MESTRADO EM SISTEMAS E COMPUTAÇÃO

SANDRO LUIZ ABREU CAMPOS DOS SANTOS

MONITORAMENTO DA CONTRIBUIÇÃO DE EQUIPES DE DESENVOLVIMENTO NA EVOLUÇÃO DE ITENS DA DÍVIDA TÉCNICA EM PROJETOS DE SOFTWARE

Salvador
2018

SANDRO LUIZ ABREU CAMPOS DOS SANTOS

MONITORAMENTO DA CONTRIBUIÇÃO DE EQUIPES DE DESENVOLVIMENTO NA EVOLUÇÃO DE ITENS DA DÍVIDA TÉCNICA EM PROJETOS DE SOFTWARE

Dissertação apresentada ao Programa de Pós-graduação em Sistemas e Computação da UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial para a obtenção do grau de Mestre em Sistemas e Computação.

Orientador: Prof. Dr. Rodrigo Oliveira Spínola.

Salvador
2018

Ficha Catalográfica
(Elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador, Laureate International Universities)

Santos, Sandro Luiz Abreu Campos dos

Monitoramento da contribuição de equipes de desenvolvimento na evolução de itens da dívida técnica em projetos de software / Sandro Luiz Abreu Campos dos Santos. .-- Salvador: UNIFACS, 2018.

96 f. : il

Dissertação apresentada ao Programa de Pós-graduação em Sistemas e Computação da UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial para a obtenção do grau de Mestre em Sistemas e Computação.
Orientador: Prof. Dr. Rodrigo Oliveira Spínola.

1. Desenvolvimento de *software*. 2. Dívida técnica. I. Spinola, Rodrigo Oliveira, orient. II. Título.

CDD.004.6

SANDRO LUIZ ABREU CAMPOS DOS SANTOS

MONITORAMENTO DA CONTRIBUIÇÃO DE EQUIPES DE DESENVOLVIMENTO NA EVOLUÇÃO DOS ITENS DA DÍVIDA TÉCNICA EM PROJETOS DE SOFTWARE

Dissertação aprovada como requisito final para obtenção do grau de Mestre em Sistemas e Computação, da UNIFACS Universidade Salvador, Laureate International Universities, pela seguinte banca examinadora:

Rodrigo Oliveira Spínola – Orientador _____
Doutor em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro - UFRJ
UNIFACS Universidade Salvador, Laureate International Universities

Ana Patrícia Fontes Magalhães Mascarenhas _____
Doutora em Ciências da Computação pela Universidade Federal da Bahia - UFBA
UNIFACS Universidade Salvador, Laureate International Universities

Renato Lima Novais _____
Doutor em Ciência da Computação pela Universidade Federal da Bahia – UFBA/UNIFACS
Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)

Salvador, 27 de abril de 2018.

A Deus, minha esposa, filha, família, amigos, colegas e orientador que contribuíram direta ou indiretamente pela realização deste trabalho.

AGRADECIMENTOS

A Deus que ilumina meus caminhos e me ajuda a superar os obstáculos.

A minha esposa por ser parceira de tantas conquistas.

A minha filha que está no ventre de sua mãe e já me proporciona grandes mudanças.

Aos meus pais por todos os incentivos que me direcionaram para esta área.

Aos irmãos e demais familiares pelo apoio.

Ao meu orientador por ser um parceiro nesta jornada.

A Thiago Souto Mendes pela integração com a equipe Visminer e demais colaborações.

Aos amigos e colegas por direta ou indiretamente contribuírem com esta realização.

RESUMO

O ambiente de desenvolvimento de software é cercado de desafios. Além de prazos curtos e orçamentos restritos, diferentes níveis de conhecimento e experiência dos desenvolvedores são fatores que contribuem para o surgimento de má qualidade no processo de desenvolvimento do software. Seja através de decisões tomadas de maneira consciente ou acidental, a falta de boas práticas favorece o surgimento da Dívida Técnica (DT). O processo de identificação e monitoramento da presença da DT é tão importante quanto a investigação dos envolvidos no surgimento, para tentar entender as principais causas e reduzir impactos negativos. Este trabalho relaciona conhecimentos sobre DT e desenvolvedores com objetivo de identificar e monitorar como os envolvidos no processo de desenvolvimento do software contribuem para a evolução da DT presente no código-fonte do projeto. Neste contexto, este trabalho propõe uma estratégia para o monitoramento da evolução da DT e como os desenvolvedores contribuíram para isso. Também apresenta a ferramenta VisminerTD, que possui uma série de funcionalidades para apoiar a aplicação da estratégia. Para avaliar a estratégia, foi planejado e executado um estudo de caso. Os resultados obtidos apontam evidências positivas em relação ao uso da estratégia proposta para monitorar a evolução da DT, podendo fundamentar a necessidade de qualificação ou remanejamento das pessoas envolvidas no processo de desenvolvimento do software, com o propósito de evitar problemas futuros.

Palavras chaves: Dívida técnica. Gerenciamento de projetos. Monitoramento de desenvolvedores. Equipe de desenvolvimento. VisminerTD.

ABSTRACT

The software development environment is surrounded by challenges. In addition to short deadlines and restricted budgets, different levels of developer knowledge and experience are contributing factors to poor quality in the software maintenance process. Through decisions made consciously or accidentally, the lack of good practices favors the emergence of Technical Debt (TD). The process of identifying and monitoring DT's presence is as important as investigating those involved in the emergence, in an attempt to understand the main causes and reduce negative impacts. This work relates knowledge about DT and developers with the objective of identify and monitoring how those involved in the software development process contribute to the evolution of DT present in the project source code. In this context, this paper proposes a strategy for monitoring DT's evolution and how the developers contributed to it. It also presents the VisminerTD tool, which has many functionalities to support the application of the strategy. As an evaluation of the strategy, a case study was elaborated and executed. The results pointed to positive evidence regarding the use of the proposed strategy to monitoring the evolution of the DT, being able to substantiate the need to qualify or relocate the people involved in the software development process, in order to avoid future problems.

Keywords: Technical debt. Project management. Monitoring of developers. Development team. VisminerTD.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASA	Análise Estática Automática
DT	Dívida Técnica
LOC	Linhas de Código
MS	Mapeamento Sistemático
RM	RepositoryMiner
TAM	Technology Acceptance Model

SUMÁRIO

Capítulo 1 - Introdução

Neste capítulo são apresentados o contexto do trabalho, o que motivou essa pesquisa e seus objetivos. São também apresentados a metodologia de pesquisa utilizada, o histórico do trabalho realizado e como este texto está estruturado.

1.1 INTRODUÇÃO

Todo ambiente de desenvolvimento de software enfrenta desafios. Além das constantes restrições de prazo e custo para execução do projeto, há também uma diversidade de níveis de conhecimento e experiência da equipe que podem levar à ocorrência da Dívida Técnica (DT) nos projetos de software. DT é uma metáfora para artefatos incompletos ou inadequados no projeto de software indicando a necessidade de esforço extra para o seu pagamento em um momento futuro (SEAMAN *et al.*, 2011).

O surgimento da DT pode ocorrer por diferentes motivos, por exemplo, quando desenvolvedores violam boas práticas de arquitetura ou de codificação, produzindo falhas estruturais no código (CURTIS *et al.*, 2012). A DT é classificada em duas formas básicas: não intencional e intencional. A dívida não intencional geralmente ocorre devido à má qualidade de práticas aplicadas na atividade de codificação (McConnel, 2008) ou devido à inexperiência dos desenvolvedores (CURTIS *et al.*, 2012). Em outras situações, a DT pode ser inserida de forma intencional. Um exemplo é quando a equipe decide assumir o risco de comprometer a qualidade do software ao antecipar uma entrega sem a conclusão dos artefatos esperados de maneira adequada. Esta escolha traz benefícios a curto prazo mas pode levar ao surgimento de custos maiores a longo prazo.

É importante que exista um procedimento para identificar a DT presente no projeto de software e, dessa forma, avaliar seu impacto (NUGROHO *et al.*, 2011). Além disso, segundo Seaman e Guo (2011), cabe aos gerentes de projeto tomarem decisões sobre em qual momento e quais itens de dívida devem ser resolvidos/pagos ou mantidos. Assim, as equipes de software precisam de um método rápido e fácil de quantificar a dívida existente. Em complemento, como o sucesso de um projeto de software depende do acompanhamento eficaz do time de desenvolvimento (FRANÇA *et al.,* 2008), é importante que haja uma relação entre a DT presente no projeto e os responsáveis pelo seu surgimento.

Uma estratégia que apoie a identificação e monitoramento da DT, considerando informações sobre sua origem e responsável, pode ajudar na compreensão da qualidade do software, criando um cenário favorável para que os responsáveis pelo projeto tomem melhores decisões para minimizar problemas atuais e futuros associados à presença da dívida. Neste contexto, esta dissertação compreende a definição de uma estratégia de gestão da DT, apoiando o monitoramento da evolução da dívida e a tomada de decisão em busca de melhorias da equipe de desenvolvimento e da qualidade do software.

1.2 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

Diante da evolução constante presente no ciclo de vida de desenvolvimento de um software, é uma tarefa difícil lidar com prazos que geralmente são curtos. Segundo Cunningham (1992), atender aos prazos curtos e pendentes é como entrar em um processo de dívida. Através da decisão de assumir dívidas ao longo do projeto, o desenvolvimento poderá ser acelerado, porém assume-se um risco de não pagá-la a tempo e arcar-se com juros. Somando-se isso ao fato de que cada desenvolvedor tem seus próprios comportamentos e características incorporados aos seus hábitos de codificação (LATOZA *et al.*, 2016), a qualidade de software está diretamente associada também ao seu desempenho (OSTRAND *et al.*, 2010).

Os membros de um time de desenvolvimento de um projeto de software são as peças fundamentais para o sucesso do software (DeMarco, 1997). Porém, em um cenário onde há colaboradores com diferentes níveis de conhecimento e experiências, as chances para o surgimento da DT tornam-se maiores (LATOZA *et al.*, 2016).

O surgimento de DT pode ser inevitável e ao mesmo tempo bem-vinda (KRUCHTEN *et al.,* 2012). Saber da existência de todas as incidências de DT pode não ser uma tarefa rápida e fácil, uma vez que sua inserção pode ocorrer também de forma não intencional. Considerando que o time de desenvolvimento saiba sobre a existência e consequência da DT, torna-se mais fácil administra-la (KRUCHTEN *et al.,* 2012), além de ser possível saber como cada membro da equipe contribui com a sua manutenção (NUGROHO *et al.*, 2011). Obter dados históricos sobre a evolução de código fonte de cada desenvolvedor pode apoiar o monitoramento da evolução da DT. De posse destas informações, os gerentes de projeto poderiam atuar na capacitação e/ou remanejamento dos desenvolvedores envolvidos, de modo a melhorar a qualidade da equipe e consequentemente do software.

Como a DT está diretamente relacionada com a qualidade do software, a ponto de reduzir a velocidade das equipes de desenvolvimento durante a evolução do software (Amanatidis *et al*., 2017), é fundamental que a identificação e monitoramento sejam constantes, de forma que o gerente consiga tomar decisões para diminuir os impactos dos problemas ocasionados pela DT. Contudo, Alves *et al.* (2016) realizaram um mapeamento sistemático sobre identificação e gerenciamento da DT e não foi identificado nenhuma abordagem de gerenciamento da DT com foco na equipe de desenvolvimento.

1.3 OBJETIVOS

O presente trabalho tem como objetivo **definir uma estratégia que permita monitorar a contribuição de equipes de desenvolvimento no acúmulo e pagamento de itens da DT em projetos de software**.

Para alcançar o propósito do trabalho, foi necessário definir três objetivos específicos:

a) **Pesquisar na literatura sobre DT e indicadores que podem apoiar a sua identificação a partir do código fonte do projeto de software**. Atender a esse objetivo é importante pois é preciso ter domínio sobre a DT para que seja possível a identificação de itens da dívida e, em um momento posterior, favorecer a definição de uma estratégia de monitoramento de itens da DT. Os resultados para esse objetivo estão descritos no Capítulo 2.

b) **Pesquisar na literatura quais são os critérios que permitem monitorar a contribuição de desenvolvedores frente ao processo de evolução do software.** É relevante identificar quais são os critérios que sustentam a contribuição dos desenvolvedores ao longo do processo de desenvolvimento do software. Os resultados para esse objetivo estão descritos no Capítulo 3.

c) **Definir uma estratégia que permita investigar como os desenvolvedores contribuem para a presença da DT no processo de evolução do software**. Para alcançar este objetivo, foi desenvolvida uma estratégia baseada no estudo apresentado no capítulo 3, que propõe o monitoramento da DT por parte do desenvolvedor, conforme o seu histórico de contribuições no projeto. Os resultados para esse objetivo estão descritos no Capítulo 4.

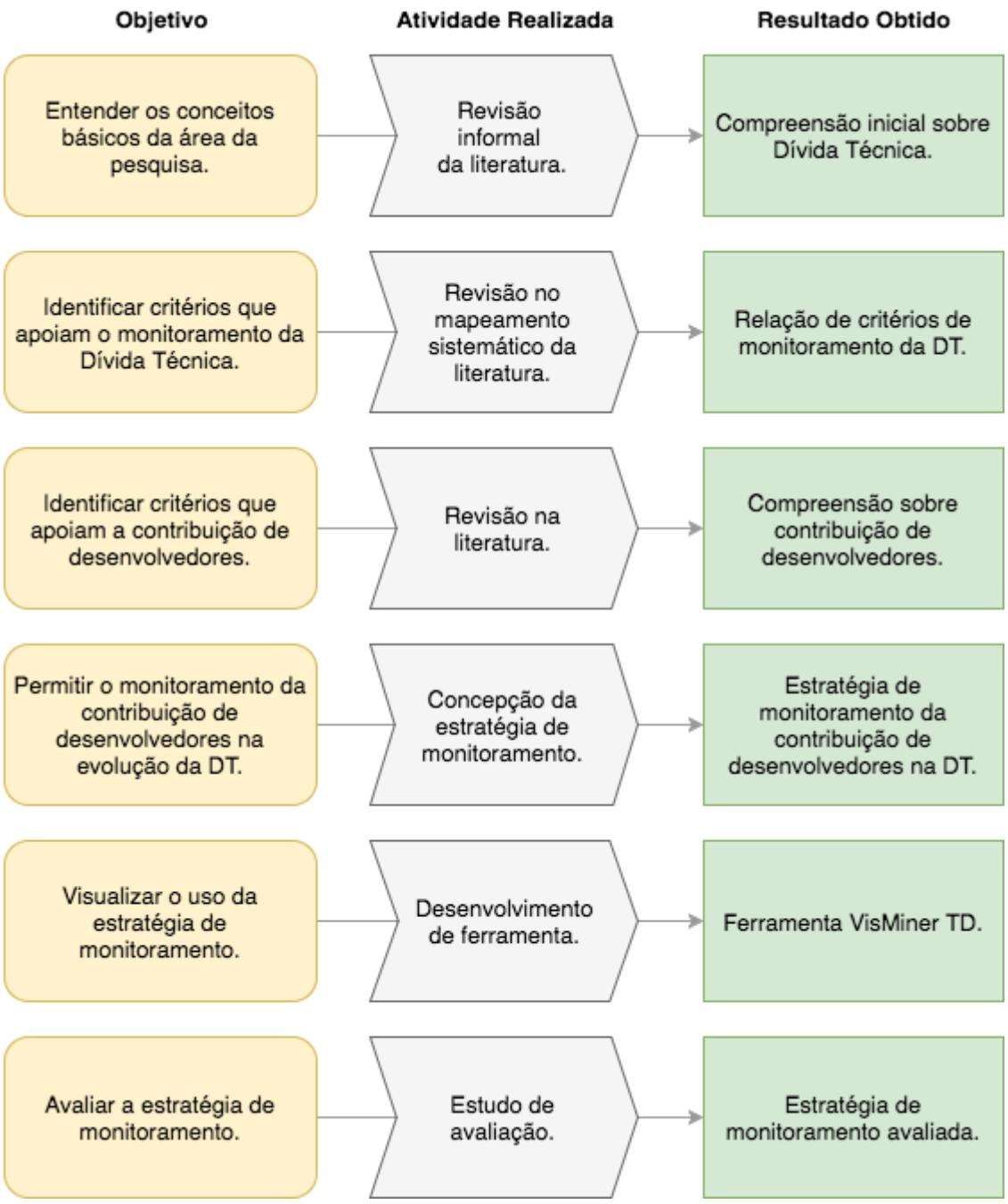
d) **Desenvolver uma ferramenta para apoiar a utilização da estratégia de monitoramento proposta.** Para alcançar este objetivo, foi desenvolvido um software que viabiliza o uso da estratégia. Os resultados alcançados para esse objetivo são descritos no Capítulo 4.

e) **Avaliar a estratégia de monitoramento para que seja possível analisar a viabilidade de sua aplicação e verificar se ela contribui positivamente no processo de tomada de decisão no gerenciamento do time de desenvolvimento.** Para alcançar este objetivo, foi planejado e executado um estudo de caso. Os resultados alcançados para esse objetivo são descritos no Capítulo 5.

1.4 METODOLOGIA E HISTÓRICO DA PESQUISA

Para atingir os objetivos definidos, foram delineadas seis atividades, apresentadas com os respectivos resultados obtidos na Figura 1.

Figura 1 - Estratégia de pesquisa



1.5 ESTRUTURA DA DISSERTAÇÃO

Este capítulo apresentou algumas das ideias que motivaram o desenvolvimento desta dissertação, seu objetivo e a metodologia de pesquisa utilizada para a realização desta pesquisa. A organização do texto deste trabalho segue a linha de tempo em que as atividades foram realizadas, conforme a Figura 1, e está estruturada segundo os itens abaixo:

Capítulo 1. Introdução: contextualiza e descreve os motivos para a realização deste trabalho, juntamente com os objetivos que o trabalho pretende alcançar;

Capítulo 2. Dívida Técnica: discute conceitos e definições sobre dívida técnica;

Capítulo 3. Contribuição de desenvolvedores em projetos de desenvolvimento de software: apresenta o planejamento e os resultados alcançados com a execução de uma revisão da literatura cujo objetivo foi identificar as dificuldades ao se trabalhar com o gerenciamento de equipes de desenvolvimento;

Capítulo 4. Uma estratégia para o monitoramento da contribuição de desenvolvedores na evolução da dívida técnica: descreve uma estratégia que permite monitorar a evolução dos itens de DT que os desenvolvedores inserem durante o processo de desenvolvimento do software;

Capítulo 5. Estudo da estratégia para o monitoramento da evolução da dívida técnica: apresenta o planejamento, execução e análise dos resultados de um estudo de caso com o intuito de avaliar a estratégia proposta no capítulo 4;

Capítulo 6. Considerações finais: discute as considerações finais do trabalho apresentando as principais contribuições alcançadas, limitações e sugestões para pesquisas futuras.

Além disto, este trabalho contém o seguinte apêndice:

APÊNDICE A – Instrumentos utilizados no estudo apresentado no capítulo 5: apresenta os formulários utilizados no estudo de caso executado neste trabalho, presente no Capítulo 5.

Capítulo 2 – Dívida Técnica

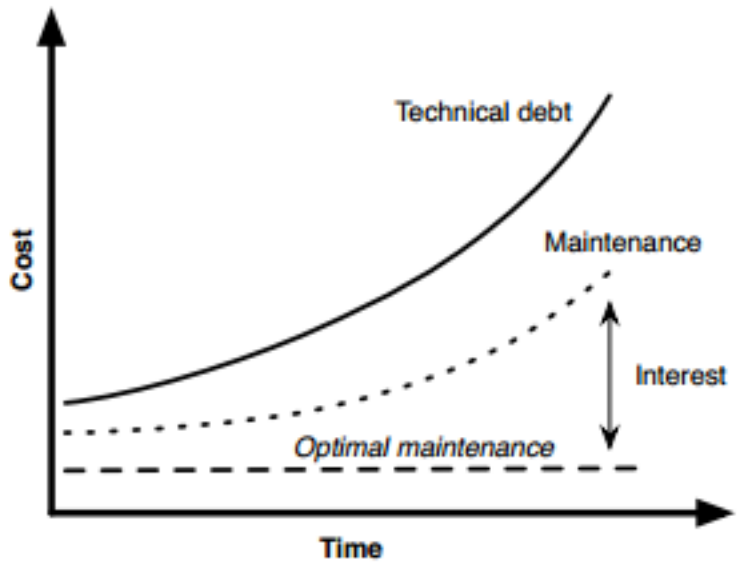
Este capítulo apresenta conceitos introdutórios sobre Dívida Técnica. Dentre eles, explicações sobre o que é dívida técnica, seus tipos, sua identificação e monitoramento.

2.1 INTRODUÇÃO

Dívida técnica pode ser definida como o custo de ajuste em problemas de qualidade estrutural no desenvolvimento de um software, que precisam ser controlados ou eliminados para evitar problemas maiores (CURTIS *et al.*, 2012).

Assim como na visão da dívida financeira, a DT pode sustentar o pagamento de juros, presente como esforço extra a ser investido para eliminar os itens de dívida no futuro. É possível optar por continuar pagando os juros ou concentrar maiores esforços a fim de quitar toda a dívida de uma única vez. Embora tenha-se os custos para saldar a dívida, há o ganho ao reduzir juros futuros (FOWLER, 2004). Através da Figura 2 é possível identificar que uma DT não tratada resulta no crescimento de custos a longo do tempo. Tal ascensão eleva também os custos da manutenção, distanciando-a de um nível ideal.

Figura 2 - Crescimento da dívida técnica se não for resolvida



A DT está relacionada a uma série de problemas práticos como os altos níveis de retrabalho, dificuldade de atingir critérios de qualidade, além da desmotivação da equipe de desenvolvimento em continuar evoluindo o software (MENDES *et al.*, 2016; KRUCHTEN *et*

al., 2012; SPÍNOLA *et al.*, 2013). Com o gerenciamento da DT, é possível saber sua dimensão, a proporção do crescimento, quanto de juros a equipe está pagando e quais são as consequências em manter uma DT para uma futura manutenção (NUGROHO *et al*., 2011).

