriores.

2.3. Tracy-TD: Ferramenta para gerência de dívidas técnicas orientadas a negócio

A Figura 2 representa a ferramenta Tracy-TD, que implementa o *framework* Tracy de Almeida, Treude e Kulesza (2019), ela permite a gerência dos ativos de TI, itens de configuração, fontes de valor e dívidas técnicas. Além disso, ela prioriza as dívidas técnicas de acordo com métricas definidas na ferramenta, e possui integração com o Redmine e SonarQube, dentre outras funcionalidades.

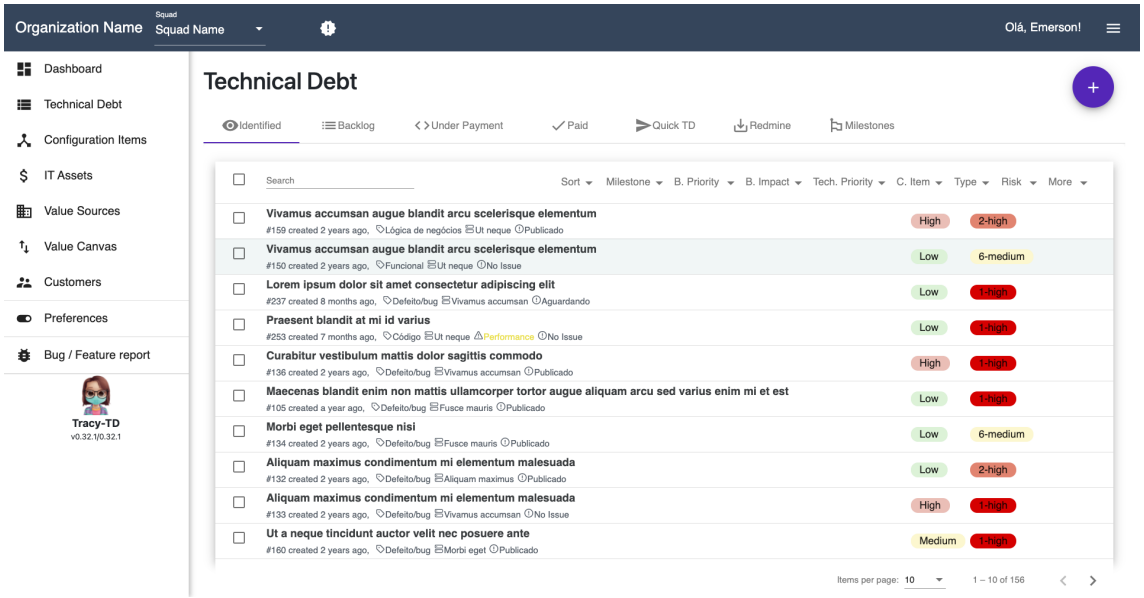


Figura 2. Visualização da listagem de dívidas técnicas de Tracy TD.

Dentre as funcionalidades de Tracy-TD, existe a gerência de *Quick Technical Debt* (*Quick TD*), que foi desenvolvida para tentar diminuir o esforço ao cadastrar dívidas técnicas, de forma a preencher menos campos do que ao cadastrar uma dívida técnica "completa".

Tanto ao cadastrar *Quick TDs*, quanto ao cadastrar uma dívida técnica "completa", é possível escolher qual o item de configuração que a dívida afeta. Isso é importante para que a gestão da organização saiba o que a dívida está impactando.

Após o cadastro da Quick TD, outra pessoa da organização com mais tempo disponível e conhecimento maior do produto pode "importar" a *Quick TD* para uma dívida técnica completa, podendo preencher outros campos, como: fontes de valor e item de configuração afetados, risco técnico, prioridade técnica, impacto técnico, esforço técnico e outros atributos da dívida técnica. Também é possível arquivar a dívida ou excluí-la.

Ao longo da experiência de implantação de Tracy-TD na empresa Phoebus Tecnologia, observou-se que manter a consistência dos itens de configuração em relação aos artefatos de software é uma tarefa trabalhosa (ALMEIDA et al., 2021), sendo algo que antes não poderia ser feito de forma automática, e que foi resolvido neste trabalho com uma solução que associa um diretório do repositório a um item de configuração em Tracy-TD.