2.2 TIPOS DE DÍVIDA TÉCNICA

A metáfora dívida técnica foi elaborada inicialmente por Ward Cunnigham, no ano de 1992, com o objetivo de retratar de forma superficial, a falta de qualidade do software em razão da produtividade em curto prazo (CUNNINGHAM, 1992). Com o passar dos anos, surgiram inúmeras pesquisas e discussões científicas sobre DT.

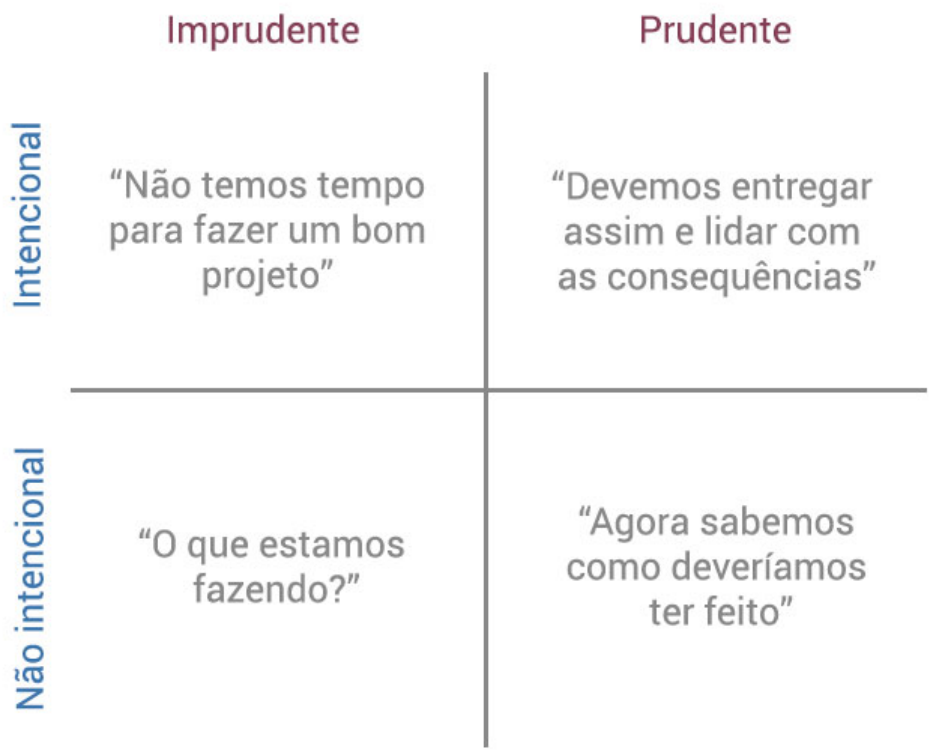
DT se refere a artefatos incompletos ou inadequados no projeto de software indicando a necessidade de esforço adicional para o seu pagamento em um momento futuro (SEAMAN; GUO, 2011). De acordo com os autores, o atalho para a entrega mais rápida do software com DT pode gerar problemas de custos extras no futuro. A maior dificuldade dos gerentes de projetos é encontrar o equilíbrio entre optar pela aquisição da DT e os custos agregados aos esforços extras para o seu pagamento.

A DT é classificada em duas formas básicas: não intencional e intencional. O surgimento da DT não intencional ocorre devido à má qualidade de práticas no processo de codificação do software (MCCONNEL, 2008), devido a inexperiência dos desenvolvedores ou à complexidade do projeto (CURTIS *et al.*, 2012). Esse tipo de DT exige mais atenção, pois como podem passar despercebidas, tornam-se mais difíceis de serem identificadas e monitoradas.

Em outros momentos a DT pode ser inserida de forma intencional, através de decisões informadas. Segundo McConnel (2007), a DT intencional de curto prazo é aquela que o desenvolvedor assume por motivos estratégicos, como a antecipação do lançamento de um produto. Essa escolha traz benefícios a curto prazo, mas pode estimular o surgimento de maiores custos a longo prazo.

Em outra classificação, Fowler (2009) elaborou um quadrante que caracteriza as naturezas da DT com o objetivo de identificar se ela pode ser o resultado da falta de cuidado ou inserida com precaução. Chamado de *Technical Debt Quadrant* (presente na Figura 3), ele classificou os tipos como imprudente/prudente e adicionou as características deliberado/inadvertido (intencional/não intencional).

Figura 3 – Technical Debt Quadrant



Fonte: Adaptado de Fowler (2009).

O primeiro quadrante representa a DT considerada como imprudente e intencional, surgindo quando a equipe decide não adotar boas práticas de codificação, por exemplo, para efetuar uma entrega mais rápida. No segundo quadrante, a DT é considerada como prudente e intencional e representa as dívidas que são inseridas com responsabilidade, a exemplo de quando a equipe tem consciência da dívida que será paga em outro momento.

O terceiro quadrante representa a DT imprudente e não intencional e traz grande risco ao projeto pelo fato de que a equipe desconhece o problema e assume a dívida sem saber. No quarto quadrante, a DT é considerada como prudente e não intencional. Neste exemplo, a equipe não tinha noção do surgimento da DT e ao tomar conhecimento, faria de forma diferente.

Por fim, segundo Alves *et al*. (2016), a DT pode ser ainda classificada de acordo com a fase do ciclo de vida do software nos seguintes tipos:

a) Dívida de Arquitetura: refere-se à dívida que tem problemas encontrados na arquitetura do software, como por exemplo, violação de modularidade.

b) Dívida de Documentação: refere-se a obstáculos encontrados na documentação de projetos e software.

c) Dívida de Teste: refere-se à dívida com problemas encontrados em rotinas de testes que afetam a qualidade dessas atividades, a exemplo de baixa cobertura.

d) Dívida de Automação de Testes: refere-se a questões envolvidas na automatização de testes funcionais previamente elaboradas para facilitar o ciclo de desenvolvimento e a integração contínua.

e) Dívida de Build (Construção): diz respeito a problemas que tornam a tarefa de compilação mais longa e difícil.

f) Dívida de Código: refere-se a problemas encontrados no código fonte que podem afetar a legibilidade do código, a exemplo de más práticas de codificação.

g) Dívida de Processo: refere-se a processos inócuos, a exemplo de algum processo que não seja mais o apropriado para as atividades atuais da organização.

h) Dívida de Defeito: diz respeito a defeitos conhecidos, geralmente identificados pelas atividades de teste ou através do usuário e reportados em sistema de rastreamento de defeitos, mas que são adiados para serem corrigidos futuramente.

i) Dívida de infraestrutura: refere-se a problemas na infraestrutura que podem atrapalhar algumas atividades de desenvolvimento.

j) Dívida de Versionamento: diz respeito a problemas de versionamento do código fonte como uso desnecessário de *forks*.

k) Dívida de Requisito: refere-se a decisões tomadas com relação a quais requisitos a equipe de desenvolvimento precisa implementar ou como fazê-los.

l) Dívida de Pessoas: diz respeito a problemas relacionados com pessoas que podem atrasar ou até mesmo impedir a realização de atividades de desenvolvimento.

m) Dívida de Projeto: refere-se à dívida que pode ser encontrada na arquitetura como violações de bons princípios de orientação a objetos.

n) Dívida de Serviço: diz respeito a mudança inadequada de serviços web que levam problemas ao projeto.

o) Dívida de Usabilidade: diz respeito a questões inadequadas de usabilidade que terão que ser corrigidas futuramente.

2.3 IDENTIFICAÇÃO DE DT

Com o propósito de evitar a DT e seu acúmulo no projeto de software, ou gerenciar o melhor momento para paga-la, é preciso antes identifica-la. Para isto, é necessário entender quais são os tipos de dívida e seus indicadores (SEAMAN; GUO, 2011).

Através do mapeamento sistemático da literatura realizado por Alves *et al.* (2016), foi possível evidenciar a existência de diversos indicadores de DT, favorecendo a classificação conforme seus tipos, presente na Tabela 1.

Tabela 1 - Relação de indicadores por tipo de DT (Alves *et al.*, 2016)

Indicador	**Tipo de DT**
Violação de modularidade	Dívida de Arquitetura
Problemas na arquitetura software	
Betweenness Centrality	
Augmented Constraint Network	
Pairwise-Dependency Relation	
Index of Package Changing Impact	
Index of Package Goal Focus	
Depenências estruturais	Dívida de Arquitetura / Dívida de Projeto / Dívida de Construção
Problemas de construção (build issues)	Dívida de Construção
Código sem padrões	Dívida de Código
Algoritmo lento	
Multithread Correctness	
Code Metrics	Dívida de Código / Dívida de Projeto
Automatic Static Analysis (ASA) Issues	
Code Smells	
Grime	Dívida de Projeto
Problemas de projeto de software	
Low External / Internal Quality	
Afferent / Efferent Couplings (AC/EC)	
Depth of Inheritance Tree (DIT)	
Referential Integrity Constraints (RICs)	
Defeitos conhecidos não corrigidos	Dívida de Defeito / Dívida de Teste
Comentários insuficientes no código	Dívida de Documentação
Falta de documentação	
Comentários (hack, fixme, is problematic, …)	
Problemas na documentação	
Lista de pendências em requisitos	Dívida de Requisitos
Serviços web que não atendem as restrições de qualidade do projeto	Dívida de Serviço
Falta de teste automatizado	Dívida de Teste
Testes incompletos	
Correção de defeitos adiada	
Cobertura de código insuficiente	
Falta de documentação de casos de teste	
Falta de planejamento de casos de teste	
Uso desnecessários de forks	Dívida de Versionamento

Como indicado na Tabela 1, existem diversos indicadores de DT. Tratando-se da identificação do tipo dívida de código, é preciso que ele seja analisado para que a dívida seja detectada, sendo este processo realizado de forma manual ou automática.

Na forma manual, os responsáveis necessitam ter acesso aos arquivos do projeto de software, abrir e vasculhar em busca das incidências de DT. Quanto maior a quantidade de código para ser analisada, maior será o tempo necessário para a identificação. Já no processo automatizado, utiliza-se softwares específicos que efetuem os mesmos processos.

Nos últimos anos, algumas ferramentas foram desenvolvidas com o objetivo de identificar defeitos de forma automática em código fonte, utilizando técnicas sofisticadas de análises, como análise de fluxo de dados, provas de teoremas, verificação de modelos e correspondência de padrões sintáticos (DAVID *et. al*., 2004). Através de processo automatizado, estas ferramentas utilizam técnicas para encontrar problemas e falta de padrões presentes no código fonte do projeto, podendo evidenciar a presença da DT.

Em softwares de código aberto, as ferramentas mais conhecidas são: FindBugs, PMD e SonarQube. FindBugs é uma ferramenta de código aberto que utiliza análise estática automática (ASA) para encontrar possíveis bugs/defeitos em softwares escritos na linguagem de programação Java (FINDBUGS, 2017). Já PMD trata-se de uma ferramenta também *open source* que procura por falhas comuns de programação como criação de objetos desnecessários, variáveis não utilizadas, blocos catch vazios, e outros. O PMD suporta Java, JavaScript, PLSQL, XML, XLS. Conta também com um detector de código duplicado (CPD) nas linguagens Java, C, C++, Ruby, PHP, Python e outras (PMD, 2017). Por fim, o SonarQube refere-se a uma ferramenta *open source* de gestão de qualidade. Ele oferece suporte a mais de 20 linguagens, entre elas C, C#, C/C++, Java, JavaScript, Python, PHP, VB. Além disso, possui um conjunto de *plugins* que permite estender as funcionalidades originais da ferramenta, dando suporte à detecção de outros tipos de problemas de código NET (SONARCUBE, 2016).

2.4 GERENCIAMENTO DE DT

O surgimento de DT de forma intencional pode caracterizar uma tomada de decisão consciente, para atingir um objetivo específico, como por exemplo, obedecer a prazos de entrega. Segundo Kruchten *et al.* (2012), quando a equipe do projeto está ciente da existência

da DT e do seu custo, em alguns casos ela poderá ser considerada como bem-vinda. Entretanto, há um risco de qualidade associado que precisa ser acompanhado.

Para melhorar a produtividade do processo de desenvolvimento e a qualidade do software, é necessário o gerenciamento da DT através dos processos de identificação, medição e gerenciamento (Guo *et al.*, 2014). Estes processos apoiam os gestores a tomar decisão fundamentada em evidências, estabelecendo benefícios de qualidade ao software.

2.4.1 Estratégias de Gerenciamento de DT

De acordo com Alves *et al*. (2016), a DT pode ser medida através do esforço para se pagar a dívida. Portanto, torna-se relevante para o gerenciamento acompanhar o acúmulo do tempo necessário para corrigir problemas identificados de DT. Na Tabela 2 são apresentadas estratégias de gerenciamento de DT identificadas no mapeamento Sistemático de Alves *et al*. (2016) somadas a outras encontradas na literatura. Ao total, foram selecionadas seis abordagens que tiveram ao menos referências de três estudos na disponíveis na literatura e que serviram de base para este trabalho.

Tabela 2 – Trabalhos sobre gerenciamento de DT

Estratégia de Gerenciamento	**Referências**
Abordagem de portfólio	(GUO; SEAMAN, 2011) (SEAMAN *et al*., 2012) (STOCHEL, 2012) (SNIPES *et al*., 2012) (POWER, 2013) (ZAZWORKA *et al*., 2013) (KEN, 2013) (GRIFFITH *et al*., 2014) (ALZAGHOUL ; BAHSOON, 2014) (OJAMERUAYE ; BAHSOON, 2014)
Análise de custo benefício	(GUO *et al*., 2011) (SEAMAN *et al*., 2012) (STOCHEL, 2012) (HOLVITIE ;VILLE, 2013) (MONTEITH ; MCGREGOR, 2013) (GRIFFITH *et al*., 2014) (ALZAGHOUL ; BAHSOON, 2014) (OJAMERUAYE ; BAHSOON, 2014) (RAMASUBBU ; KEMERER, 2014) (GUO *et al*., 2014)
Método SQALE	(LETOUZEY, 2012) (LETOUZEY e ILKIEWICZ, 2012) (KRUCHTEN, *et al*., 2012) (MAYR, PLÖSCH ; KÖRNER, 2014) (MAMUN, BERGER; HANSSON, 2014)
Options	(ZAZWORKA *et al*., 2011) (SEAMAN *et al*., 2012) (ALZAGHOUL ; BAHSOON, 2013) (GRIFFITH *et al*., 2014) (ALZAGHOUL ; BAHSOON, 2014) (OJAMERUAYE ; BAHSOON, 2014)
Technical Debt Management Framework - TDMF	(HOLVITIE ; LEPPANEN, 2013) (SEAMAN *et al*., 2011) (HOLVITIE, 2014) (GUO; SPÍNOLA; SEAMAN, 2014)
Threshold based approach to technical debt	(OZKAYA *et al*., 2010) (EISENBERG, 2012) (GRIFFITH *et al*., 2014)

2.4.1.1 Abordagem de portfólio

De acordo com Seaman *et al*. (2012), esta abordagem baseia-se em princípios da gestão de portfólio de finanças para determinar itens de DT que precisam ser incorridos ou eliminados. Seu propósito é selecionar quais são os conjuntos de ativos que podem ter mais retorno ou reduzir o risco do investimento.

Nesta abordagem, é preciso reunir itens de DT que irão compor uma lista. As informações presentes em cada item de DT são: data, tipo, localização, responsável, descrição, estimativa do valor da dívida e juros para o seu pagamento, caso não seja resolvida imediatamente. Desta forma, é possível comparar os itens identificados para facilitar a tomada de decisão para seu pagamento.

2.4.1.2 Análise de Custo Benefício

Trata-se de uma abordagem para justificar o pagamento da dívida em relação a quantidade de juros esperado. Se a taxa de juros for alta demais, justifica-se o pagamento da dívida.

Segundo Seaman *et al.* (2012), a composição da taxa de juros baseia-se em dois fragmentos: a probabilidade da dívida acumular custo adicional e a quantidade de trabalho caso em ambas situações, a dívida não seja paga.

2.4.1.3 Método SQALE

Segundo Letouzey e Ilkiewicz (2012), o método SQALE (*Software Quality Assessment based on Lifecycle Expectations* – Avaliação da Qualidade de Software com base em Expectativa de Ciclo de Vida) fornece uma orientação para o gerenciamento de DT presentes no código fonte de projetos de software.

A utilização deste método deve partir de uma lista de requisitos não funcionais para indicar o que gera a DT. Essa informação servirá como base para que a equipe de desenvolvimento consiga estimar a DT presente no código fonte.

2.4.1.4 *Options*

De acordo com Power (2013), *options* são oportunidades de investimento escolhidas no intuito de promover a redução de DT em projetos de software. Seaman *et al.* (2012), apontam que a alternativa de investir em refatoração no projeto de software é uma opção que facilita a manutenção do software no futuro, mas não há lucros de forma imediata. Dessa

forma, o poder de decisão do pagamento da DT é entendido como opção de investimento que pode incorrer em custos em busca de benefícios no projeto de software.

2.4.1.5 *Technical Debt Management Framework* - TDMF

Segundo Guo *et al.* (2011), essa técnica consiste em organizar as informações e atividades para gerir a DT, favorecendo a tomada de decisão no projeto de software. Para isso é preciso passar pelos processos de identificação, medição e monitoramento.

No TDMF, toda DT identificada possui informações sobre o seu tipo, data que foi incorrida, localização e responsável. Como passo seguinte, é preciso registrar o valor do principal (quantidade de tempo necessário para corrigir o item de DT), juros e probabilidade de pagar os juros. Com essas informações, é possível tomar decisões sobre quais itens de DT precisam ser pagos e quando.

2.4.1.6 Threshold Based Approach to Technical Debt

De acordo com Lanza e Marinescu (2006), *thresholds* são utilizados em conjunto com métricas para ajudar a identificar problemas relacionados à falta de boas práticas de software. Neste sentido, nessa abordagem são determinados limites de valores para identificar itens de DT presentes no projeto de software. Os valores dos limites podem ser customizados conforme características de cada projeto.

2.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou informações sobre dívida técnica, seus tipos, identificação e gerenciamento. Embora o termo DT seja relativamente novo, segundo o MS apresentado por Alves *et al*. (2016), ao longo do tempo a comunidade científica tem se expandido, apresentando muitos trabalhos sobre esta temática e sendo bem discutido no meio acadêmico.

É sabido que adquirir dívida de forma intencional, de curto prazo, pode trazer benefícios ao projeto de software, como a entrega antecipada de funcionalidades. Porém, é preciso saber o momento ideal para pagá-la, e evitar o acúmulo de juros excessivos. Quando estes ocorrem, podem até mesmo levar o projeto de software ao fracasso (GUO ; SEAMAN, 2011).

O processo de gerenciamento efetivo da DT em projetos de software permite diagnosticar a qualidade do projeto de modo a reduzir os riscos de reincidência e de pagamento de autos juros. Neste sentido, foram apresentadas seis abordagens sobre gerenciamento de DT disponíveis na literatura que servirão como base para este trabalho.

O desafio seguinte é saber como lidar com o gerenciamento de equipes de desenvolvimento de software para que seja possível relacionar as informações obtidas com o gerenciamento da DT. O próximo capítulo abordará sobre esta temática.

Capítulo 3 – Identificação e Gerenciamento da Qualidade da Contribuição de Desenvolvedores em Projetos de Software

Este capítulo apresenta conceitos relacionados à contribuição de desenvolvedores em atividades de codificação e uma maneira de qualificá-la ao longo do processo de evolução de projetos de software.

3.1 INTRODUÇÃO

Ostrand *et al.*, (2010) destaca que estudos têm revelado que a qualidade do software está diretamente associada ao desempenho dos desenvolvedores. Durante o processo de concepção é preciso que os desenvolvedores estejam comprometidos para cumprirem prazos e efetuarem entregas com qualidade. Esta informação ressalta a importância de se manter as equipes de desenvolvedores capacitadas para se obter melhores resultados.

Durante as últimas décadas, as pesquisas que relacionam a qualidade de software e o desempenho dos profissionais têm sido bem reconhecidas pela comunidade de engenharia de software (CATALDO *et al.*, 2013). Este fato demonstra como os fatores humanos necessitam de atenção e cuidado. Tais fatores são incertos e complicados, tornando o processo de desenvolvimento de software complexo (ZHOU *et al.*, 2014).

A contribuição do desenvolvedor no projeto de software está associada a diversas atividades, em várias etapas no desenvolvimento do projeto, como na escrita da documentação, codificação, desenvolvimento de testes, apresentação de erros, entre outros (GOUSIOs *et al.*, 2008). Segundo Latoza *et al.* (2006), cada desenvolvedor possui suas próprias características e comportamentos. Seus hábitos de codificação fazem parte de suas peculiaridades que ao longo do tempo irão sofrer mudanças de acordo com a aquisição de novas experiências e conhecimento.

Qui *et al.* (2015) afirmam que quanto mais desenvolvedores com mais conhecimento e tempo de experiência contribuem com o projeto de software, menos bugs serão introduzidos e o produto produzido tende a ser mais confiável. Como os defeitos de software são inevitáveis,

é preciso gerencia-lo através de fontes de informações disponíveis, como códigos fontes e *commits* (KHATUN *et al.,* 2016).

Este capítulo discute o processo de identificação e monitoramento da contribuição de desenvolvedores, tendo como fonte de informações o histórico da contribuição presente no código fonte, durante a etapa de evolução de software de cada desenvolvedor. Além disto, apresenta uma estratégia para mensurar a qualidade de contribuição de desenvolvedores.

3.2 QUALIDADE DA CONTRIBUIÇÃO DOS DESENVOLVEDORES

Para se ter software de qualidade, é preciso que os desenvolvedores sejam comprometidos e tenham bom desempenho. Desenvolvedores que possuem alta qualidade conseguem codificar melhor, reduzindo a incidência de erros em um tempo menor (QIU *et al.,* 2015).