Um exemplo dessa associação entre um item de configuração em Tracy-TD e artefato de software seria: associar um diretório Java contendo classes que fazem a autenticação de uma API ao item de configuração "Autenticação" em Tracy-TD. Além disso, o fato de associar novas dívidas manualmente ao item de configuração também é um fator que acrescenta esforço adicional às equipes de desenvolvimento.

3. Tracy-SATD: Identificando dívidas técnicas auto-admitidas e itens de configuração através de artefatos de software

O módulo Tracy-SATD foi desenvolvido baseando-se no conceito de dívida técnica auto-admitida, no qual, para essa solução, a pessoa responsável pelo artefato de *software* em desenvolvimento identifica a dívida técnica e registra a dívida como um comentário no próprio código.

O módulo funciona identificando anotações pré-definidas em forma de comentário nos códigos. Atualmente, Tracy-SATD identifica as anotações em qualquer arquivo de texto que seja passível da realização de *diff* pelo GitHub ou GitLab, como por exemplo: arquivos Java, Python, SQL, arquivos de configurações do Maven ou Gradle, arquivos de especificação YAML, entre outros.

As dívidas técnicas auto-admitidas identificadas nos códigos, por fim, são cadastradas como *Quick TDs* por Tracy-SATD, sendo possível vê-las na listagem de *Quick TDs* em Tracy-TD.

Para resolver o problema de manter a consistência das relações entre as dívidas técnicas e os itens de configuração, foi desenvolvida uma forma de associar diretórios do repositório a itens de configuração em Tracy-TD, também por meio de uma estrutura de texto predefinida em um nome de arquivo, que não possui conteúdo interno.

As dívidas técnicas identificadas estarão relacionadas ao item de configuração mais próximo, a partir do diretório atual onde o arquivo em que a dívida foi admitida se encontra, até o último diretório ascendente. Mais detalhes sobre o funcionamento da solução serão explicados na Seção 6.

4. Por que comentários no código?

Para resolver a problemática do esforço necessário para identificar uma dívida técnica foi pensado inicialmente em três soluções: Desenvolver um plugin para a IDE (*Integrated Development Environment*) que possibilitasse a identificação rápida de uma dívida técnica,

criar uma anotação usando linguagem de programação - que receberia parâmetros para identificar uma dívida - e por fim, usar comentários com padrões pré-definidos no código. Para cada uma das soluções existem prós e contras.

Desenvolver um plugin para IDE possibilitaria identificar a dívida sem deixar artefatos no repositório, mas seria complexo desenvolver e manter o mesmo plugin para diferentes IDEs (como Eclipse, Intellij, NetBeans e outras), e desenvolver somente para uma dessas IDEs criaria uma limitação.

Criar uma anotação usando linguagem de programação possibilitaria a identificação de erros de sintaxe em tempo de compilação, até mesmo durante o desenvolvimento, se a IDE utilizada oferecer este suporte. Esta possível solução limitaria a identificação de dívidas técnicas a somente linguagens de programação que possibilitam criar anotações, nas linguagens que não possuem este recurso, como Javascript, Python e outras, não seria possível identificar as dívidas.

Por fim, foi pensado na identificação usando comentários no código que seguem um padrão pré-definido, dessa forma é possível abranger a maioria dos artefatos de softwares utilizados no repositório, como: arquivos de código-fonte, arquivos SQL(*Standard Query Language*), arquivos YAML e qualquer outro arquivo de texto que seja passível da realização de *diff* pelo Git. Em contrapartida, é mais fácil de errar a sintaxe do comentário e não perceber, podendo perder a identificação da dívida.

5. Processo de identificação do comentário da dívida técnica em código

A solução pode ser dividida em 4 partes principais: O servidor de Git, a API RESTFul [1] Tracy-TD, o módulo Tracy-SATD e a lib tracy-text-processor.

As plataformas de hospedagem de código fonte GitHub e GitLab possuem um recurso chamado *webhook*. Esse recurso possibilita configurar o envio de informações via chamada POST a um endereço de recurso especificado. Esses envios ocorrem quando determinados eventos acontecem no repositório, como: eventos de *push*, de criação de *releases*, de criação de *pull-requests*, entre outros. Para a Tracy-SATD, inicialmente foi escolhido o evento de *push*, ou seja, a cada *push* em um repositório configurado, uma requisição HTTP POST é enviada.

A API Tracy-TD já existia antes deste trabalho, conforme explicado na Seção 2.3. A API foi construída em Java, utilizando o framework Spring Boot com a persistência dos dados sendo em um banco de dados relacional, o PostgreSQL.

O módulo Tracy-SATD foi desenvolvido como sendo uma nova funcionalidade de Tracy-TD API, tratando-se de um diretório interno contendo as classes Java responsáveis pelo recebimento das requisições *PUSH* dos servidores Git configurados, e pelo tratamento das dívidas identificadas em código e dos itens de configuração identificados nos diretórios.

E por fim, tracy-text-processor é uma *lib* desenvolvida em Java que auxilia Tracy-SATD a identificar no código os padrões que identificam uma dívida técnica, e nos nomes de arquivos adicionados, identificando itens de configuração. As relações entre estas

[1] O que é API REST? Disponível em: ⟨https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api⟩. Acesso em 27/06/2021

soluções estão representadas na Figura 3.

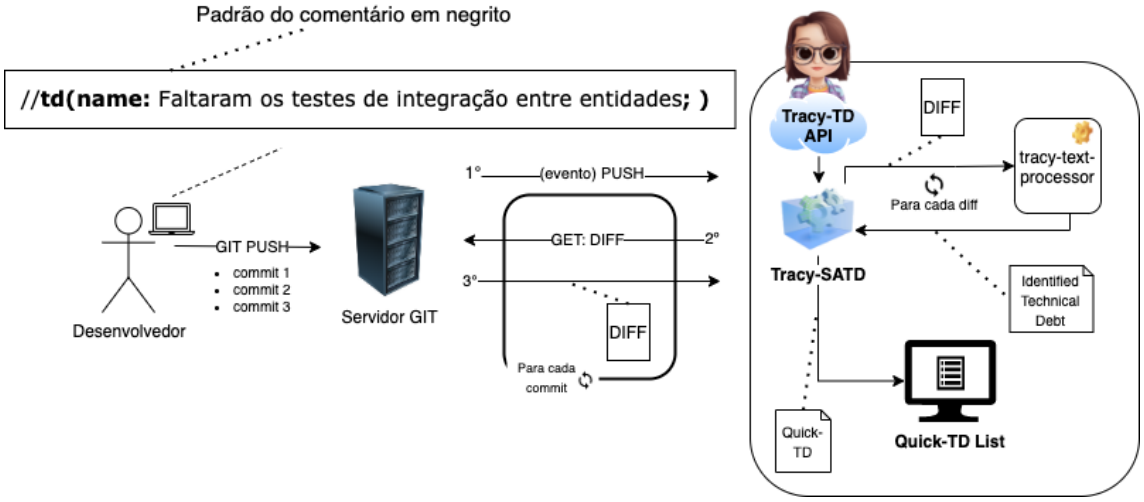


Figura 3. Relação entre os elementos que compõem a solução.

O fluxo começa após o desenvolvedor identificar a dívida técnica, a partir daí, ele deve criar um comentário identificando ela, esse comentário deve seguir o padrão apresentado na Figura 3, caso contrário não será identificado. Após isso, quando ele realizar *push* para o repositório GitHub ou GitLab configurado conforme a Seção 7, o servidor de hospedagem de código realizará uma requisição HTTP POST para o endereço da API de Tracy-TD, apontando para um endpoint do módulo de Tracy-SATD.

O módulo Tracy-SATD recebe a requisição HTTP e trata a execução de forma assíncrona, possibilitando um retorno rápido ao servidor de Git, evitando ultrapassar o tempo máximo de resposta. Após receber a requisição, Tracy-SATD verifica se o repositório está ativado nas configurações da solução, conforme explicado na Seção 7, caso esteja ativado, o módulo realiza uma requisição GET HTTP para o servidor Git de origem, para cada commit, requerendo o *DIFF* do commit.