Manter equipes de desenvolvedores bem preparadas ainda é um grande desafio, visto que o processo de se obter informações válidas sobre o desempenho de cada um pode ser uma tarefa trabalhosa. Segundo Cataldo *et al.* (2013), não é fácil quantificar a qualidade de desenvolvedores e, mesmo com uma medição, ainda há a necessidade de vincular a qualidade dos desenvolvedores aos arquivos trabalhados de uma maneira efetiva.

Uma das formas de se medir a qualidade de software pode ser através da medição da introdução de erros (QIU *et al.*, 2015). Em termos gerais, quanto mais erros surgirem de um desenvolvedor durante o processo de codificação, menor será a sua contribuição em termos de qualidade.

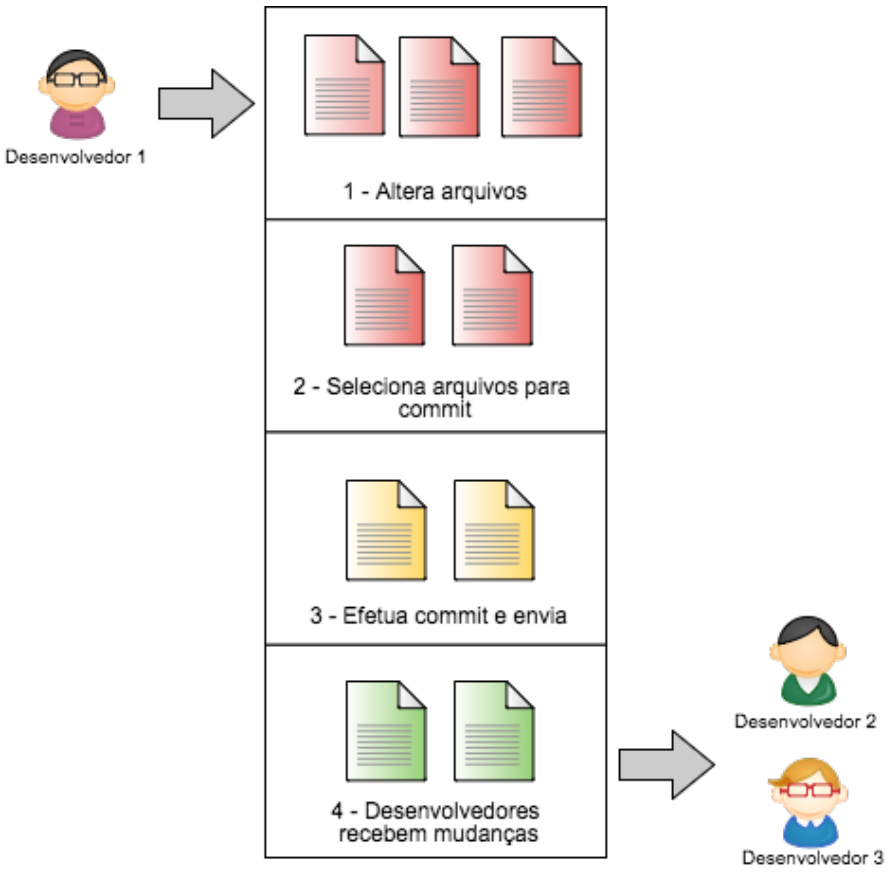
Uma análise criteriosa no código fonte do projeto de software pode revelar a introdução de defeitos, que são indicadores de má qualidade. Com estes dados é possível calcular uma taxa da introdução de erros e relacionar com os desenvolvedores. O código fonte é um dos principais artefatos de um projeto de software. Para que seja possível analisar a qualidade do desenvolvedor na etapa de codificação, é preciso investigar o histórico de atividades de cada envolvido no desenvolvimento do projeto.

Em projetos de software mantidos no sistema de versionamento GIT, todas as interações (*commits*) ficam armazenadas em uma base de dados específica, fornecendo acesso às informações através do seu log, possibilitando rastrear as alterações sempre que necessário. Por meio destes dados, é possível relacionar a contribuição de todos desenvolvedores ao longo do tempo através dos *commits* realizados.

Um *commit* representa um instante que um desenvolvedor realiza o envio de todas as mudanças realizadas durante o processo de desenvolvimento. Ele armazena informações sobre o autor (nome, e-mail, data e horário), mensagem de referência (descrição das atividades realizadas), além dos arquivos trabalhados (JUST *et al.*, 2016).

A Figura 4 representa um cenário colaborativo de uma equipe de desenvolvimento formada por três colaboradores utilizando o GIT. No primeiro passo, o desenvolvedor 1 evolui os trechos de código e salva os arquivos. Em seguida (passo 2), ele opta pelos arquivos que devem ser submetidos ao repositório. No passo seguinte ele efetua o *commit* informando as mudanças realizadas. A partir deste instante, os demais desenvolvedores estarão aptos a receberem as mudanças realizadas pelo primeiro desenvolvedor.

Figura 4 - GIT e equipes de desenvolvimentos



O sistema de versionamento possibilita manter as informações de mudanças realizadas durante a toda a evolução do software. De posse de um projeto de software mantido pelo GIT, qualquer usuário poderá investigar as mudanças realizadas e voltar para uma versão desejada.

Conforme Wu *et al.* (2014), é possível mensurar a qualidade do desenvolvedor através da taxa de introdução de erros, relacionando a quantidade de *commits* enviados com os que apresentaram incidência de problemas, conforme a equação a seguir:

$Qualidade(d_i) = 1 - \frac{CommitsBugs(d_i)}{Commits(d_i)}$, onde:

- $Qualidade(d_i)$ é o valor da qualidade do desenvolvedor.
- $CommitsBugs(d_i)$ representa o número de todos os *commits* que possuem a incidência de *bugs*.
- $Commits(d_i)$ é o número de todos os commits realizados pelo desenvolvedor.

Para fortalecer o entendimento na utilização da fórmula, pode-se utilizar o seguinte exemplo: se um determinado colaborador efetuar 10 *commits*, e forem encontradas incidências de *bugs* em 4 dos *commits*, a aplicação da fórmula será da seguinte maneira:

$$Qualidade(d_i) = 1 - \frac{CommitsBugs(d_i)}{Commits(d_i)}$$

$$Qualidade(d_i) = 1 - \frac{4}{10}$$

$$Qualidade(d_i) = 1 - 0{,}4$$

$$Qualidade(d_i) = 0{,}6$$

Para que seja possível efetuar uma leitura onde quanto maior for o resultado, será a qualidade, é realizada a subtração $1 - \frac{CommitsBugs(d_i)}{Commits(d_i)}$. Além disto, é possível compreender que, quanto maior a quantidade de commits com incidência de bugs, menor será o índice de qualidade.

3.3 MONITORAMENTO DE DÍVIDA TÉCNICA E EQUIPES DE DESENVOLVIMENTO

De acordo com França *et al.* (2008), muitos dos problemas que levam um projeto de software à falha são referentes à ineficácia no acompanhamento do time de desenvolvimento, causando erros, como de inconformidade de requisitos e outros. Além disso, através de uma pesquisa de campo, os autores também indicaram que a análise do perfil técnico, experiência e produtividade, personalidade e habilidades comportamentais estão positivamente relacionadas ao sucesso do projeto.

O trabalho de desenvolvimento de software necessita do engajamento da equipe de desenvolvimento e do líder da equipe (gerente de projeto) de modo que juntos, consigam superar os desafios que podem surgir. Em termos de código produzido, o gerente de projetos precisa estar atento para acompanhar a evolução e tentar reduzir o surgimento e acúmulo de problemas.

Em outro cenário, Lim *et al.* (2012) realizaram uma pesquisa com desenvolvedores de projetos de software para entender o contexto em que ocorre a DT, incluindo suas causas, sintomas e efeitos. Através de entrevistas realizadas em 26 empresas, totalizando 35 profissionais com experiências variadas na indústria, constataram que 75% dos participantes não estavam familiarizados com o termo DT mas tiveram facilidade em entender a sua metáfora. A pesquisa revelou que 25% das dívidas técnicas encontradas tratavam-se de dívida não intencional e o restante era resultado de decisões intencionais, fruto do curto prazo em reação às pressões do momento. Além disto, os autores identificaram que a maioria das equipes do projeto reconheceram que a dívida técnica é inevitável e necessária. Se a DT não pode ser evitada, é preciso gerenciá-la. Para isso, é fundamental reconhecer, rastrear, tomar decisões para evitar consequências futuras piores. Não é possível lidar com a DT sem antes identificá-la e comunicar com aqueles que podem tomar algum tipo de decisão. O processo de monitoramento deve ser constante durante toda a etapa de evolução do software.

O surgimento da DT evidencia a introdução de problemas, muitas vezes relacionados com a qualidade dos desenvolvedores. O monitoramento da DT possibilita o acompanhamento da equipe de modo a identificar a qualidade da contribuição de cada integrante.

O processo de monitoramento da evolução da DT exige uma grande quantidade de esforço e tempo para buscar no código fonte indícios de sua existência. Com a identificação da DT, é possível gerenciá-la de modo a reduzir os impactos negativos do seu acúmulo. Além disso, é possível entender a sua origem, quando, como e quem foi o responsável pelo seu surgimento. Essas informações podem subsidiar tomadas de decisão em um processo de capacitação da equipe de desenvolvimento para que seja possível reduzir problemas futuros.

3.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou relatos disponíveis na literatura que evidenciam a relação entre desenvolvedores e qualidade de software. Embora o processo de desenvolvimento de software seja complexo (ZHOU *et al.*, 2014), quanto mais conhecimento e experiência

desenvolvedores possuem, há mais chances do software ser desenvolvido com mais qualidade.

Ao longo do capítulo foi apresentada uma abordagem sobre o processo de identificação e monitoramento da contribuição de desenvolvedores, tendo como fonte de informações o histórico da contribuição presente no código fonte além de uma estratégia para mensurar a qualidade de contribuição de desenvolvedores.

No próximo capítulo será apresentada uma estratégia proposta para possibilitar o monitoramento da contribuição de desenvolvedores na evolução da DT. Será apresentado um estudo com o objetivo de vincular aos desenvolvedores atributos de qualidade que são relacionados à presença da DT.

Capítulo 4 – Uma Estratégia para o Monitoramento da Contribuição de Desenvolvedores na Evolução da Dívida Técnica

Neste capítulo é apresentada a estratégia proposta para apoiar o monitoramento da contribuição de desenvolvedores na evolução da DT. O capítulo também apresenta a ferramenta desenvolvida de apoio à estratégia.

4.1 INTRODUÇÃO

Segundo Cataldo *et al.* (2013), não é fácil detectar e corrigir módulos defeituosos em sistemas de software antes das liberações de versões. Além do mais, há um custo elevado para este processo. No âmbito do código fonte, evidenciar a existência da DT em grande quantidade de arquivos e linhas de código não é uma tarefa fácil e rápida, pois é preciso efetuar uma varredura em todos os arquivos, linha a linha. Em termos de tempo, é um processo que tende a ser custoso.

Estudos apresentados por Nugroho *et al.* (2011) evidenciaram a importância de se ter um método eficiente para detectar a presença da DT de forma que seja possível descrever seu impacto no projeto de software. Além disso, Lim *et al.* (2012) indicam que indicadores de DT presentes em código como *code smells*, complexidades e outras medidas não são fáceis de serem encontradas, o que motiva o uso de análise estática e dinâmica para a descoberta de itens de dívida.

Neste sentido, o apoio de uma estratégia para facilitar a detecção das incidências da DT pode ajudar no processo de identificação e viabilizar o monitoramento e contribuição de desenvolvedores na evolução da DT. Este capítulo apresenta uma estratégia para o processo de monitoramento da contribuição de desenvolvedores na evolução da DT, tendo como fonte de informações o histórico da contribuição de cada desenvolvedor, presente no código fonte,

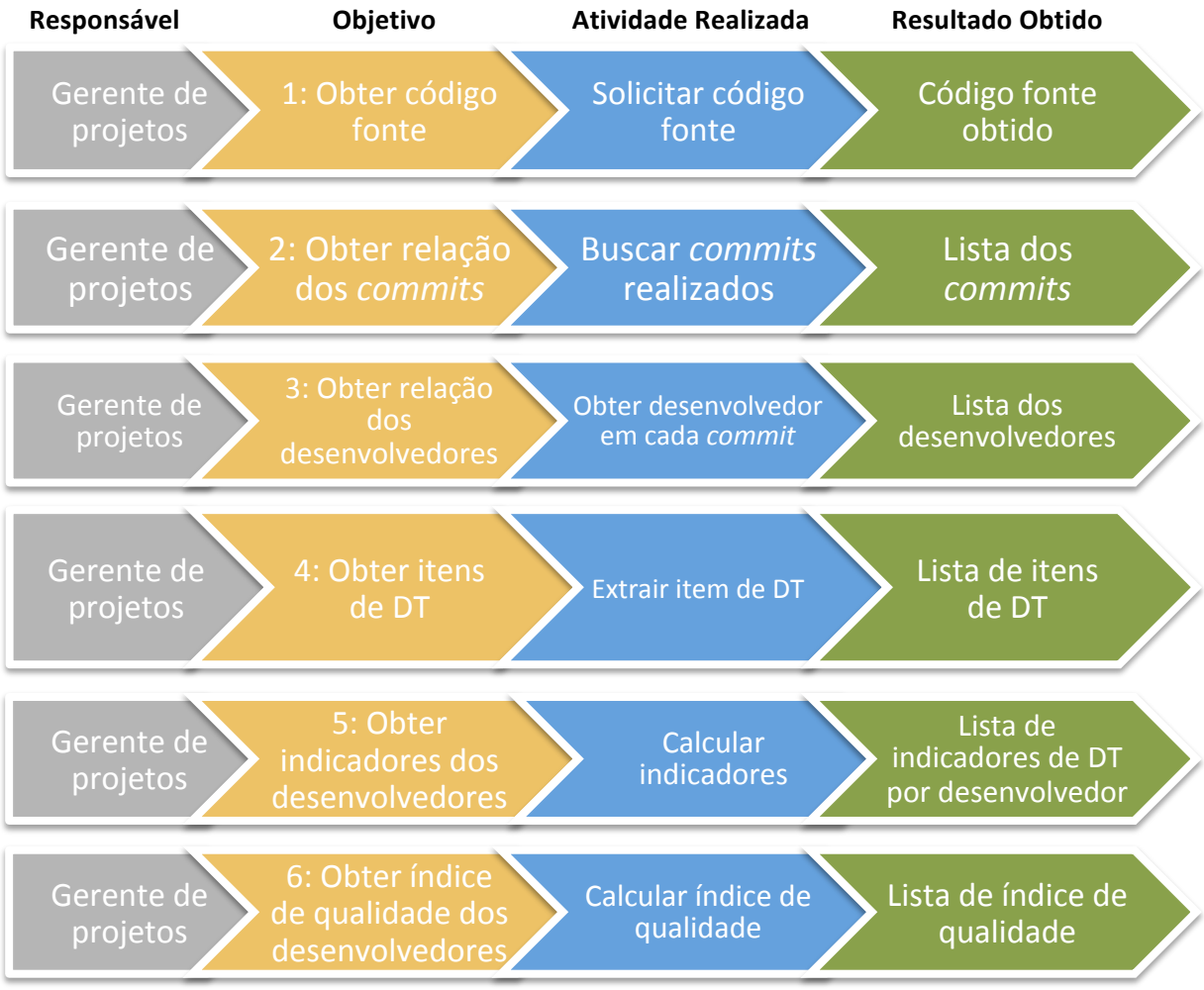
durante o processo de evolução do software. Além disto, este capítulo também apresenta uma ferramenta para apoiar a utilização da estratégia proposta.

4.2 ESTRATÉGIA PARA O MONITORAMENTO DA CONTRIBUIÇÃO DE DESENVOLVEDORES NA EVOLUÇÃO DA DÍVIDA TÉCNICA

As incidências de DT, chamadas de itens de DT, podem estar presentes em código fonte que precisam ser modificados, em casos de testes que precisam ser praticados, documentações que precisam ser atualizadas, entre outros. Como o escopo deste trabalho está direcionado para código fonte, será necessário obter os itens de DT inseridos nesse artefato.

Para apoiar o monitoramento da evolução da DT sob a perspectiva da contribuição dos desenvolvedores na incidência de itens de dívida, a estratégia proposta é formada pelas atividades representadas na Figura 5.

Figura 5 - Processos da estratégia



A estratégia parte do princípio de que há um software ou parte dele desenvolvido. Logo, há a presença de código fonte. É indicado que um profissional que tenha conhecimento

sobre o projeto de software, conhecimentos técnicos e não faça parte da equipe de desenvolvimento, seja o responsável em executar os processos. Esta orientação busca aumentar a imparcialidade sobre a análise dos desenvolvedores a partir do código fonte. O desejável é que um gerente de projetos assuma esta responsabilidade, uma vez que este normalmente já conhece e atua com os membros da equipe de desenvolvimento. Desta forma, ele precisa obter acesso ao código fonte do projeto, conforme recomendação do passo 1.

O passo 2 orienta que o gerente de projetos obtenha a relação dos *commits* presentes durante a evolução do software. Os dados necessários estão descritos no *log* do GIT. Os *commits* possuem diversas informações sobre a evolução do código fonte, inclusive sobre os responsáveis. Neste momento, os dados que serão extraídos são:

a) Commit hash – código único que representa um objeto e referência de um commit;

b) Commiter name – nome do desenvolvedor que submeteu o código fonte;

c) Commiter email – endereço de e-mail do desenvolvedor que submeteu o código fonte;

d) Committer date – data que o desenvolvedor submeteu o código fonte;

e) Arquivos – nomes dos arquivos e linhas que foram evoluídos.

Com a lista de *commits* do projeto que está sendo analisado, é possível prosseguir para o passo 3, obtendo a lista de todos os desenvolvedores que atuaram no projeto de software, independente da participação na incidência de DT. É relevante que todos os desenvolvedores sejam considerados para que, posteriormente, sejam atribuídos os indicadores de DT, mesmo que existam casos de desenvolvedores com nenhuma participação na incidência de DT.

O passo 4 instrui a identificação e extração de itens de DT presentes no código. Para encontrar esses itens presentes no código fonte, é preciso analisar todo o seu conteúdo. Uma vez identificado, cada item de DT deve então ser descrito considerando (ver Tabela 3) sua localização, o momento em que ela foi identificada, a pessoa responsável, a razão por que é considerada dívida técnica, uma estimativa do valor da dívida e a estimativa do valor de juros (Seaman *and* GUO, 2011) e também a data de identificação e o tipo de indicador (ex.: código duplicado, código complexo, método longo, entre outros).

Tabela 3 - Modelo de classificação do item de DT, segundo Zazworka *et al.* (2013)

ID	Um código único identificador.
Responsável	A pessoa responsável pelo item de DT.
Tipo	Design, código, documentação, teste ou outro tipo.
Localização	Lista de arquivos/classes/métodos ou documentos/páginas envolvidas.
Descrição	Informação sobre a anomalia os possíveis impactos em uma manutenção futura.
Estimativa do valor da dívida	Quanto de trabalho é necessário para pagar o item de DT.
Estimativa de juros	Quanto de trabalho extra será necessário para executar se o item de DT não for pago logo.
Estimativa de juros (probabilidade)	Qual a probabilidade de ter trabalho extra caso este item de DT não for pago logo.
Intencional?	Se o item de DT foi adicionado intencionalmente ou não.

Com a relação de itens de DT, é possível passar para o passo 5, relacionando por desenvolvedor, a quantidade total da estimativa do valor da dívida, tipo de indicador e intencionalidade. De posse dessas informações, é possível apontar indicadores importantes como quais são os desenvolvedores que:

a) Mais acumularam incidências de itens de DT;
b) Mais acumularam horas estimadas para pagar os itens de DT;
c) Mais acumularam itens de DT de forma não intencional (que pode apontar falta de conhecimento ou negligência);
d) Mais acumularam itens de DT por tipos de indicadores.

E, por fim, com o passo 6 é possível obter de cada desenvolvedor, o índice de qualidade dos *commits*, através da equação: $Qualidade(d_i) = 1 - \frac{CommitsBugs(d_i)}{Commits(d_i)}$, apresentada na seção 3.2, em uma escala de 0 a 1, onde 0 é a pior qualidade possível e 1 a

melhor. Exemplo: supondo que um desenvolvedor efetuou 10 *commits* e em 4 deles foram encontrados itens de DT, o resultado seria:

- $Qualidade(d_i) = 1 - \frac{CommitsBugs(d_i)}{Commits(d_i)}$
- $Qualidade(d_i) = 1 - \frac{4}{10}$
- $Qualidade(d_i) = 1 - 0.4$
- $Qualidade(d_i) = 0.6$

O índice de qualidade neste exemplo é de 0.6. Se multiplicarmos a razão $\frac{CommitsBugs(d_i)}{Commits(d_i)}$ por 100, teremos o percentual de *commits* que apresentaram incidência de DT (40%) e se multiplicarmos a razão da qualidade por 100, teremos o percentual de *commits* que não apresentaram incidência de DT. Com o uso deste passo espera-se obter a lista contendo o índice de qualidade de todos os desenvolvedores, podendo favorecer a tomada de decisão para qualificar a equipe de desenvolvimento.

4.3 VISMINERTD

Com o objetivo de facilitar a utilização da estratégia de monitoramento da DT proposta, foi elaborado um software chamado VisminerTD. Ele funciona em conjunto com o software Visminer, desenvolvido em parceria com outros pesquisadores dos grupos de pesquisa SoftVis (Grupo de Visualização de Software – UFBA), TDResearchTeam (TechnicalDebtResearch Team – UNIFACS/UFBA/IFBA) e o GIA (Grupo de Informática Aplicada – IFBA).

Foram necessárias participações em reuniões presenciais e remotas, com o grupo de desenvolvedores do Visminer, para que, juntos, houvesse o entendimento de que a ferramenta necessitaria evoluir para atender necessidades atuais. As seguintes sugestões foram acolhidas pela equipe:

a) Utilizar uma nova identidade visual – Foi aceito um *template* gratuito que utiliza padrões gráficos modernos;

b) Utilizar layout responsivo – Tecnologia que permite a adaptação do layout em dispositivos com telas com tamanhos distintos;

c) AngularJS – *Framework* JavaScript de código aberto, mantido pelo Google, para tornar a interface mais dinâmica e interativa;

d) Single-Page Applications (SPA) – Arquitetura que possibilita carregar conteúdo de páginas web por demanda (sem recarregar todas as páginas);

e) Possibilitar adicionar perspectivas – Agregar funcionalidades por área, como aconteceu com o VisminerTD, no formato de perspectiva para dívida técnica, para que o Visminer comporte outras áreas de visualização.

A partir destas definições, junto com a equipe do Visminer, foi planejada e executada a nova versão, obedecendo os critérios mencionados.

Durante o processo de desenvolvimento da ferramenta foram utilizadas as seguintes tecnologias e ferramentas:

a) Eclipse Java EE – IDE gráfica utilizada para desenvolvimento do *backend* do Visminer, para fornecer os dados necessários para execução do VisminerTD;

b) Sublime Text – IDE gráfica utilizada para desenvolvimento do *frontend* (páginas html, js – contendo regras da aplicação).

c) Studio 3T – IDE gráfica para gerenciamento dos dados presentes no MongoDB;

d) Servidor Apache Tomcat – Software que permite a execução de aplicações java, necessário para rodar os Servlets na camada do Visminer.

O VisminerTD surgiu como uma perspectiva destinada à análise de DT em projetos de software. Parte do trabalho desenvolvido incluiu o planejamento e execução da nova versão do Visminer que utiliza novas tecnologias. O processo de utilização da ferramenta é dividido em dois cenários: um de mineração e outro de visualização.

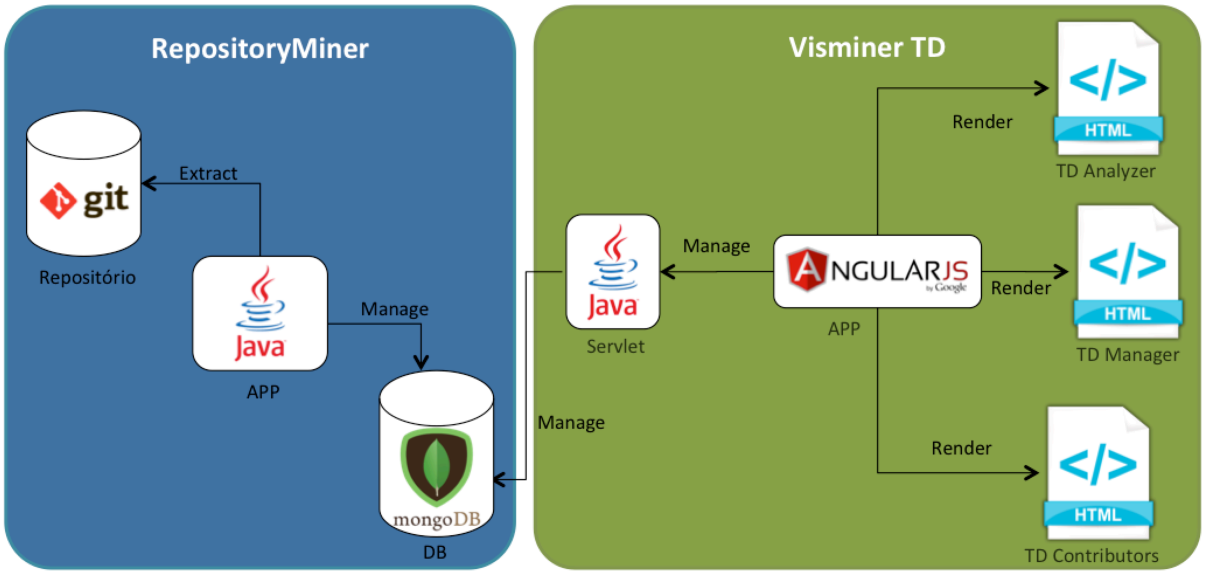
O primeiro passo é extrair métricas presentes no código fonte através do projeto vinculado ao repositório GIT, para tornar possível o armazenamento das métricas em um sistema de banco de dados. Estes processos apoiam as atividades 2, 3 e 4 definidas na estratégia, apresentadas na seção 4.2. Em um segundo momento é possível recuperar as informações do banco de dados e realizar análises da DT através de um navegador web, que contemplam as atividades 5 e 6 definidas na estratégia, também presentes na seção 4.2.

A Figura 6 apresenta uma ilustração da comunicação entre o RepositoryMiner e o VisminerTD. Eles compartilham as informações presentes na base de dados MongoDB. O

VisminerTD utiliza um Servlet em Java para buscar as informações necessárias para gerar e apresentar as interfaces, além de persistir informações exclusivas do próprio VisminerTD.

O VisminerTD contempla três componentes: TD Analyzer, TD Manager e TD Contributors. Cada componente possui responsabilidades distintas. O TD Analyzer é encarregado de obter dados preliminares sobre os itens de DT que estão presentes no código fonte, que contempla parte da Tabela 4. O TD Manager apoia o gerenciamento dos itens de DT complementando as informações da Tabela 5. Por fim, no TD Contributors, é possível obter os indicadores que indicam as atuações dos desenvolvedores.

Figura 6 - *VisminerTD* - Arquitetura



4.3.1 Extração de indicadores de Dívida Técnica

De acordo Sillitti *et al.* (2003), o processo manual de coleta de métricas é uma prática não confiável considerando que erros demasiados ou falta de dados afetam drasticamente o processo da análise.

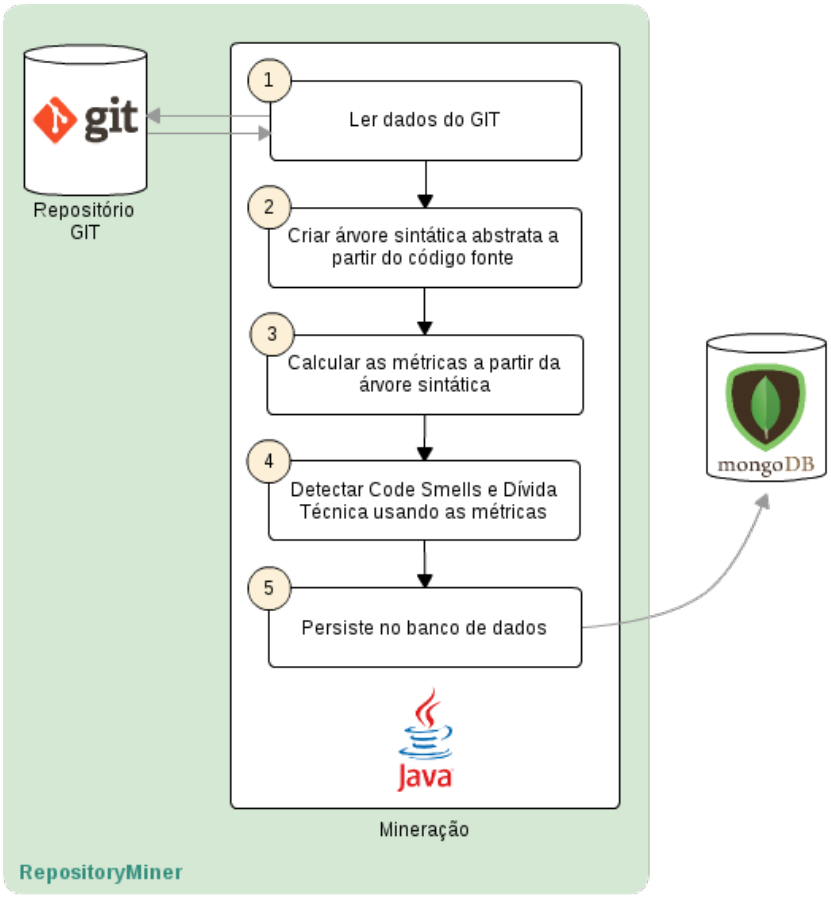
Neste aspecto, a utilização de um processo automatizado e confiável poderá reverter este cenário. Na comunidade, há várias ferramentas que têm a capacidade de extrair informações sobre incidência de defeitos no código fonte do projeto de software, fruto de más práticas de codificação. O RepositoryMiner é uma ferramenta *open source* que traz uma proposta de minerar o código fonte de projetos de software mantidos pelo sistema de

gerenciamento de código fonte (SCM) GIT (sistema de controle de versão distribuído), em busca de defeitos.

Os tipos de indicadores do tipo *code smells* disponíveis através da mineração com o apoio do RepositoryMiner são: *Brain Class, Brain Method, Conditional Complexity, Data Class, Feature Envy, God Class, Long Method, Refused Parent Bequest* e *Depth of Inheritance Tree.*

Conforme representação dos processos do RepositoryMiner presente na Figura 7, o primeiro passo é obter todas estruturas do repositório GIT que mantém informações históricas do código fonte como as *branchs* (ramificações), tags, responsáveis, datas, arquivos e modificações efetuadas.

Figura 7 - *VisminerTD* - Processos do RepositoryMiner

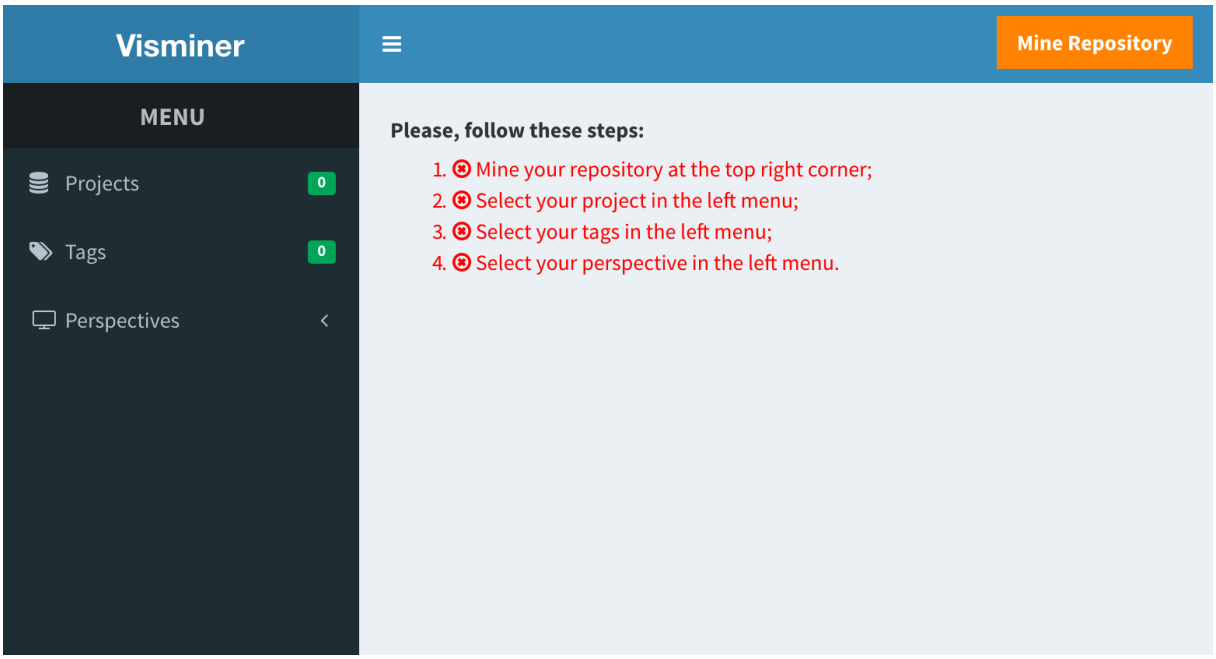


Os passos seguintes são: criar árvore sintática com todas as ramificações, calcular e detectar os *code smells* e DT usando as métricas pré-estabelecidas e salvar as informações na base de dados MongoDB. Todo o processo de mineração pode ser efetuado utilizando

comandos Java do RepositoryMiner, mas com o objetivo de facilitar o uso, o VisminerTD possui uma interface gráfica que apresenta um guia rápido e prático que efetua este processo.

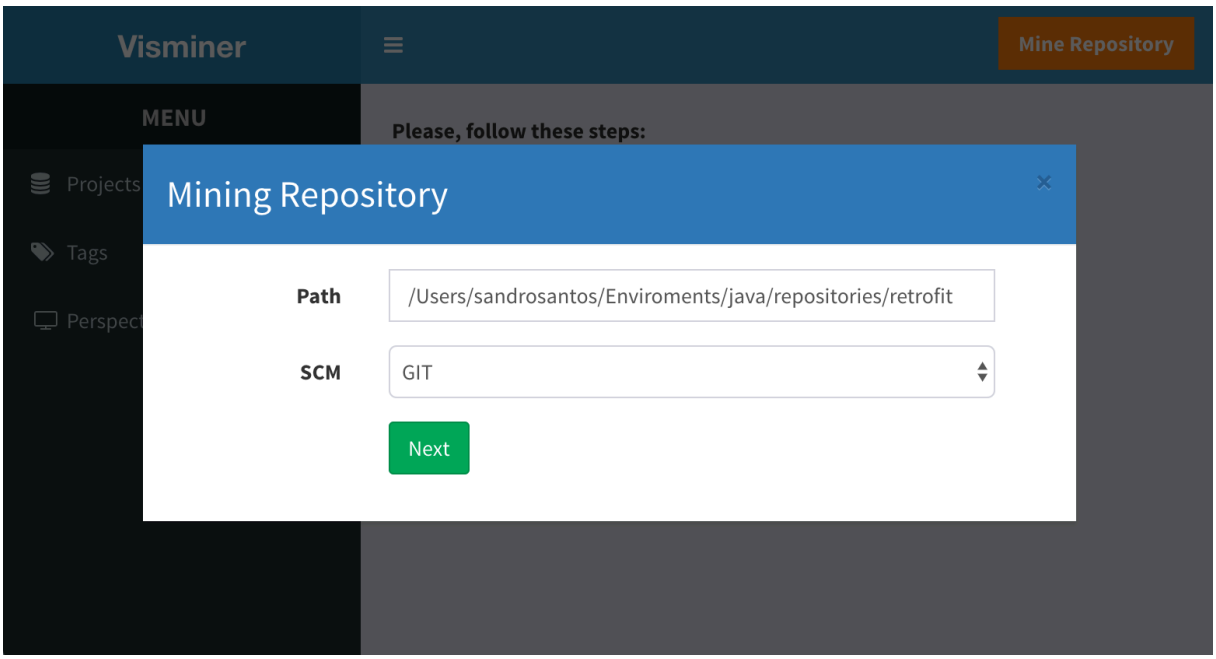
Para apoiar a realização das atividades 2 e 3 da estratégia, descritas na seção 4.2, será preciso efetuar a mineração do repositório de um projeto. O primeiro passo é acessar a ferramenta através de um navegador (ex.: http://localhost/visminer). Então será apresentada a interface conforme a Figura 8. Ela contém um menu à esquerda com três opções (discutidos posteriormente), orientações na área central, e um botão na área superior chamado Mine Repository. O passo seguinte é clicar neste botão para iniciar o guia que possibilita minerar informações de um repositório.

Figura 8 - *VisminerTD* - Tela inicial



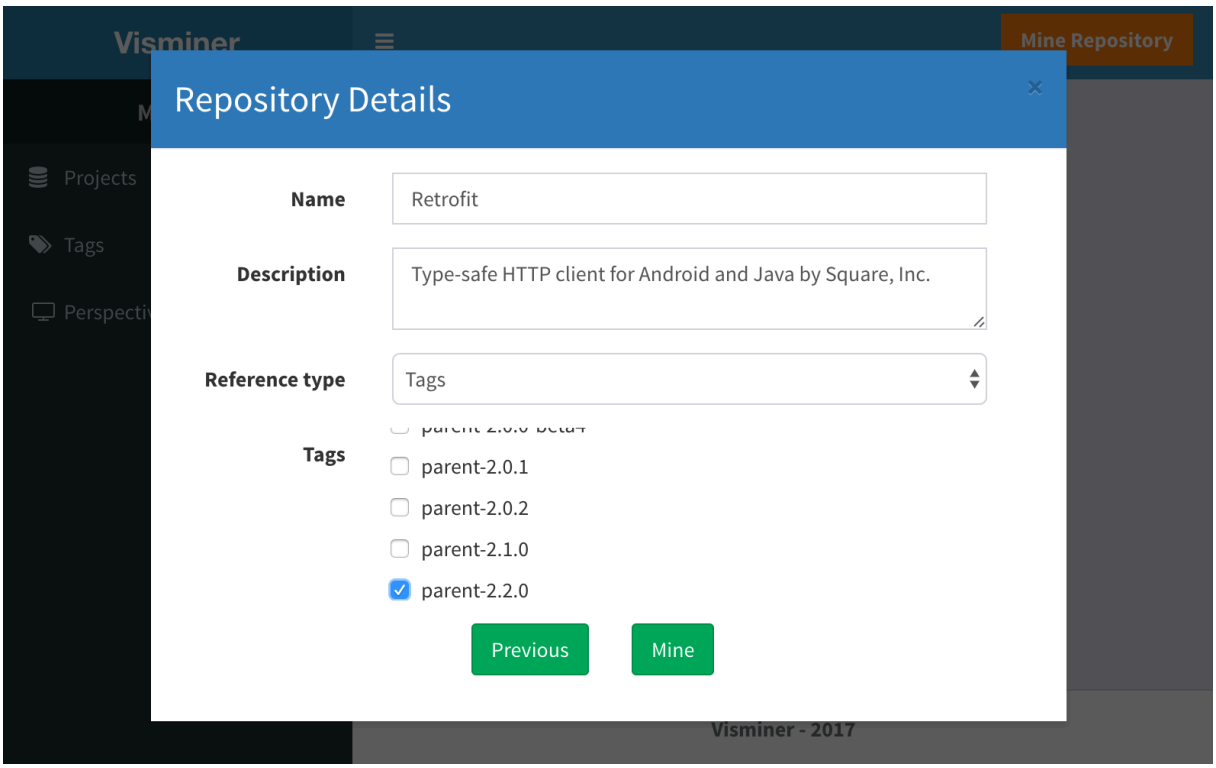
De acordo com a Figura 9, o Visminer apresenta uma tela solicitando o caminho do repositório (onde o projeto está salvo no computador) e o sistema de SCM (Software Configuration Manager), que é o tipo do sistema de controle de código. Escolheremos a opção GIT. Para prosseguir deve-se clicar no botão Next (próximo).

Figura 9 - *VisminerTD* - Escolhas do caminho e SCM



Conforme apresentado na Figura 10, o passo seguinte consiste em informar o nome do projeto, descrição e escolher o tipo de referência (se iremos analisar *branchs* ou *tags*). Escolhido o tipo de referência, o sistema apresentará as *branchs* ou *tags* encontradas no projeto para que o usuário indique quais ele quer minerar. Para finalizar o processo, o usuário deve optar pelo botão Mine (minerar).

Figura 10 - *VisminerTD* - Informações adicionais



Uma vez tendo projetos já minerados, a ferramenta apresentará a quantidade de projetos minerados e a lista para que o usuário possa escolher qual ele gostaria de visualizar, conforme apresentado na Figura 11. Além disso, como a ferramenta possibilita a mineração de mais de uma *tag*, o usuário poderá escolher uma ou mais para trabalhar, conforme apresentado na Figura 12.