Já com o *DIFF* do *commit*, Tracy-SATD o envia para a *lib* tracy-text-processor, a qual usa expressões regulares (RegEx - *Regular Expression*) para identificar no texto o padrão que identifica a dívida técnica, retornando um objeto contendo informações, como: nome, *path*, ID do item de configuração e condição (adicionado ou removido). Com esses objetos, Tracy-SATD passa a identificar qual o item de configuração afetado por cada dívida, para isso o módulo verifica para cada diretório acima do diretório onde o arquivo em que a dívida foi identificada está. Essa busca será melhor explicada na Seção 6.

Já com os objetos das dívidas técnicas e itens de configuração prontos, eles são persistidos no banco de dados, e para as dívidas técnicas identificadas são criadas *Quick TDs*. Com as *Quick TDs* salvas, é possível listá-las no *front-end* da Tracy-TD, possibilitando a importação como dívida técnica completa, arquivamento ou deleção da dívida, conforme explicado na Seção 2.3.

Quando um comentário que identifica uma dívida técnica auto-admitida é removido, a *Quick TD* correspondente não é alterada, sendo necessário arquivar ou deletar a *Quick TD* caso seja preciso. O módulo Tracy-SATD faz o trabalho de identificar a dívida

técnica auto-admitida, não incluindo a gerência delas, ou seja: deleção, edição, e controle de "pagamento" da dívida.

6. Associação entre Item de Configuração e diretórios no repositório

Itens de configuração são necessários para a gerência das dívidas técnicas orientadas a negócios (ALMEIDA; TREUDE; KULESZA, 2019). Sabendo disso, viu-se a necessidade de relacionar as dívidas técnicas auto-admitidas identificadas por Tracy-SATD de forma automática a os itens de configuração às quais elas afetam, pois fazer essa atividade manualmente geraria um esforço maior para o desenvolvedor.

Para que essa associação automática entre dívida técnica auto-admitida identificada e o item de configuração afetado funcione, foi criado uma relação entre um item de configuração em Tracy-TD e um diretório (pasta). Essa associação se dá por meio de um arquivo seguindo o padrão de nome: `ci-IC_ID-nome.tracy`, onde `IC_ID` é o ID do item de configuração em Tracy-TD, e em "nome" deve-se usar o nome do item de configuração de forma resumida, para identificar visualmente no repositório, conforme apresentado na Figura 4, no arquivo de cor verde. Este arquivo não possui conteúdo escrito.

Caso o padrão não seja seguido, ou o ID do item de configuração informado for inválido, o identificador do item de configuração do diretório será inválido e não servirá para ser associado às dívidas técnicas identificadas.



Figura 4. Exemplo de item de configuração em diretório e dívidas técnicas em código.

Um diretório pode ser associado a somente um item de configuração, e um item de configuração pode estar associado a vários diretórios, de um ou mais de um repositório. Idealmente, a pessoa do time de desenvolvimento que possui maior conhecimento do produto deve cadastrar os arquivos identificadores dos itens de configuração antes dos desenvolvedores começarem a codificar, para que as possíveis dívidas técnicas identificadas por eles já possam ser associadas automaticamente ao item de configuração correspondente.

Para seguir a cardinalidade da relação entre o item de configuração e o diretório, é preconizado que exista apenas um arquivo identificador do item de configuração dentro do diretório, não considerando os sub-diretórios a partir deste. Mas, caso aconteça de ser criado mais de um arquivo identificador, o último arquivo adicionado sempre será o mais atual.

No caso de não haver nenhum arquivo identificador de item de configuração nos diretórios percorridos por Tracy-SATD para encontrar o item de configuração que será associado a uma dívida, essa dívida será cadastrada da mesma forma, porém sem item de configuração afetado. Posteriormente, em Tracy-TD, ao importar a *Quick TD* gerada por essa dívida, será possível definir o item de configuração afetado.