Figura 11 - *VisminerTD* - Tela inicial com projetos minerados

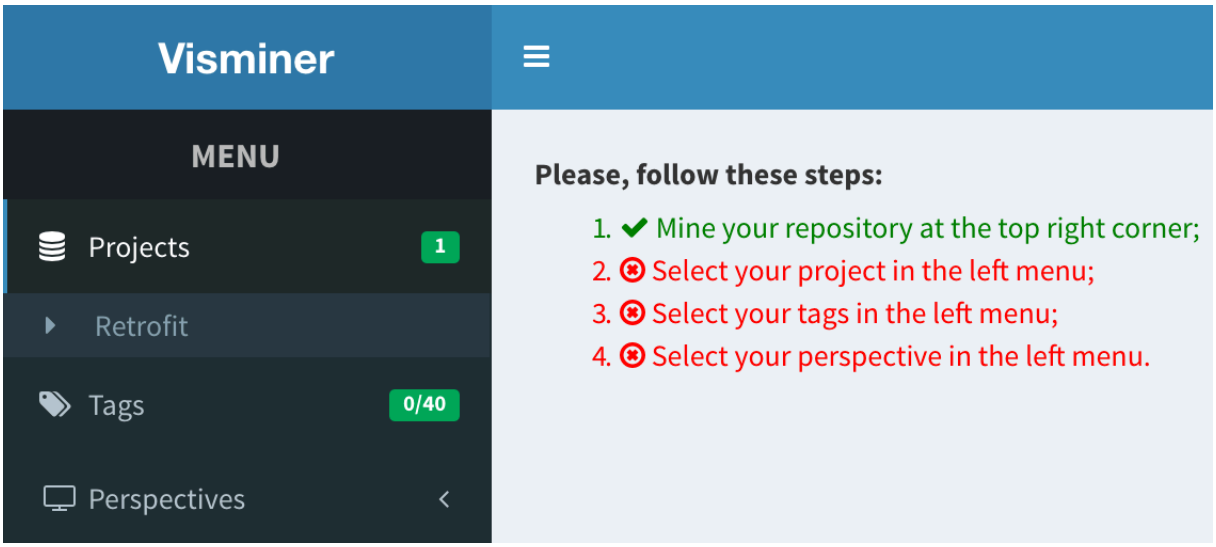
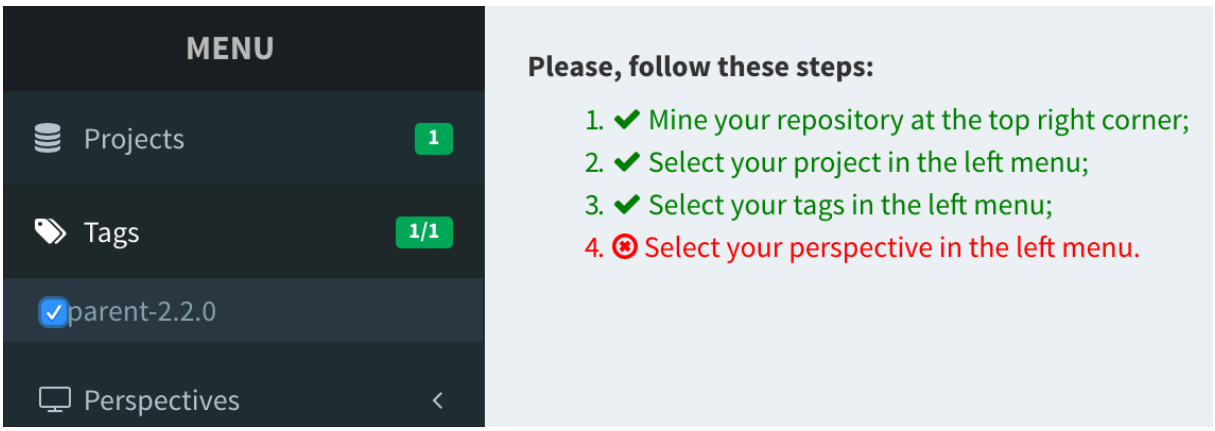


Figura 12 - *VisminerTD* - Escolha da referência



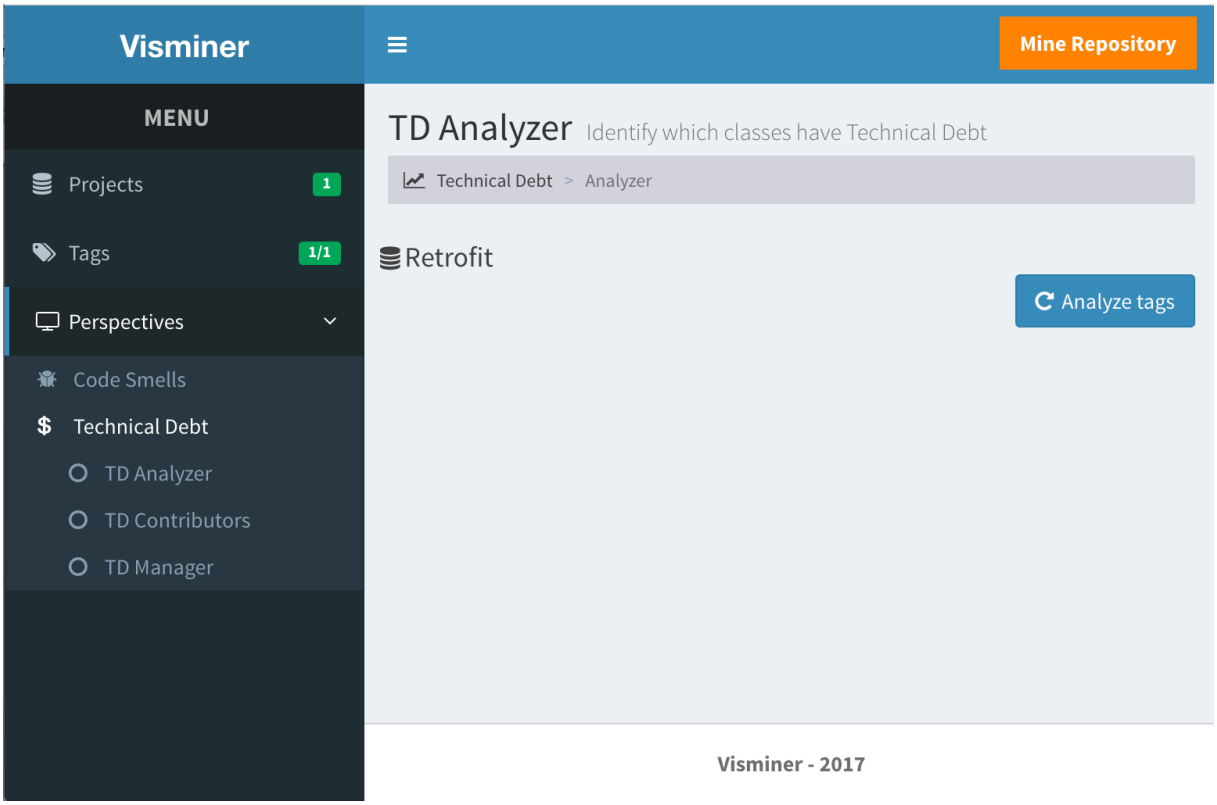
4.3.2 Componente TD Analyzer

Para apoiar a realização parcial da atividade 4 da estratégia, descrita na seção 4.2 e presente neste componente, o usuário deverá buscar informações sobre DT disponíveis no projeto, semelhante ao passo 5 da Figura 5 que explica a aquisição dos itens de DT presentes no código fonte do projeto. Esta opção contemplará em parte a realização da atividade 4 da estratégia, que diz respeito a obtenção das informações dos itens de DT, pois neste momento ainda não é possível gerenciar todas as informações dos itens de TD. As informações complementares estão disponíveis no componente TD Manager, explicado na próxima seção.

Internamente este processo consiste em abrir cada arquivo presente no projeto de software com o objetivo de identificar os trechos de código que apresentam a incidência de DT. Para cada item de DT encontrado em um arquivo, este componente deverá manter a data que a DT surgiu (com base na modificação do arquivo), o tipo de DT, os indicadores de DT e desenvolvedores responsáveis pelo surgimento.

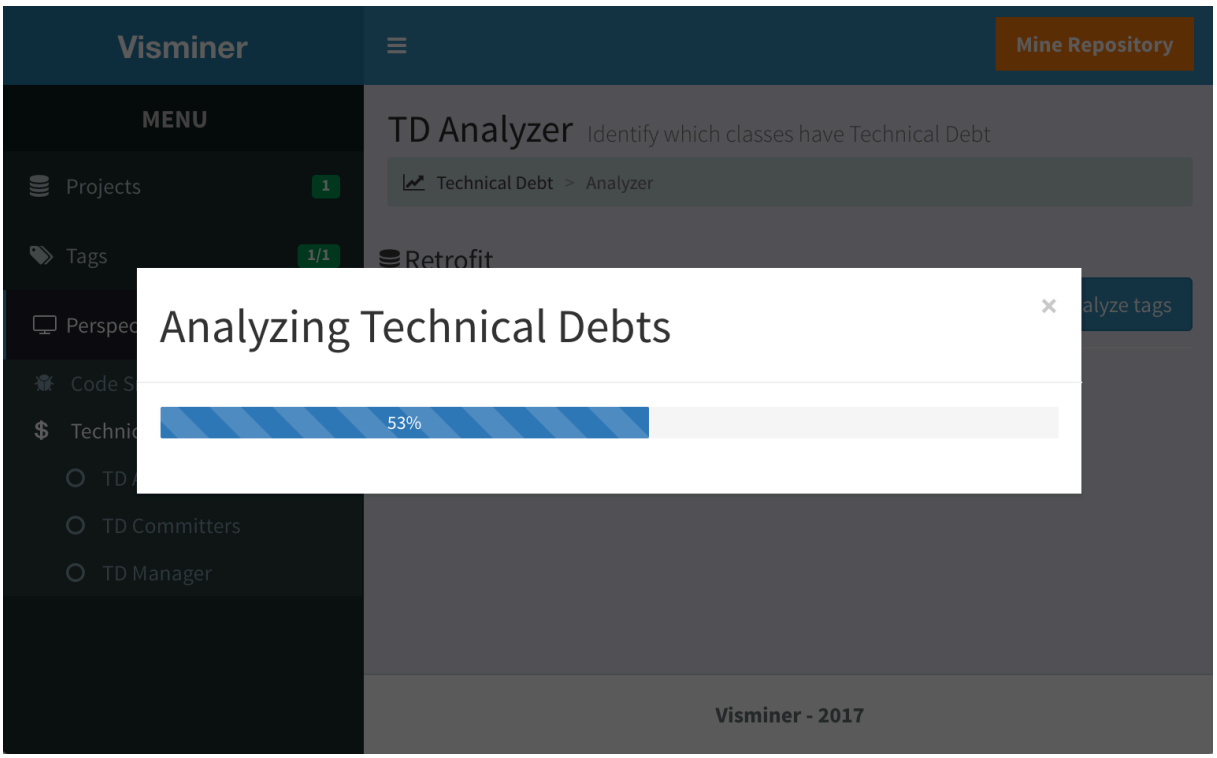
Para utilizar este componente, o usuário deve acessar a perspectiva do VisminerTD, optando pelas opções Technical Debt e em seguida TD Analyzer, ambas disponíveis no menu esquerdo da ferramenta conforme a Figura 13.

Figura 13 - *VisminerTD* - Análise das informações da DT



O VisminerTD efetuará uma requisição ao RepositoryMiner, solicitando que ele extraia todos os indicadores de itens de DT presentes na seleção escolhida, de acordo com a Figura 14. Após o término do processo, os dados dos itens de DT estarão prontos e o usuário poderá gerenciar as informações na interface TD Manager, descrita na próxima seção.

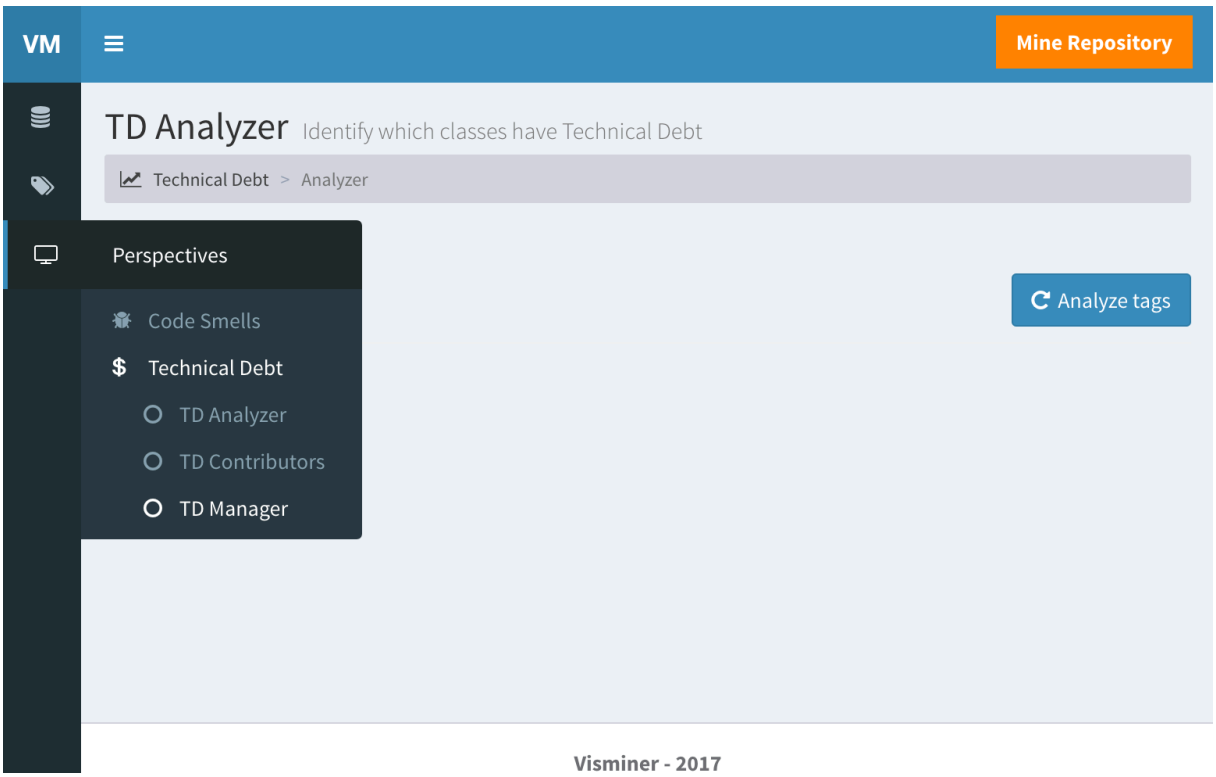
Figura 14 - *VisminerTD* - Andamento da análise da DT



4.3.3 TD Manager

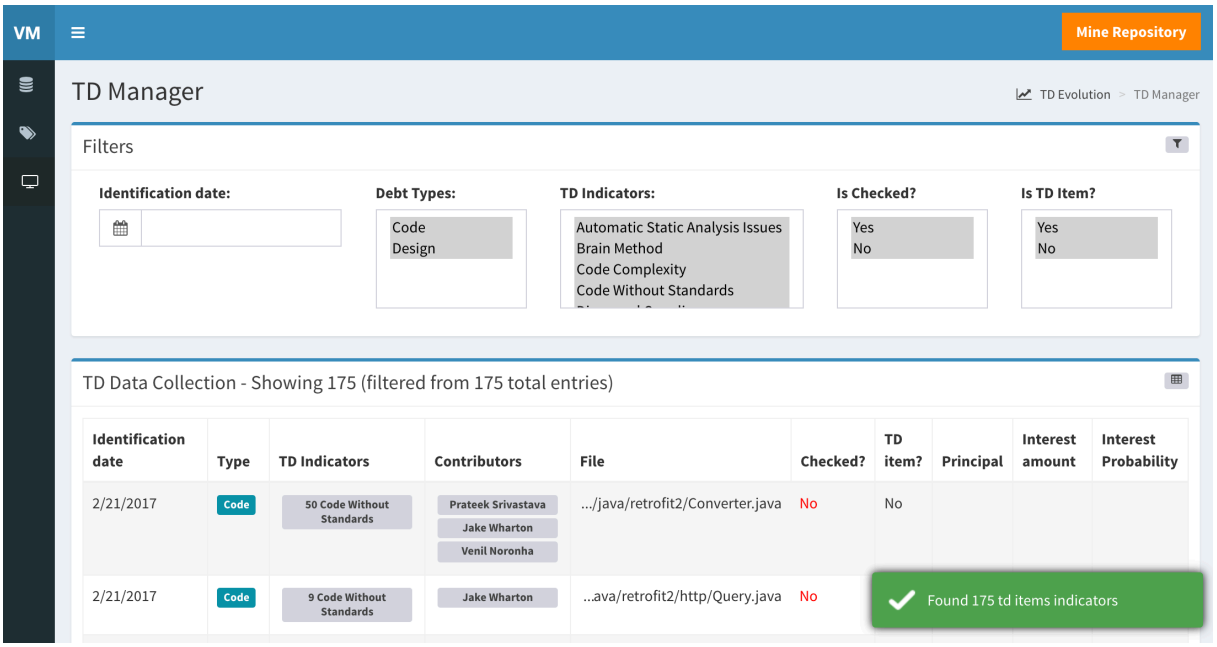
Para apoiar a realização da atividade 4 definida na estratégia, esse componente oferece a opção para gerenciar informações sobre os itens de DT. Para isto, o usuário deverá escolher a opção “TD Manager” presente no menu esquerdo da ferramenta, conforme Figura 15. Esta opção levará para a interface que possibilitará o processo de gerenciamento de itens da DT.

Figura 15 - *VisminerTD* - Acesso ao TD Manager



A ferramenta então apresentará o TD Manager, que se trata de uma interface contendo uma tabela com todos os itens de DT encontrados no processo de análise da DT, por arquivo, conforme é apresentado no Figura 16.

Figura 16 - *VisminerTD* - Visualizar itens de DT no TD Manager



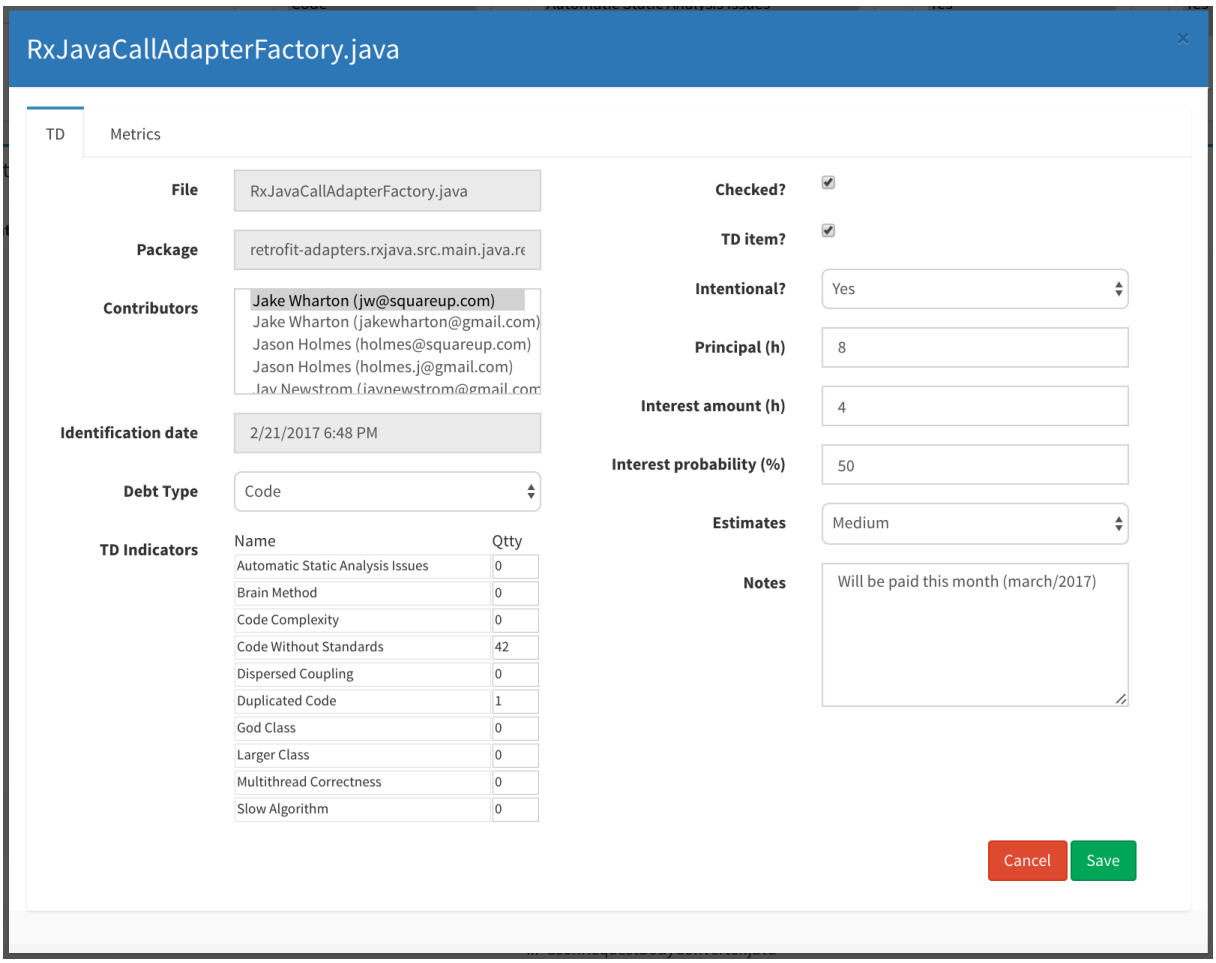
A tabela contém:

a) “Identification date”: data de quando o item de DT foi adicionado ao projeto;
b) “Type”: o tipo de DT que neste caso é code, por tratar-se de DT presente no código fonte do projeto;
c) “TD Indicators”: quais são os indicadores de DT (exemplo: 4 códigos duplicados);
d) “Contributors”: quais são os desenvolvedores que contribuíram com a existência do item de DT;
e) “File”: o arquivo onde foi encontrado a presença da DT;
f) “Checked?”: indicador se o item de DT foi analisado/verificado;
g) “TD Item?”: indicador se o item de DT foi confirmado;
h) “Principal”: a quantidade de tempo (em horas) necessária para correção;
i) “Interest amount”: a quantidade de tempo (em horas) necessária para correção no código em produção (ou seja, no futuro);
j) “Interest probability”: indicador (percentual) das chances de pagar juros no futuro.

No intuito de facilitar o processo de gerenciamento dos itens de DT, o TD Manager possui um filtro permitindo restringir uma faixa de tempo, o tipo de dívida, indicadores de DT, se os itens já foram verificados e/ou se eles já foram confirmados como DT. A representação do filtro está presente na parte superior da Figura 16.

Para gerenciar as informações presentes em um item de DT, basta clicar na linha da tabela onde se encontra o item desejado. A aplicação exibirá uma tela (conforme Figura 17) contendo duas abas: TD e Metrics.

Figura 17 - *VisminerTD* - Gerenciar itens de DT



Na aba TD estão:

a) “File”: o arquivo onde foi encontrado a presença da DT;
b) “Package”: o pacote/caminho para se chegar até o arquivo;
c) “Contributors”: a lista de todos os desenvolvedores que contribuíram com o projeto;
d) “Identification date”: data de quando o item de DT foi adicionado ao projeto;
e) “Debt Type”: o tipo de DT;
f) “TD Indicators”: relação de todos os indicadores e valores de DT que o VisminerTD dá suporte;
g) “Checked?”: indicador se o item de DT foi analisado/verificado;
h) “TD Item?”: indicador se o item de DT foi confirmado;
i) “Intentional?”: indicador se o item de DT foi intencional;
j) “Principal”: a quantidade de tempo (em horas) necessária para correção;

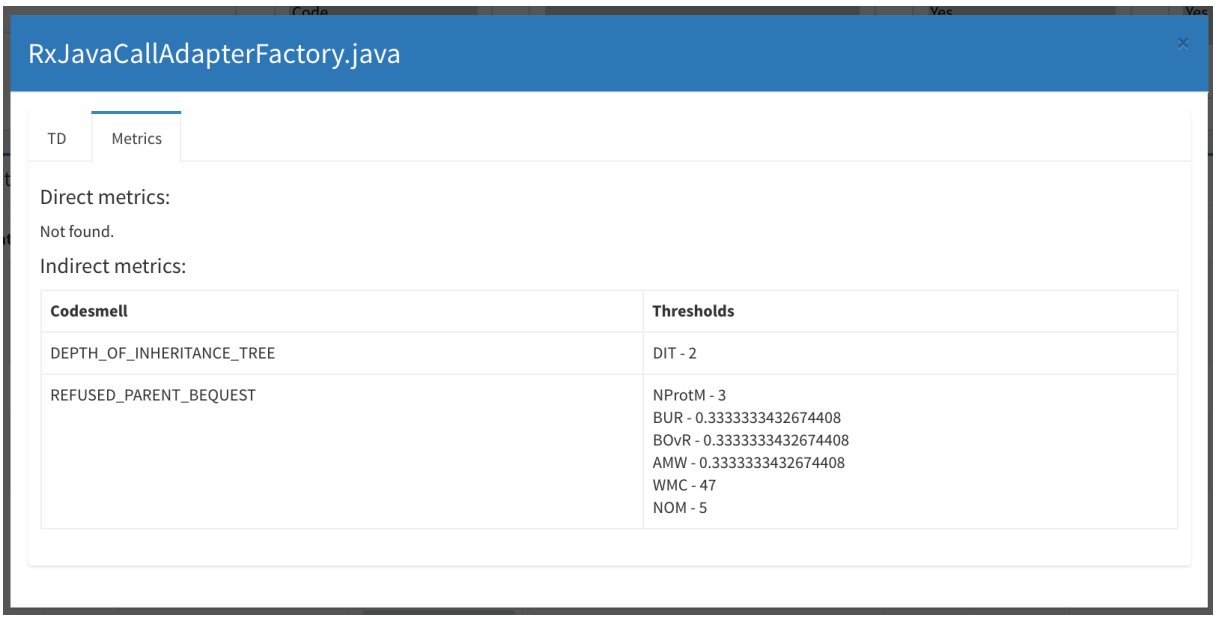
k) “Interest amount”: a quantidade de tempo (em horas) necessária para correção no código em produção (ou seja, no futuro);
l) “Interest probability”: indicador (percentual) das chances de pagar juros no futuro;
m) “Estimates”: indicador da prioridade para correção da DT (low/baixa, médium/média, high/alta);
n) “Notes”: notas/observações que o usuário achar pertinente.

Apenas os campos File, Package e Identification date não são editáveis. Todas as demais informações podem ser alteradas conforme necessidade do usuário.

Já na aba Metrics (métricas) estão as métricas diretas (Direct metrics) e indiretas (Indirect metrics). Segundo Olsina *et al.* (2008), as métricas diretas são quantitativas e não dependem de nenhum outro atributo ou métrica. LOC (*Lines of Code*), por exemplo, são métricas diretas que apresentam a quantidade de linhas presentes no código fonte. As métricas indiretas são consideradas como qualitativas e dependem de outras como atributos, como no caso de God Class, que são classes que centralizam a inteligência em um sistema. Para detecta-la é preciso utilizar três métricas: ATFD, TCC, WMC.

Na Figura 18 temos o exemplo de métricas indiretas de *code smells*. As métricas são parâmetros de referência que o RepositoryMiner utiliza para encontrar indicadores de DT durante o processo de mineração. Os valores de cada item são chamados de *thresholds*. Os valores de referências utilizados pelo RepositoryMiner foram obtidos por trabalhos referenciados na literatura, mas é possível ajustar de acordo com a percepção ou necessidade do usuário.

Figura 18 - *VisminerTD* - Visualizar métricas do item de DT

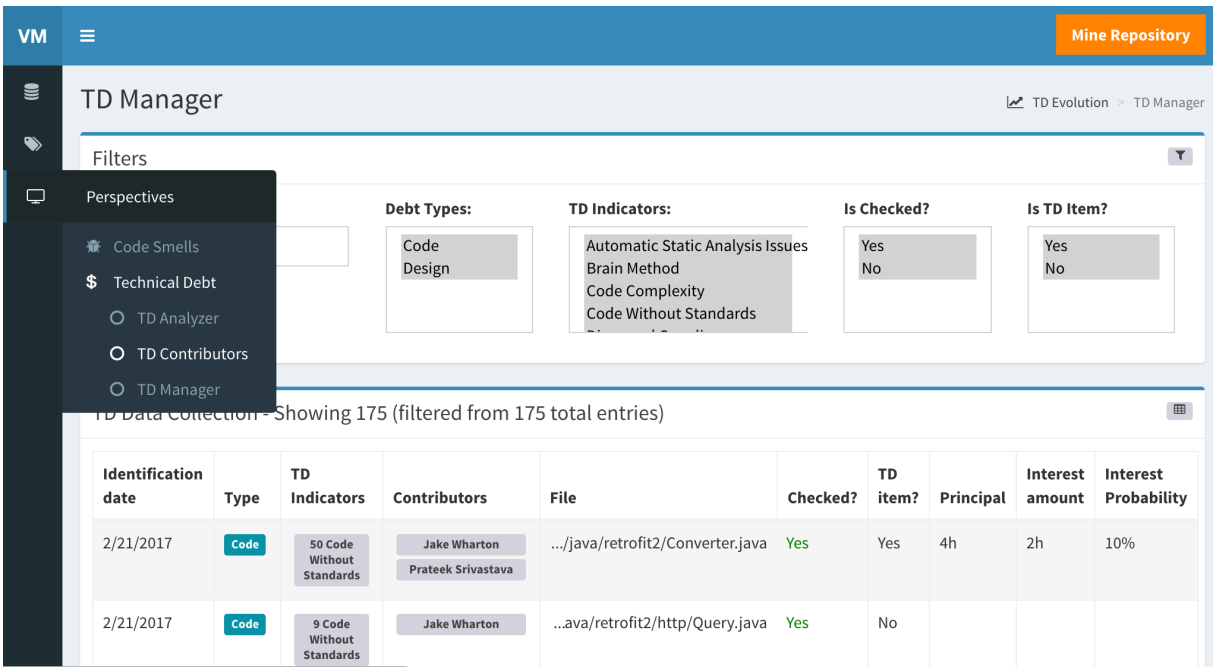


4.3.4 TD Contributors

Para apoiar a realização das atividades 5 e 6 definidas na estratégia, esse componente oferece a opção de acompanhar os indicadores e índices de qualidade dos desenvolvedores. Após o usuário gerenciar as informações dos itens de DT apresentados na seção anterior, ele poderá utilizar a interface que o VisminerTD disponibiliza para visualização dos dados relacionados aos desenvolvedores. É nesta interface que ele obterá indicadores que poderá apoiar a tomada de decisão sobre requalificar a sua equipe de desenvolvimento.

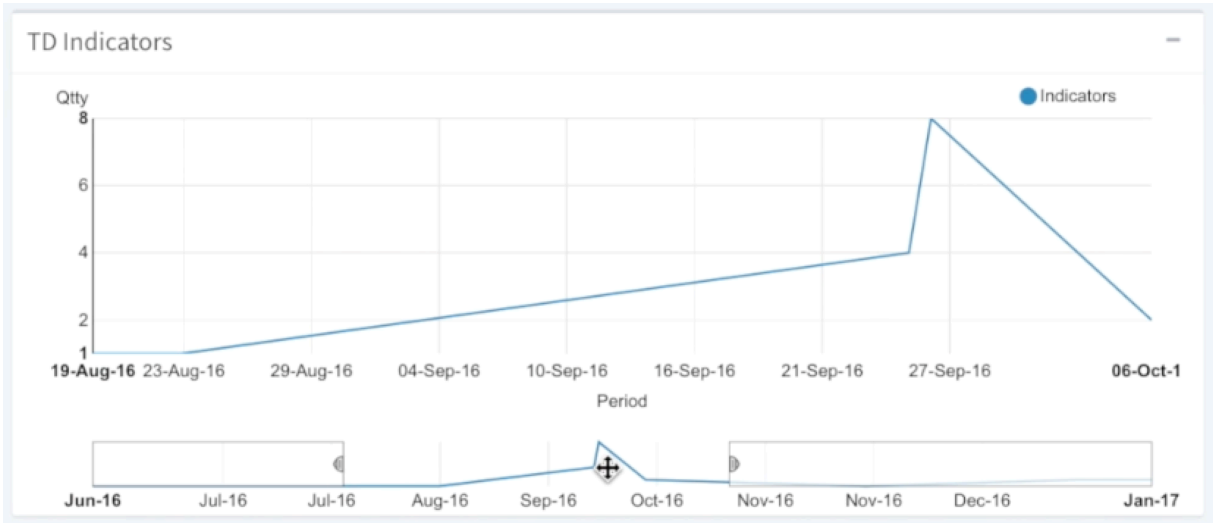
Para ter acesso à interface onde estão os indicadores dos desenvolvedores, o usuário precisa escolher a opção TD Contributors, disponível no menu lateral conforme representação da Figura 19.

Figura 19 - *VisminerTD* - Visualizar indicadores dos desenvolvedores



A interface apresentada será composta de cinco seções: “TD Indicators”, “By Principal”, “ByIndicator”, “ByIntention”, e “ByQuality”. A seção TD Indicators (indicadores de DT) tem como objetivo apresentar a quantidade de indicadores identificados ao longo do tempo. A Figura 20 representa os indicadores da DT. O eixo y exibe a soma da quantidade de indicadores enquanto que o eixo x apresenta a data correspondente.

Figura 20 - *VisminerTD* - Filtros do TD Indicators

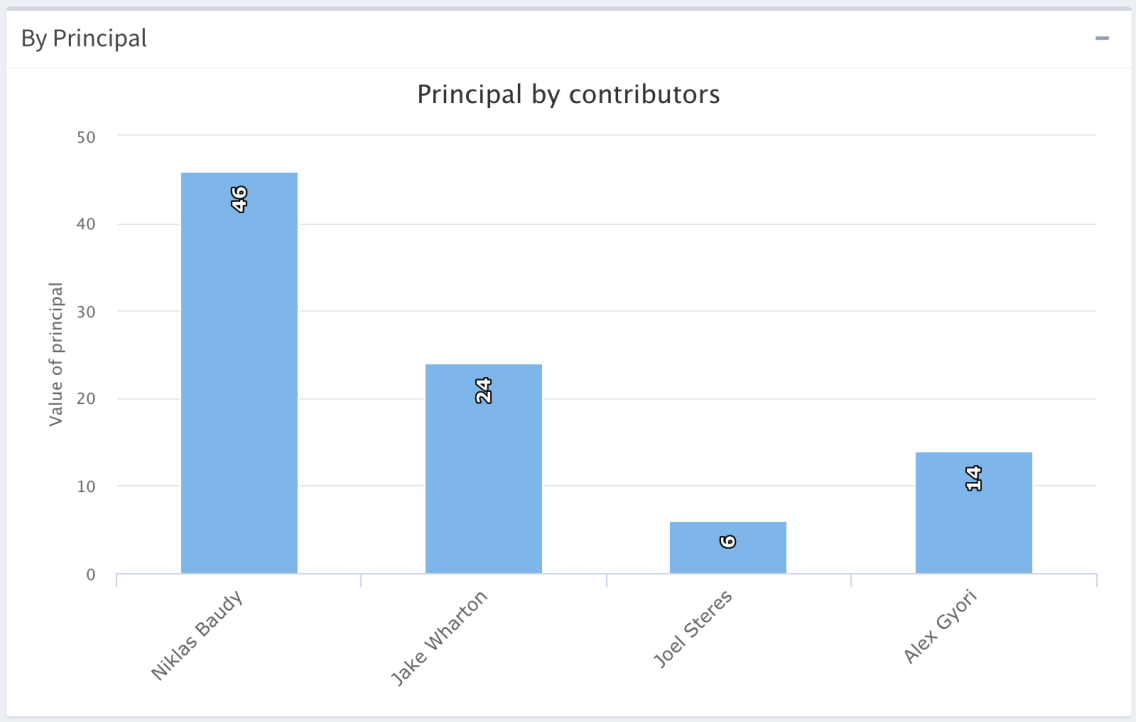


Além disto, apresenta uma interação que permite ao usuário obter informações específicas de um espaço de tempo desejado, selecionando o período desejado. Desta forma,

todas as outras seções serão atualizadas com novas informações referentes a este espaço de tempo.

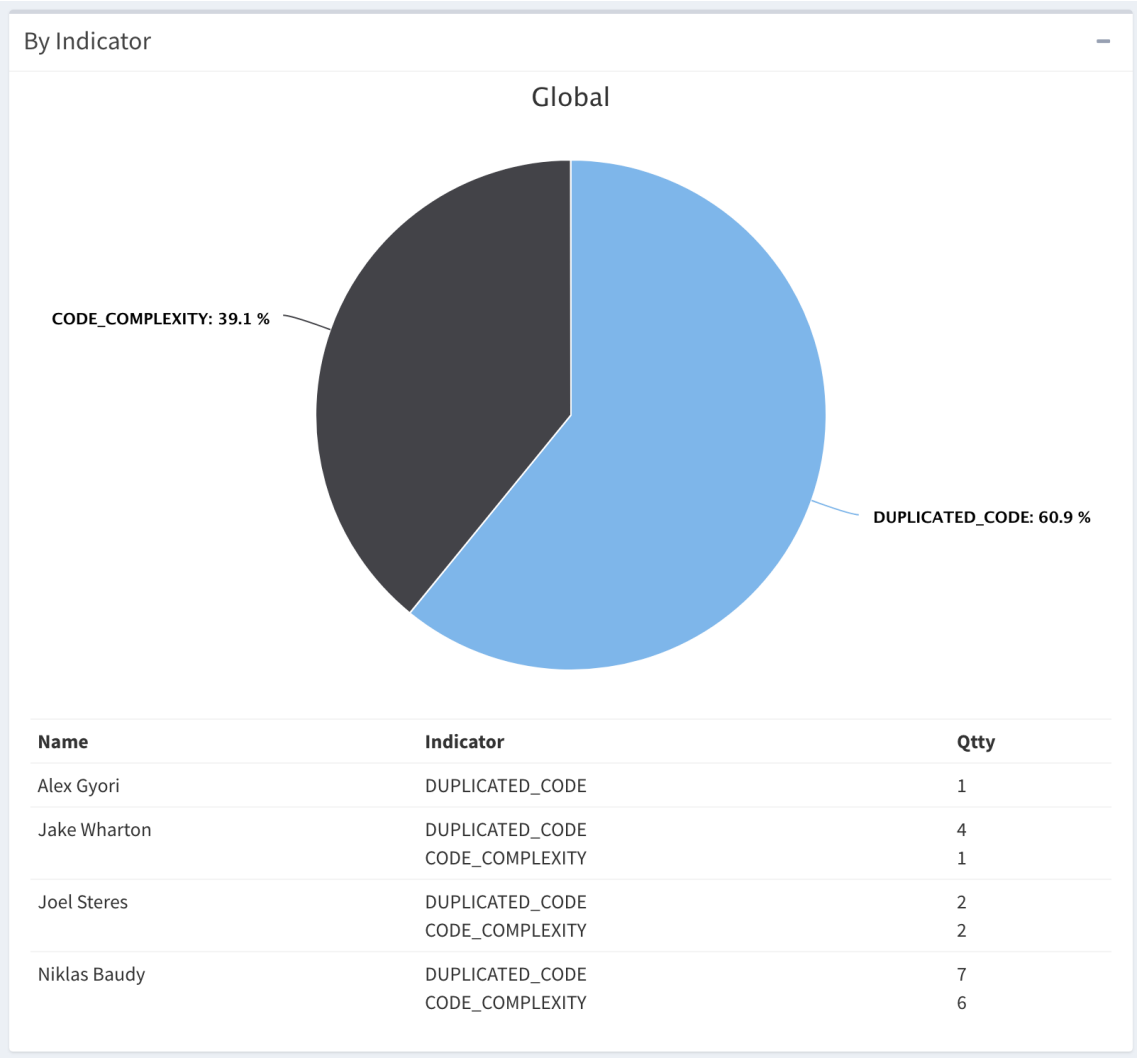
A segunda seção traz informações a respeito dos indicadores de horas do valor da dívida agrupado por colaborador. De acordo com a Figura 21, o gráfico de barras intitulado By Principal traz no eixo y a soma da quantidade do principal (em horas) de cada colaborador. O eixo x traz os nomes dos colaboradores.

Figura 21 - *VisminerTD* - By Principal



A terceira seção apresenta informações dos indicadores de DT de forma global e por colaborador. Conforme apresentação da Figura 22, o VisminerTD apresenta um gráfico de pizza e uma tabela para representar os indicadores da DT. O gráfico apresenta uma visão global (independente de colaborador) e exibe os nomes dos indicadores e quanto cada um representa percentualmente. Em seguida, é apresentada uma tabela contendo os nomes dos colaboradores, os indicadores e a quantidade total de cada indicador agrupado por colaborador.

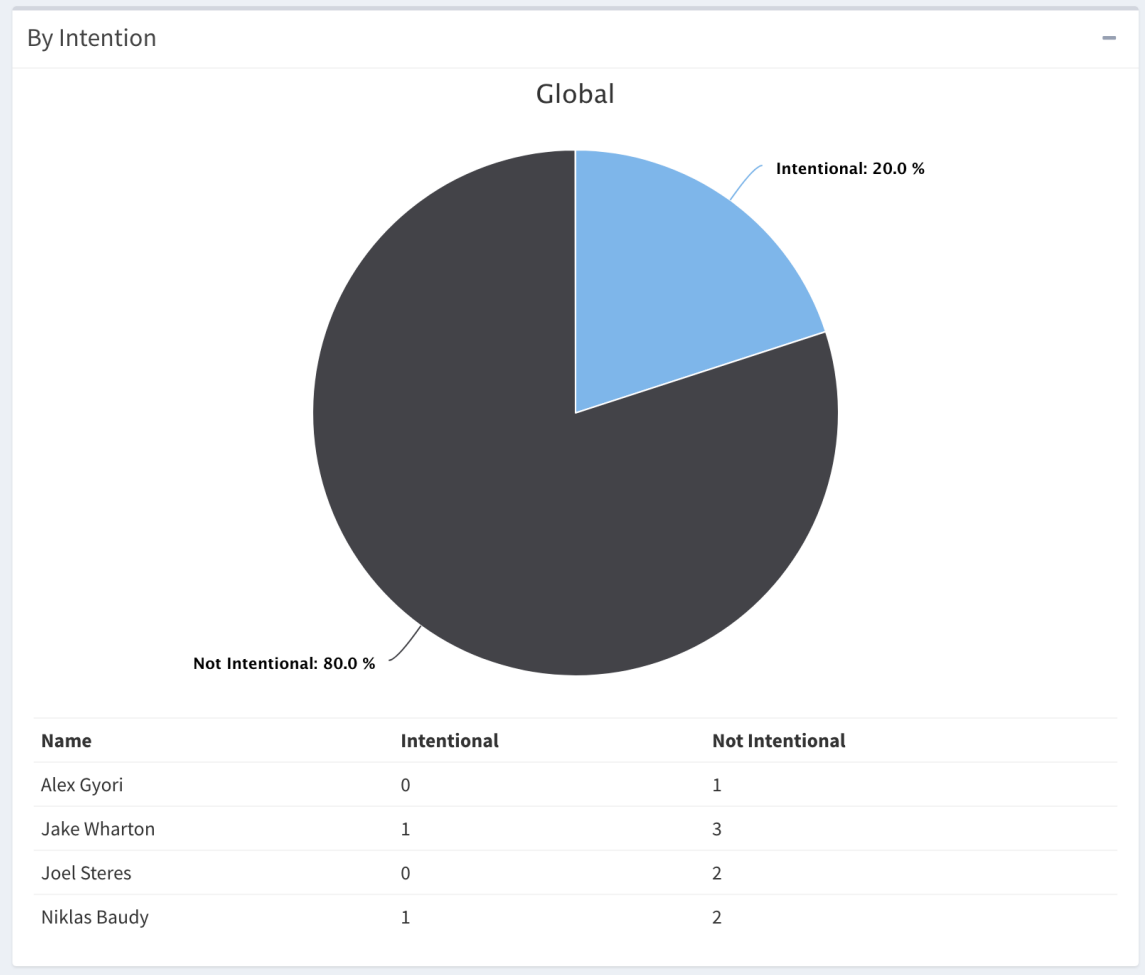
Figura 22 - *VisminerTD* – By Indicator



Name	Indicator	Qtty
Alex Gyori	DUPLICATED_CODE	1
Jake Wharton	DUPLICATED_CODE CODE_COMPLEXITY	4 1
Joel Steres	DUPLICATED_CODE CODE_COMPLEXITY	2 2
Niklas Baudy	DUPLICATED_CODE CODE_COMPLEXITY	7 6

A quarta seção retrata as incidências dos itens DT por intenção. De acordo com a Figura 23, a ferramenta apresenta um gráfico de pizza e uma tabela para representar as incidências da DT de acordo com a intencionalidade. O gráfico apresenta uma visão global (independente de colaborador) e exibe o percentual dos itens de DT que foram intencional e não intencional. No exemplo apresentado, 80% dos itens de DT foram inseridos de forma não intencional. Estas informações podem indicar a necessidade de capacitação e/ou relocação de pessoal. Posteriormente é apresentada uma tabela contendo os nomes dos colaboradores, a soma de itens de DT que surgiram de forma intencional e, em seguida, a soma dos itens de DT não intencionais.

Figura 23 - *VisminerTD* - By Intention



Name	Intentional	Not Intentional
Alex Gyori	0	1
Jake Wharton	1	3
Joel Steres	0	2
Niklas Baudy	1	2

A quinta e última seção retrata indicadores associados à qualidade dos colaboradores no projeto de software. Conforme a apresentação da Figura 24, o VisminerTD traz um gráfico de barras e uma tabela para representar indicadores de qualidade dos colaboradores. O gráfico traz como eixo x os nomes dos colaboradores e como eixo y uma escala de qualidade que vai de 0 a 1 onde 0 é a pior qualidade e 1 é a melhor (o índice de qualidade foi abordado na seção 4.2 deste trabalho). Além disso, a ferramenta apresenta uma tabela contendo os nomes dos colaboradores, a quantidade de commits realizados, a quantidade de commits que tiveram incidência de DT e o índice de qualidade. Por exemplo, analisando a tabela presente na Figura 27, é possível observar que *Nicklas Baudy* foi o colaborador que possui o menor índice de qualidade possível (0). Este fato ocorre porque todos os *commits* submetidos possuem incidência de DT. Por outro lado, o desenvolvedor Joel Steres possui o melhor índice apresentado neste cenário (0.8). Apenas 20% dos commits apresentaram incidência de DT. Os colaboradores Alex Gyori e Jake Warton possuem o índice de qualidade 0.5 e 0.2, respectivamente.