Em Tracy-SATD o processo de associação entre uma dívida técnica auto-admitida identificada e o item de configuração afetado pode ocorrer de duas formas: pelo ID do item de configuração informado no comentário da dívida - como apresentado na Figura 4 no comentário de dívida técnica do arquivo de cor azul, e pela busca do item de configuração do diretório ascendente mais próximo - a partir do diretório onde a dívida foi identificada, como demonstrado também na Figura 4, onde a seta de cor vermelha indica que não há item de configuração no diretório atual, e em seguida, na seta de cor verde, indica que existe um item de configuração no diretório pai, o qual será usado como item de configuração afetado pela dívida técnica do arquivo de cor laranja.

Para usar o ID do item de configuração é necessário que ele exista em Tracy-TD, sendo possível obtê-lo na listagem de itens de configuração ou em detalhes do item de configuração selecionado na listagem.

Vale ressaltar que o uso do ID do item de configuração no comentário da dívida só deve ser usado em casos excepcionais, pois ele requer o conhecimento prévio do ID do item de configuração, podendo ser um fator que gera esforço na identificação. Na Figura 4 está um exemplo deste caso, onde uma dívida no arquivo de cor azul afeta um item de configuração que foi definido em outro diretório. Nesse caso em específico, não é correto adicionar o mesmo identificador de item de configuração no diretório onde o arquivo está - apesar de possível - pois é um diretório que possui vários arquivos de itens de configuração diferentes, tendo em conta que nesse exemplo, vendas (*sales*) e pedidos (*orders*) são itens de configuração.

7. Configurando a integração de Tracy-SATD com o repositório Git

É preciso realizar uma configuração inicial antes de começar a identificar as dívidas técnicas auto-admitidas, e os itens de configuração nos diretórios. É necessário que o líder da equipe de desenvolvimento ou alguém por permissão de administrador, informe um *token* válido com permissão de leitura da plataforma de Git escolhida (GitHub ou GitLab), o endereço da plataforma (caso a plataforma escolhida seja o GitLab, que permite

servidores privados) e por fim selecionar quais repositórios Tracy-SATD deve analisar. Ambas informações estão demonstradas na Figura 5.

Após essa configuração inicial, é preciso ir em cada repositório que foi selecionado e criar um *webhook*, preencher o campo de URL com o endereço da API de Tracy-TD e marcar o evento de push.

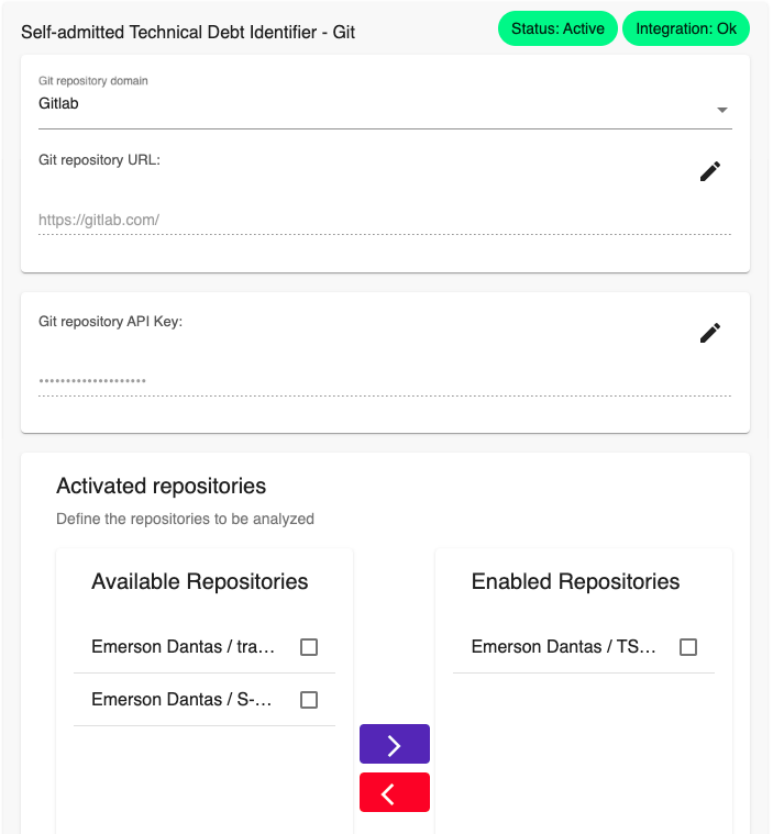


Figura 5. Tela de configuração da integração com a plataforma Git, no front de Tracy-TD.