Figura 24 - *VisminerTD* - By Quality

Name	Commits	Commits with TD	Quality Index
Alex Gyori	2	1	0.5
Jake Warton	5	4	0.2
Joel Steres	10	2	0.8
Nicklas Baudy	3	3	0.0

4.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma proposta de estratégia que permite o monitoramento da contribuição de desenvolvedores na evolução da DT. Foram apresentadas formas de obter indicadores e índices que apresentam relevância no processo de análise da participação dos desenvolvedores.

Além disto, também foi apresentada a ferramenta VisminerTD, que apoia a utilização da estratégia proposta. Com três componentes, este software possibilita automatizar o processo de análise de código fonte e usufruir de componentes gráficos para facilitar a leitura e análise dos resultados obtidos.

No capítulo seguinte será apresentado um estudo realizado com objetivo de avaliar a estratégia apresentada.

Capítulo 5 – Avaliação da estratégia

Neste capítulo é apresentado um estudo de caso com o objetivo de avaliar a estratégia proposta neste trabalho. São apresentados o planejamento, execução e resultados do estudo executado.

5.1 INTRODUÇÃO

Este capítulo apresenta o planejamento, execução e resultados de um estudo de caso executado com o objetivo de avaliar a estratégia proposta no capítulo 4. O estudo será realizado para avaliar a viabilidade de uso da estratégia definida como ferramenta de apoio no acompanhamento de itens da DT com o objetivo de monitorar como os membros da equipe de desenvolvimento contribuem para o acúmulo de itens de DT em projetos de software.

Além desta introdução, este capítulo está dividido em mais sete seções. A seção 5.2 retrata o objetivo do estudo de caso executado. A seção 5.3 apresenta o planejamento do estudo, sendo os instrumentos utilizados apresentados na seção 5.4. Em seguida, a seção 5.5 apresenta os resultados obtidos. Na seção 5.6 é realizada a discussão dos resultados. Já as ameaças à validade do estudo são apresentadas na seção 5.7. Por fim, a seção 5.8 apresenta as considerações finais do capítulo.

5.2 OBJETIVO DO ESTUDO

O estudo de caso realizado neste trabalho tem como objetivo **analisar** a estratégia de monitoramento de itens de DT proposta, **com o objetivo de** caracterizá-la **com respeito a** utilidade, facilidade de uso e possível uso futuro da estratégia no apoio ao monitoramento sobre como os membros da equipe contribuem para o acúmulo ou eliminação de itens de DT, **sob o ponto de vista de** estudantes de graduação cursando a disciplina de engenharia de software **no contexto de** projetos de desenvolvimento de software.

Considerando que os participantes do estudo irão utilizar uma nova tecnologia, as questões de pesquisas do formulário de avaliação estarão fundamentadas no *Technology Acceptance Model* (TAM), descrito por Davis *et al.* (1989). Segundo Sardinha *et al.* (2011), a utilização do TAM permite verificar a utilidade e facilidade de uso percebida aos usuários utilizarem uma nova tecnologia.

A caracterização da utilidade possibilita avaliar se a proposta é útil no processo de identificação e avaliação dos itens de DT para nortear como cada desenvolvedor contribui com o surgimento da DT.

Como cenário, será utilizado um projeto real de código aberto disponível na comunidade GitHub.

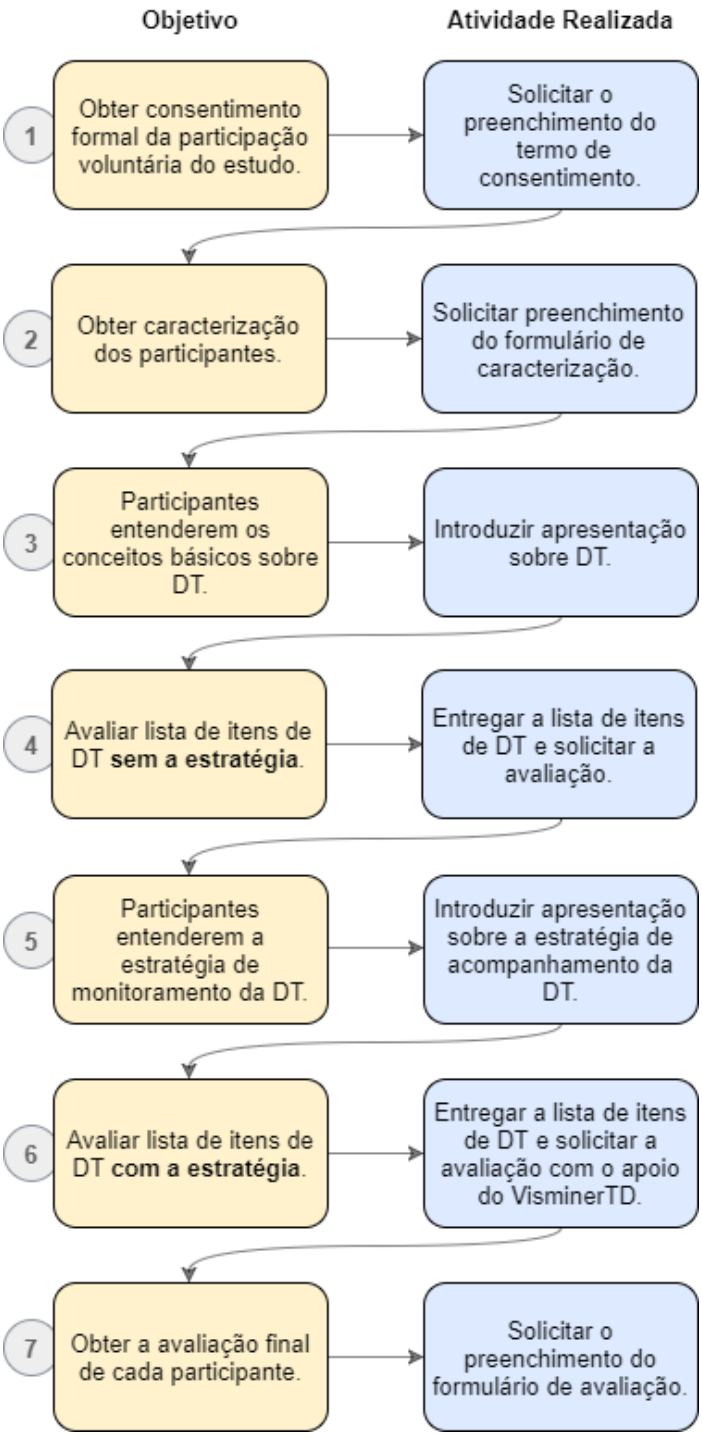
5.3 PROCEDIMENTO E COLETA DE DADOS

Como parte do planejamento do estudo, é preciso considerar um ambiente de desenvolvimento de software para obter dados de sua evolução a partir de interações de desenvolvedores. Para isso, foi utilizado um projeto real, de código aberto, para que o ambiente seja mais próximo de cenários reais de desenvolvimento.

Os participantes do estudo serão estudantes de graduação cursando a disciplina de engenharia de software, em duas turmas distintas. Ambas as turmas executarão os mesmos processos: serão solicitados a identificar a contribuição dos desenvolvedores na incidência de itens da DT em um projeto de software previamente definido. A finalidade do procedimento é avaliar, a partir de uma lista de itens de DT, quais são os desenvolvedores que mais contribuem com a evolução da DT.

A Figura 25 apresenta os objetivos e etapas realizadas no processo de execução do estudo. Inicialmente serão apresentados o termo de consentimento e formulário de caracterização para que todos os participantes leiam e os assinem. Como pré-requisito da avaliação, será realizada uma capacitação para o público alvo acerca do contexto da DT. O objetivo desta etapa é fornecer o embasamento teórico sobre o tema.

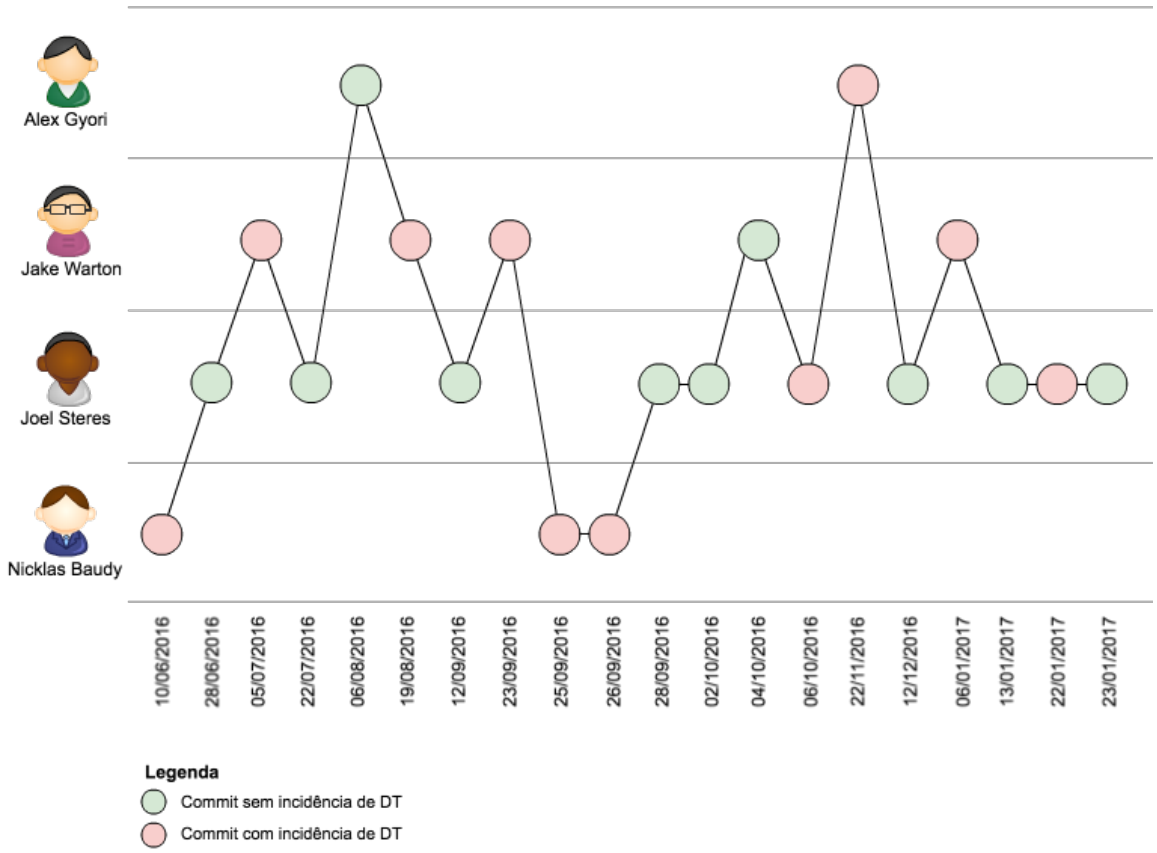
Figura 25 - Procedimentos para a aplicação do estudo



Os participantes do estudo serão então apresentados a um cenário de desenvolvimento, no software Retrofit. O projeto conta um grande número de *commits* e desenvolvedores (apresentados na seção 5.3.1). Desta forma, foi selecionada uma amostra contendo uma quantidade menor de commits, e que tenha a participação de mais de um colaborador. Sendo assim, foram selecionados 20 *commits*, envolvendo 4 desenvolvedores, em datas distintas.

Conforme apresentado na Figura 29, 10 *commits* tiveram incidência de DT, totalizando 13 itens de DT.

Figura 26 - Amostra dos *commits* utilizados no estudo



Juntamente com a Figura 26, foi elaborada uma tabela complementando as informações da DT presente neste cenário, conforme pode ser observado na Tabela 4.

ID	Identificação	Tipo	Item de DT	Ocorrido por	Localidade	Descrição
1	10/06/2016	Código	Código duplicado	Niklas Baudy	Query.java	Possui o mesmo trecho de código em 2 métodos
2	05/07/2016	Código	Código duplicado	Jake Wharton	MockRetrofit.java	
3	19/08/2016	Código	Código duplicado	Jake Wharton	GsonRequestBodyConverter.java	O método getStatus precisa ser refatorado
4	23/08/2016	Código	Código duplicado	Jake Wharton	PartMap.java	Duplicação presente no método publicvoidcancel()
5	25/09/2016	Código	Código duplicado	Niklas Baudy	RxJavaCallAdapterFactoryTest.java	
6	26/09/2016	Código	Código complexo Código duplicado	Niklas Baudy	RecordingMaybeObserver.java	
7	06/10/2016	Código	Código complexo Código duplicado	Joel Steres	CompletableThrowingTest.java	Duplicação no método public Executor callbackExecutor()
8	22/11/2016	Código	Código duplicado	Alex Gyori	RxJava2CallAdapterFactoryTest.java	
9	06/01/2017	Código	Código complexo Código duplicado	Jake Wharton	CompletableThrowingTest.java	
10	22/01/2017	Código	Código complexo Código duplicado	Joel Steres	package-info.java	Possui código duplicado no construtor

Tabela 4 - Itens de DT apresentados nos *commits*

Nessa tabela, os itens de DT são detalhados considerando as seguintes características:

a) ID: número incremental de identificador;
b) Data de identificação: data que a dívida foi ocorrida;
c) Tipo: tipo da DT. Exemplo: código, documentação, etc.;
d) Item de DT: indicador da presença da dívida. Exemplo: código complexo, código duplicado, etc.;
e) Ocorrido por: nome do desenvolvedor;
f) Localidade: nome do arquivo;
g) Descrição: informativo sobre o item de DT.

De posse do cenário apresentado na Figura 26, foi solicitado aos participantes que realizassem as seguintes atividades:

a) Considerando a quantidade de contribuição (*commit*) de cada membro da equipe de desenvolvimento com o projeto, classifique os membros do mais atuante para o menos atuante.
   O objetivo desta solicitação é identificar, de uma forma geral (sem considerar a presença de DT), quais são os integrantes que mais interagem com o projeto.
b) Considerando a quantidade de commits (geral - com e sem DT) e a quantidade total de commits com DT de cada colaborador, classifique os colaboradores pela qualidade de seus commits (da melhor qualidade para a menor qualidade).
   O propósito desta solicitação é identificar a qualidade dos commits de cada colaborador, obtido pela razão da quantidade de commits com DT e quantidade total de commits (com e sem DT).
c) Levando em consideração a quantidade de horas previstas para o pagamento de cada item, classifique os integrantes que mais contribuíram com o acúmulo de horas (do mais relevante para o menos relevante);
   O objetivo deste item é identificar e classificar os integrantes de acordo com o acúmulo de horas previstas para correção da DT no projeto através de uma lista ordenada de forma decrescente, onde os colaboradores que mais contribuíram com a DT apareçam primeiro.

d) Observando os itens de DT, ordene-os conforme número de ocorrência encontrada (em ordem decrescente);

O propósito desta solicitação é saber quais são os itens de DT que afetam o projeto de software conforme número de ocorrência/repetição.

e) Observando os itens de DT, informe o percentual que surgiu de forma não intencional;

O propósito desta solicitação é saber qual a proporção de itens de DT que afetam o projeto de forma não intencional. Este dado pode apontar o desconhecimento do desenvolvedor em relação a más prática de codificação e conduzir a uma capacitação para reprimir reincidências futuras.

Após coletar as respostas, será realizado um segundo treinamento para descrever a estratégia de monitoramento da evolução da DT proposta neste trabalho. Para apoiar a realização desta etapa do estudo de caso, foi disponibilizado (como parte da estratégia de monitoramento proposta) um complemento para a tabela de itens de DT, que pode ser visualizada parcialmente na Tabela 5 (a versão completa pode ser encontrada no Apêndice A). É possível observar que três novos campos foram adicionados à tabela inicial:

Tabela 5 - Complemento dos itens de DT apresentados nos *commits*

ID	Identificação	Tipo	Item de DT	Ocorrido por	Localidade	Descrição	Intencional?	Principal (horas)	Juros (horas)	Probabilidade de juros (%)
1	10/06/2016	Código	Código duplicado	Niklas Baudy	Query.java	Possui o mesmo trecho de código em 2 métodos	Não	6	2	20
2	05/07/2016	Código	Código duplicado	Jake Wharton	MockRetrofit.java		Não	8	4	20
3	19/08/2016	Código	Código duplicado	Jake Wharton	GsonRequestBodyConverter.java	O método getStatus precisa ser refatorado	Sim	4	2	10
4	23/08/2016	Código	Código duplicado	Jake Wharton	PartMap.java	Duplicação presente no método publicvoid cancel()	Não	2	1	20
5	25/09/2016	Código	Código duplicado	Niklas Baudy	RxJavaCallAdapterFactoryTest.java		Sim	10	4	30
6	26/09/2016	Código	Código complexo Código duplicado	Niklas Baudy	RecordingMaybeObserver.java		Sim	30	10	40
7	06/10/2016	Código	Código complexo Código duplicado	Joel Steres	CompletableThrowingTest.java	Duplicação no método public Executor	Não	4	3	30

						callbackExecutor()				
8	22/11/2016	Código	Código duplicado	Alex Gyori	RxJava2CallAdapterFactoryTest.java		Não	14	4	30
9	06/01/2017	Código	Código complexo Código duplicado	Jake Wharton	CompletableThrowingTest.java		Não	10	4	40
10	22/01/2017	Código	Código complexo Código duplicado	Joel Steres	package-info.java	Possui código duplicado no construtor	Não	2	1	50

a) Principal: quantidade de horas necessárias para pagar a DT;
b) Juros: quantidade de horas extras necessárias para pagar a DT em um momento futuro, caso a mesma seja mantida;
c) Probabilidade de juros: percentual que indica a probabilidade de pagamento dos juros.

Além disso, a estratégia proposta também disponibiliza os dados apresentados na Tabela 6, contendo informações consolidadas sobre cada desenvolvedor. Os dados exibidos são: a quantidade total de *commits* (com e sem DT), a quantidade total de *commits* com DT, o Índice de Qualidade, o total de horas e o percentual em relação ao total de todos os colaboradores.

Tabela 6 - Índice de qualidade e percentual de participação nas horas para pagamento da DT de cada desenvolvedor

Colaborador	Commits	Commits com DT	Índice de Qualidade (1 – (commits com DT / commits))	Total de horas	Percentual da participação total de horas de todos os colaboradores
Alex Gyori	2	1	0.50	14	16%
Jake Warton	5	4	0.20	24	27%
Joel Steres	10	2	0.80	6	7%
Nicklas Baudy	3	3	0.00	46	51%

Para apoiar a utilização da estratégia, foi apresentada a ferramenta VisminerTD. Realizou-se uma carga de dados na ferramenta para que represente as mesmas informações apresentadas nas tabelas e imagens utilizadas anteriormente pelos participantes. Seguidamente, foram exibidos vídeos demonstrando todos os processos apresentados na sessão 4.3 deste trabalho.

Em seguida, os participantes serão solicitados a responder as mesmas atividades, solicitadas anteriormente, mas desta vez com as tabelas que compõem a estratégia definida.

Ao final, os participantes responderão a um questionário de percepção de uso da estratégia proposta.

5.3.1 Critérios e seleção do projeto de software utilizado

Para avaliar a proposta de monitoramento, foi necessário utilizar um projeto de software real. A escolha do projeto se deu através de busca realizada no serviço de armazenamento de projetos GIT chamado GitHub.

Com mais de 53 milhões de repositórios mantidos em março de 2017 (GITHUB, 2017), o GitHub é o maior e mais popular serviço de armazenamento de repositórios GIT (KALLIAMVAKOU *et al.*, 2014). Ele permite o armazenamento de projetos de código aberto e oferece um ambiente social de trabalho que estabelece relação entre usuários e seus artefatos de trabalho de forma transparente (TSAY *et al.*, 2014).

O GitHub oferece uma ferramenta de pesquisa em seu acervo de repositórios públicos. Utilizando a ferramenta no modo avançado, é possível restringir os resultados ao construir consultas mais criteriosas, sendo possível filtrar os resultados por uma variedade de fatores como o número de *forks* ou estrelas que um repositório possui. *Fork* é a denominação ao processo de obter uma cópia de um repositório, que permite experimentar um repositório livremente sem que as mudanças efetuadas afetem o projeto original. É possível propor mudanças no repositório original através de um repositório fruto de um *fork* (GITHUB, 2017).

Os parâmetros de pesquisa utilizados foram:

a) Projeto ser escrito na linguagem de programação Java: o minerador de dados (RepositoryMiner) até o presente momento efetua análise apenas em projetos desenvolvidos nesta linguagem de programação;
b) Possuir mais de 4.000 *forks*: o objetivo é obter o projeto que seja mais popular na comunidade GitHub. Dessa forma, foi estabelecida uma grande quantidade de *forks* para eliminar projetos menos populares dos resultados do filtro.

O projeto posicionado no topo da lista no resultado da pesquisa foi o Retrofit. Ele é um cliente HTTP do tipo seguro para Android e Java, mantido pela empresa Square (RETROFIT, 2017). O uso do Retrofit facilita a manipulação de dados JSON (ou outros tipos

de dados estruturados) através de serviços baseados em REST, tornando possível configurar qual conversor é utilizado para a serialização de dados. Para JSON, por exemplo, o conversor utilizado é o Gson, mas é possível personalizar para processar XML ou até mesmo outros protocolos (VOGUELLA, 2017). O repositório do Retrofit conta com 1.442 *commits*, 11 *branches*, 39 *tags*/*releases* e 105 contribuidores.