8. Resultados

Tracy-SATD foi implementada na empresa Phoebus Tecnologia, companhia esta que atua na área de pagamentos eletrônicos, processamento de cartões de crédito e processamento de vendas.

Para que a implantação pudesse ser feita, foi preciso realizar as configurações apresentadas na Seção 7. Após configurar a integração e resolver questões de *firewall* e domínio específicas da empresa, foi criado um repositório piloto na plataforma GitLab interna da companhia, para verificar se a integração e reconhecimento das dívidas técnicas auto-admitidas estavam funcionando como esperado.

Após verificar que a integração funcionou, e as dívidas técnicas auto-admitidas e itens de configuração foram identificados, foi realizado um novo teste em um repositório real da empresa, também obtendo sucesso. Concluído a etapa de testes, foram configurados todos os repositórios principais do time de desenvolvimento.

Para que as dívidas técnicas auto-admitidas fossem associadas ao item de configuração afetado de forma automática, foram identificados itens de configuração nos repositórios de um produto da empresa, antes de apresentar a solução ao time de desenvolvimento.

Visando auxiliar no entendimento da solução, foi escrito um artigo na *wiki* interna da empresa, contendo informações diretas e exemplos de como identificar dívidas técnicas através de comentários no código. Também, neste artigo, foi feito um tutorial de como criar um template para a IDE Intellij, com a sintaxe correta do comentário. Com esse template é possível criar o comentário somente escrevendo "td" e apertando a tecla TAB, requerendo somente escrever o nome da dívida após isso. Este artigo foi compartilhado com a equipe de desenvolvimento após uma reunião por vídeo-chamada para apresentar a solução.

A solução de template de comentário - que possibilita criar o comentário com a abreviação "td" - não pode ser aplicada a todas as IDEs usadas pela equipe de desenvolvimento, ficando limitado a IDE Intellij, que é usada pela maioria na equipe de desenvolvimento. Mas, para outros tipos de tecnologias, alguns desenvolvedores preferem outras IDEs, e nesse caso, poderiam surgir erros de sintaxe do comentário.

Foi observado que no período de 1 mês, entre os dias 01/06 e 01/07 de 2021, foram identificadas 4 dívidas técnicas através de comentários no código e 1 dívida técnica identificada pelo cadastro de *Quick TDs*.

Observou-se que os desenvolvedores tiveram dificuldades ao escrever corretamente a sintaxe do comentário que identifica a dívida técnica, fazendo-os tirar dúvidas sobre a sintaxe correta no canal de comunicação da empresa.

Para avaliar a opinião da equipe de desenvolvimento sobre a solução Tracy-SATD, foi aplicado um questionário com 4 perguntas abertas, destas, 3 eram para quem já identificou alguma dívida técnica através de comentários no código, e 1 para quem ainda não identificou. O questionário pode ser visto no Apêndice A.

As perguntas abordavam a opinião dos respondentes a respeito da solução, levantando possíveis impedimentos, dificuldades e sugestões. Foram obtidas 5 respostas, de um total de 6 desenvolvedores do time. Destas, 3 são de desenvolvedores que já identificaram dívidas através de comentários de código, e 2 de quem ainda não realizou essa atividade.

De maneira geral, as respostas de quem usou a solução mostram que houve uma aceitação, e que Tracy-SATD facilitou a identificação de dívidas técnicas, tornando isso uma atividade mais prática e menos trabalhosa. Também foi identificado nas respostas que a solução de identificação de dívidas técnicas através de comentários auxilia no processo de desenvolvimento, evitando que dívidas técnicas identificadas sejam esquecidas ou ignoradas.

As duas respostas de desenvolvedores que ainda não identificaram dívidas técnicas através de comentários no código mostram que os principais motivos foram: a falta de conhecimento sobre dívida técnica, o conhecimento incipiente dos ambientes de desenvolvimento, e falta de conhecimento da solução Tracy-SATD.

9. Trabalhos futuros

Uma melhoria futura para a solução de identificação de dívidas técnicas em código, seria o desenvolvimento de um plugin simples para as principais IDEs. O plugin teria a principal função de criar o comentário da dívida com a sintaxe correta, diminuindo a chances do desenvolvedor errar o padrão de comentário.

Outras melhorias também poderiam ser feitas com plugin, como por exemplo: quando é preciso definir manualmente qual item de configuração a dívida afetará, o plugin poderia listar os itens de configuração disponíveis, a fim do desenvolvedor escolher um deles; e destacar comentários de dívidas técnicas pagas, para facilitar a remoção no código. Mas ainda assim, existiriam as complicações de desenvolver um plugin, citadas na Seção 4.

Um outro trabalho futuro seria a realização de uma avaliação de usabilidade com um número maior de participantes, visando avaliar se a solução de identificação de dívidas técnicas através de comentários no código também reduziu o esforço na identificação para um número maior de desenvolvedores, e se existe mais algum impedimento.

Ademais, poderiam ser feitos testes de integração automatizados, visando garantir uma qualidade da solução, possibilitando a identificação de erros durante a manutenção do código.

10. Considerações finais

Para tentar reduzir o esforço necessário na identificação de dívidas técnicas, foi criada uma solução chamada Tracy-SATD que usa comentários nos códigos para identificá-las. Também viu-se a necessidade de associar as dívidas técnicas alto-admitidas identificadas por essa solução a um item de configuração afetado, de forma automática. Sabendo disso, foi criada uma solução que associa um diretório a um item de configuração em Tracy-TD.

Conclui-se que a solução de identificação de dívidas técnicas auto-admitidas através de comentários no código dá indícios de que pode reduzir o esforço necessário para identificação da dívida, visto que houveram mais dívidas identificadas por comentários que pela interface de cadastro de *Quick TDs*, da ferramenta Tracy-TD, no período de tempo observado, e houve uma aceitação por parte dos desenvolvedores. Entretanto, existe um esforço inicial para se adaptar com a sintaxe do comentário que identifica a dívida.

Trabalhos futuros poderiam focar principalmente na criação semi-automática do comentário da dívida, seja usando *plugins* ou outra tecnologia, de modo a evitar que o desenvolvedor erre a sintaxe do comentário e que possibilite um esforço ainda menor na identificação da dívida técnica.

Referências

ALMEIDA, R. R. de et al. *Aligning Technical Debt Prioritization with Business Objectives: A Multiple-Case Study*. 2018. 2

ALMEIDA, R. R. de et al. *Business-Driven Technical Debt Prioritization: An Industrial Case Study*. 2021. 3, 6

ALMEIDA, R. R. de; TREUDE, C.; KULESZA, U. *Tracy: A Business-driven Technical Debt Prioritization Framework*. 2019. 2, 3, 5, 9

ALMEIDA, R. R. de; TREUDE, C.; KULESZA, U. *Beyond tight deadlines: what are the business causes for technical debt?* 2021. 2

AMPATZOGLOU, A. et al. The financial aspect of managing technical debt: A systematic literature review. *Information and Software Technology*, v. 64, p. 52–73, 2015. ISSN 0950-5849. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0950584915000762⟩. 2

AVGERIOU, P. et al. Managing technical debt in software engineering (dagstuhl seminar 16162). *Dagstuhl Reports*, v. 6, 01 2016. 2

CUNNINGHAM, W. The wycash portfolio management system. *SIGPLAN OOPS Mess.*, Association for Computing Machinery, New York, NY, USA, v. 4, n. 2, p. 29–30, dez. 1992. ISSN 1055-6400. Disponível em: ⟨https://doi.org/10.1145/157710.157715⟩. 1

GRIFFITH, I. et al. A simulation study of practical methods for technical debt management in agile software development. In: *Proceedings of the Winter Simulation Conference 2014*. [S.l.: s.n.], 2014. p. 1014–1025. 2

GUO, Y.; SEAMAN, C. A portfolio approach to technical debt management. In: *Proceedings of the 2nd Workshop on Managing Technical Debt*. New York, NY, USA: Association for Computing Machinery, 2011. (MTD '11), p. 31–34. ISBN 9781450305860. Disponível em: ⟨https://doi.org/10.1145/1985362.1985370⟩. 2

KRUCHTEN, P. et al. Technical debt: Towards a crisper definition report on the 4th international workshop on managing technical debt. *SIGSOFT Softw. Eng. Notes*, Association for Computing Machinery, New York, NY, USA, v. 38, n. 5, p. 51–54, ago. 2013. ISSN 0163-5948. Disponível em: ⟨https://doi.org/10.1145/2507288.2507326⟩. 1, 2, 3

POTDAR, A.; SHIHAB, E. An exploratory study on self-admitted technical debt. In: *2014 IEEE International Conference on Software Maintenance and Evolution*. [S.l.: s.n.], 2014. p. 91–100. 2

RIOS, N. et al. The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from brazil. *Empirical Software Engineering*, v. 25, 09 2020. 2

WEHAIBI, S.; SHIHAB, E.; GUERROUJ, L. Examining the impact of self-admitted technical debt on software quality. In: *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. [S.l.: s.n.], 2016. v. 1, p. 179–188. 1

Apêndice A - Avaliação da solução de identificação de dívidas técnicas através de comentários no código - Tracy-SATD

Apresentação

Olá, você foi convidado(a) para participar como voluntário em uma pesquisa de avaliação da solução de identificação de dívidas técnicas através de comentários no código, chamada Tracy-SATD. O objetivo desta avaliação é identificar pontos positivos e negativos desta solução. Esta pesquisa é de natureza acadêmica e está sendo conduzida pelo aluno de Graduação em Sistemas de Informação Emerson Dantas, sob orientação do Prof. Dr. Rodrigo Rebouças. Os dados fornecidos serão publicados apenas de forma agregada no trabalho de conclusão de curso de Bacharelado em Sistemas de Informação na UFPB do referido aluno. Por razões de ética e privacidade, a identificação do respondente nesta pesquisa não será realizada. Garantimos que qualquer informação ou elemento que possa identificá-lo será mantida em sigilo, e tampouco os dados serão divulgados individualmente, a fim de preservar a identidade dos participantes. Ao participante também é reservado o direito de retirar o seu consentimento a qualquer momento, mesmo após o preenchimento dos dados, sem precisar justificar, e sem que isto leve a qualquer prejuízo. Em caso de dúvida, entrar em contato com o aluno pelo e-mail abaixo. Estimamos que o questionário levará entre 3 e 5 minutos para ser preenchido, e o questionário permanecerá aberto para respostas apenas até o dia 02/07/2021. Agradecemos a sua contribuição!

Emerson Dantas - emerson.ruan@dcx.ufpb.br

Rodrigo Rebouças - rodrigor@dcx.ufpb.br

Seção I - Identificando se o participante já identificou dívidas técnicas através de comentários no código

Pergunta objetiva 1: Você reportou alguma dívida usando comentário de código?

Alternativas: Sim ou Não

Seção II - Obtendo a avaliação do participante que já identificou dívidas técnicas através de comentários no código

Pergunta aberta 1: O que você achou da solução de identificar dívidas técnicas através de comentários no código?

Pergunta aberta 2: Você teve algum impedimento ou dificuldade para usar a solução de identificação de dívidas técnicas através de comentários no código? Caso sim, quais foram?

Pergunta aberta 3: Tem alguma sugestão de melhoria? Caso sim, qual, ou quais?

Seção III - Obtendo o parecer do participante que ainda não identificou dívidas técnicas através de comentários no código

Pergunta aberta 1: Qual o principal motivo de não ter identificado dívidas técnicas através de comentários no código?