5.4 INSTRUMENTAÇÃO

Para apoiar o estudo na coleta de dados, foram utilizados o termo de consentimento, o formulário de caracterização do participante e o formulário de avaliação. O termo de consentimento apresentado aos participantes descreve o tema, objetivos do estudo, instituição responsável, informações sobre o pesquisador, dados sobre a confidencialidade, e o termo do consentimento propriamente dito (disponível no APÊNDICE A).

O formulário de caracterização do participante tem como objetivo entender o nível de experiência e conhecimento do indivíduo sobre desenvolvimento de software e dívida técnica. Disponível no APÊNDICE A – A2, ele possui quatro questões para obter informações sobre o integrante do estudo, sendo três sobre a experiência em relação ao desenvolvimento de software e um sobre conhecimentos em DT.

Já o formulário de avaliação, disponível no APÊNDICE A – A3 contém oito questões relacionadas ao monitoramento da contribuição de desenvolvedores no surgimento e evolução da dívida técnica. Neste formulário, cada participante deve avaliar afirmações relacionadas à utilidade, facilidade de uso, produtividade, velocidade e possível uso da estratégia proposta e indicar a opção que melhor representa seu ponto de vista de acordo com a seguinte escala: (1) Concordo totalmente; (2) Concordo parcialmente; (3) Neutro; (4) Discordo parcialmente; e (5) Discordo totalmente.

As afirmações que devem ser avaliadas são:

1. Em relação à utilidade:
    a. Usando a estratégia proposta, eu seria capaz de identificar e avaliar itens de DT mais rapidamente.
    b. Usando a estratégia proposta, eu iria melhorar o meu desempenho na avaliação de itens de DT.
    c. Usando a estratégia proposta, eu iria melhorar a minha eficácia em avaliar itens de DT.

d. Usando a estratégia proposta, eu seria capaz de identificar quais são todos os desenvolvedores que trabalharam na evolução do software.
e. Usando a estratégia proposta, eu seria capaz de identificar quais são os desenvolvedores que mais contribuem com o surgimento da DT.
f. Usando a estratégia proposta, seria mais fácil de decidir quais são os itens de DT mais relevantes para pagamento.
g. Eu acredito que a estratégia proposta seria útil para a tomada de decisão na qualificação da equipe de desenvolvimento.

Correspondente à produtividade, as afirmações a serem avaliadas são:

h. Eu acredito que através de dados obtidos com a estratégia proposta eu poderei capacitar melhor a equipe de desenvolvimento para torná-la mais produtiva.
i. Seria mais fácil identificar a necessidade de efetuar uma rotatividade de pessoal para tornar a equipe mais produtiva com a estratégia proposta.

Quanto à velocidade, as seguintes afirmações deverão ser avaliadas:

j. Eu acredito que com a estratégia proposta eu serei mais rápido no processo de identificação e monitoramento da equipe de desenvolvimento.
k. Eu considero que a estratégia proposta otimizaria o tempo e proporcionaria agilidade no processo de tomada de decisão para melhorar a equipe de desenvolvimento.

2. Quanto à facilidade de uso, as seguintes afirmações deverão ser avaliadas:
    a. Aprender a utilizar a estratégia proposta seria fácil para mim.
    b. Minha interação com a estratégia proposta seria clara e compreensível.
    c. Eu acharia fácil de utilizar a estratégia proposta para avaliar itens de DT.
    d. Seria fácil tornar-se hábil no uso da estratégia proposta.
    e. Seria fácil se lembrar de como executar a avaliação e monitoramento da DT utilizando a estratégia proposta.
    f. Eu acredito que a estratégia proposta é fácil de utilizar.
3. Em relação ao possível uso futuro, as afirmações a serem avaliadas são:
    a. Assumindo que a estratégia proposta estaria disponível para monitorar a DT, eu a utilizaria no futuro.

b. Eu preferiria utilizar a estratégia proposta a avaliar e monitorar a DT de maneira usual (sem o apoio da ferramenta).

Ademais, os participantes devem escrever suas percepções a respeito dos aspectos positivos, negativos e sugestões de melhorias da estratégia.

Por fim, as Tabelas 4, 5 e 6 descritas na seção anterior, o material de treinamento e a estratégia proposta também compõem o conjunto de instrumentos do estudo.

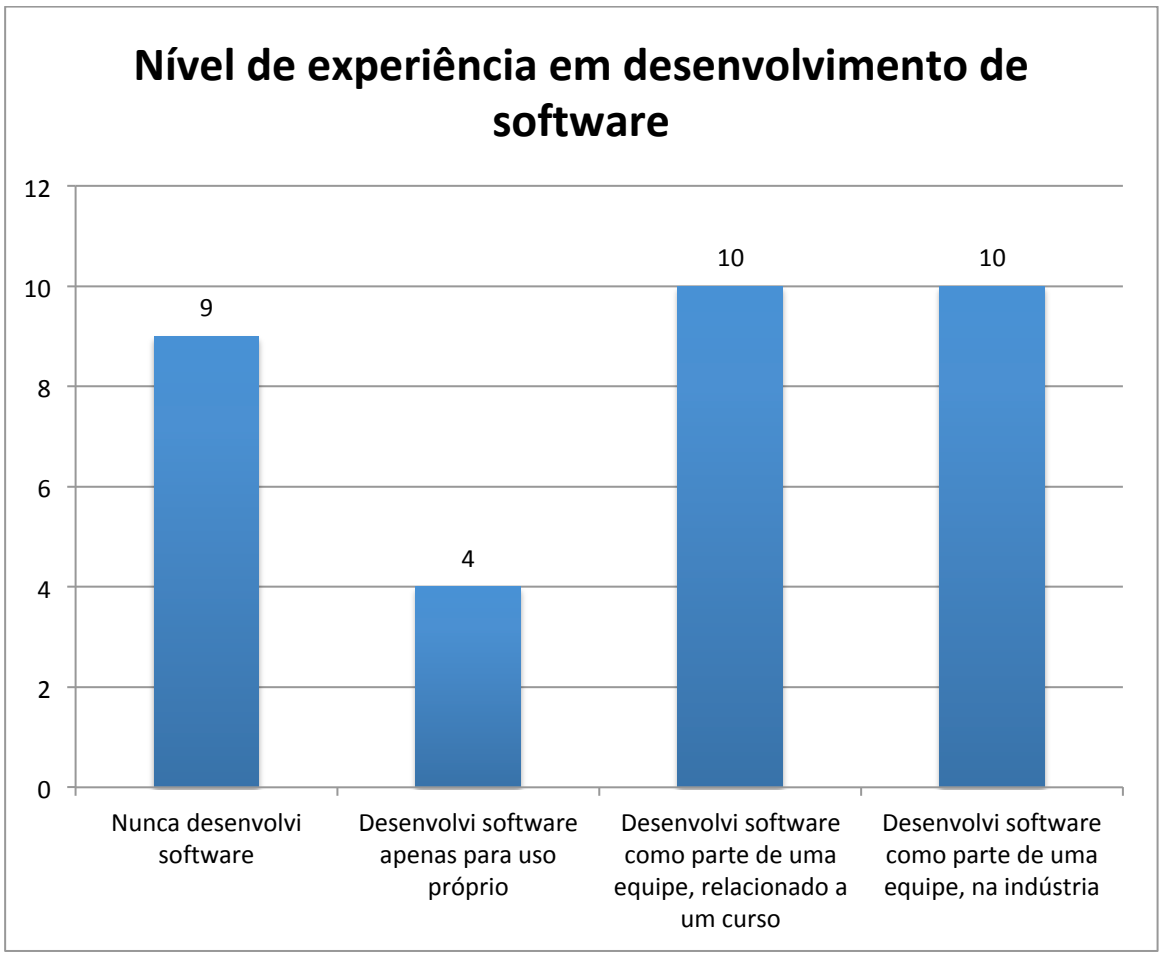
5.5 RESULTADOS

Os participantes deste estudo são alunos da Universidade Salvador (UNIFACS), no nível de graduação, que estão cursando a disciplina Engenharia de Software, em duas turmas. As avaliações foram realizadas com uma turma no dia 27/11/2017 e na outra no dia 12/12/2017. A quantidade total de participantes foi de 33 alunos, sendo que a atividade foi realizada de forma individual.

5.5.1 Caracterização dos participantes

Os primeiros resultados analisados foram sobre o nível de experiência dos participantes em relação ao desenvolvimento de software, que estão apresentados na Figura 27. De acordo com os resultados obtidos, observa-se que nem todos os participantes possuem experiência prática com desenvolvimento de software, no entanto, conceitos teóricos foram explorados no decorrer da disciplina de engenharia de software sobre o assunto.

Figura 27 - Experiência dos participantes em relação ao desenvolvimento de software

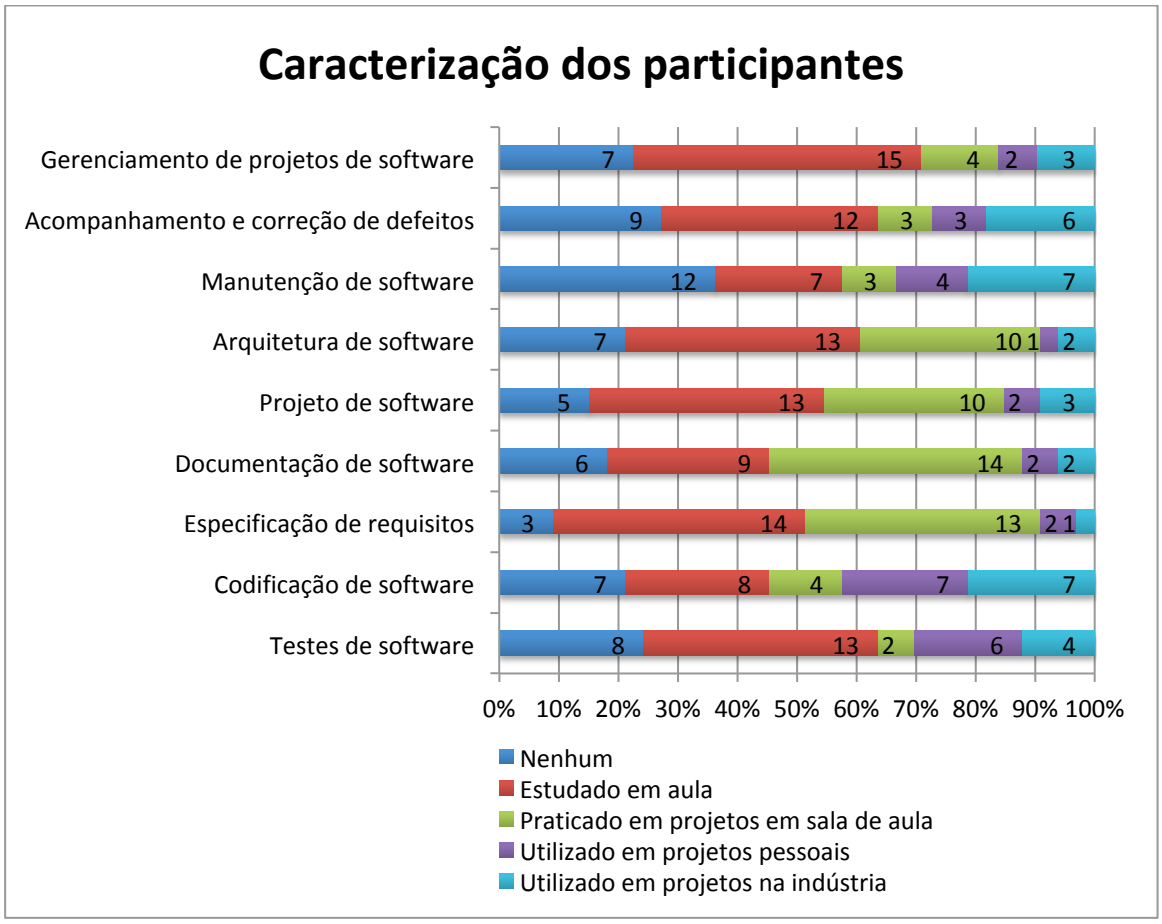


Para complementar as informações sobre o nível de experiência dos participantes, foi solicitado que eles informassem a quantidade de anos de experiência em relação ao desenvolvimento de software. O resultado é apresentado na Figura 28. Conforme pode ser observado, a maioria dos participantes indicou ter um ou mais anos de experiência em desenvolvimento de software. Entretanto, uma quantidade considerável indicou não possuir experiência com atividades de desenvolvimento de software.

Figura 28 - Tempo de experiência em desenvolvimento de software

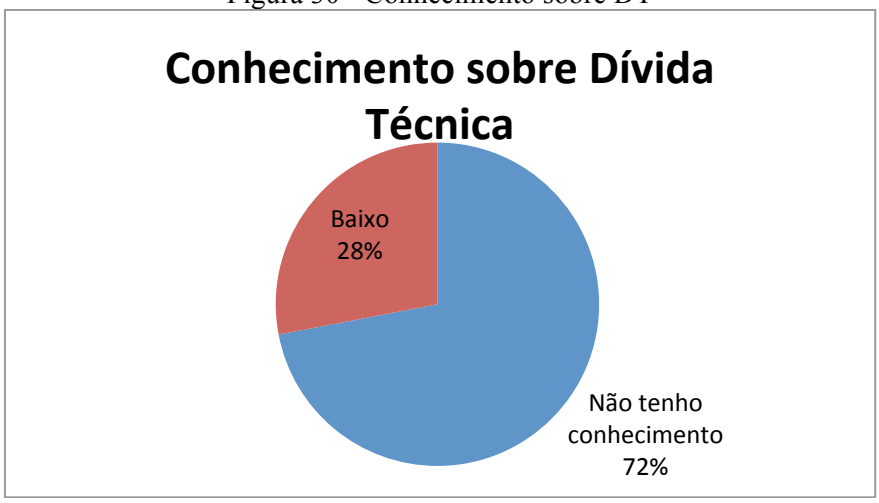
Em seguida, na Figura 29, são apresentados os resultados sobre o nível de experiência entre nove atividades de desenvolvimento de software. Observa-se que, para todas as áreas apresentadas, existem participantes que possuem experiência na indústria.

Figura 29 - Experiência em desenvolvimento de software



Posteriormente, perguntou-se sobre o conhecimento dos participantes sobre o tema DT. Como pode ser observado na Figura 30, a maioria dos participantes (72%) não possuía conhecimento prévio no assunto. Esse já era um resultado esperado. Desta forma, foi planejado um treinamento sobre o assunto que foi realizado logo após a caracterização dos participantes.

Figura 30 - Conhecimento sobre DT



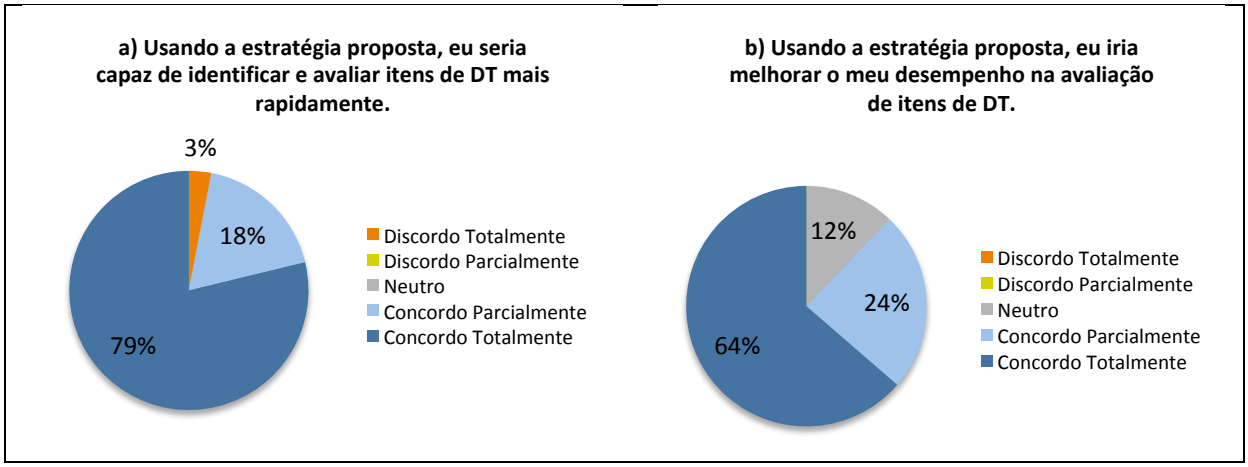
5.5.2 Análise da estratégia

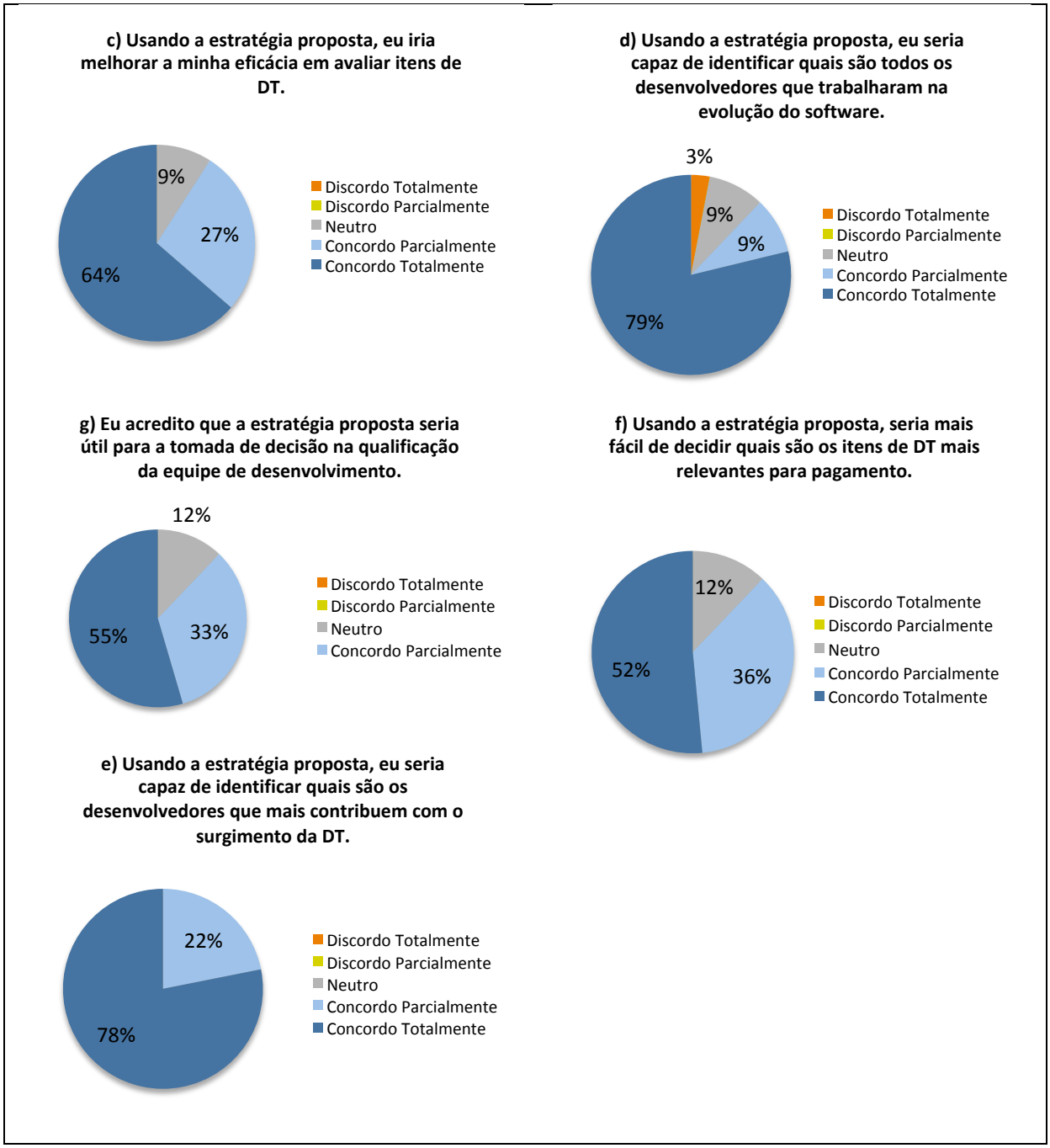
Esta seção apresenta a análise das respostas dos participantes em relação à estratégia proposta. Os participantes foram solicitados a responder questões sobre a utilidade, facilidade de uso e possível uso da estratégia proposta. Ademais, eles deveriam apresentar pontos positivos e negativos, bem como sugestões de melhoria para a estratégia.

5.5.2.1 Utilidade

Sobre a utilidade, os participantes responderam a sete questões (conforme indicado na seção 5.4) e optaram pela melhor opção que representava a sua opinião de acordo com a escala seguinte: (1) Concordo totalmente; (2) Concordo parcialmente; (3) Neutro; (4) Discordo parcialmente; e (5) Discordo totalmente. Os resultados estão apresentados na Figura 31.

Figura 31 - Utilidade com a estratégia





Na afirmação **a) Usando a estratégia proposta, eu seria capaz de identificar e avaliar itens de DT mais rapidamente**, 97% dos participantes concordaram que a estratégia torna o processo de avaliação mais rápido. Os outros 3% demonstraram ser neutro e não houve nenhuma manifestação de discordância. Em relação à afirmação **b) Usando a estratégia proposta, eu iria melhorar o meu desempenho na avaliação de itens de DT**, 88% dos participantes afirmaram que há um ganho de desempenho na avaliação de itens de DT ao utilizar a estratégia proposta. Os outros 12% se mantiveram neutro e não houve registro de discordância.

Na afirmação **c) Usando a estratégia proposta, eu iria melhorar a minha eficácia em avaliar itens de DT**, 91% dos participantes concordaram com o aumento da eficácia ao utilizar a estratégia enquanto 9% se demonstraram ser neutro. Não houve manifestação de discordância. Em seguida, na afirmação **d) Usando a estratégia proposta, eu seria capaz de identificar quais são todos os desenvolvedores que trabalharam na evolução do software**, 88% dos participantes indicaram que saberiam identificar todos os desenvolvedores que atuaram no projeto de software enquanto que 9% se manifestaram como neutros e apenas um participante discordou.

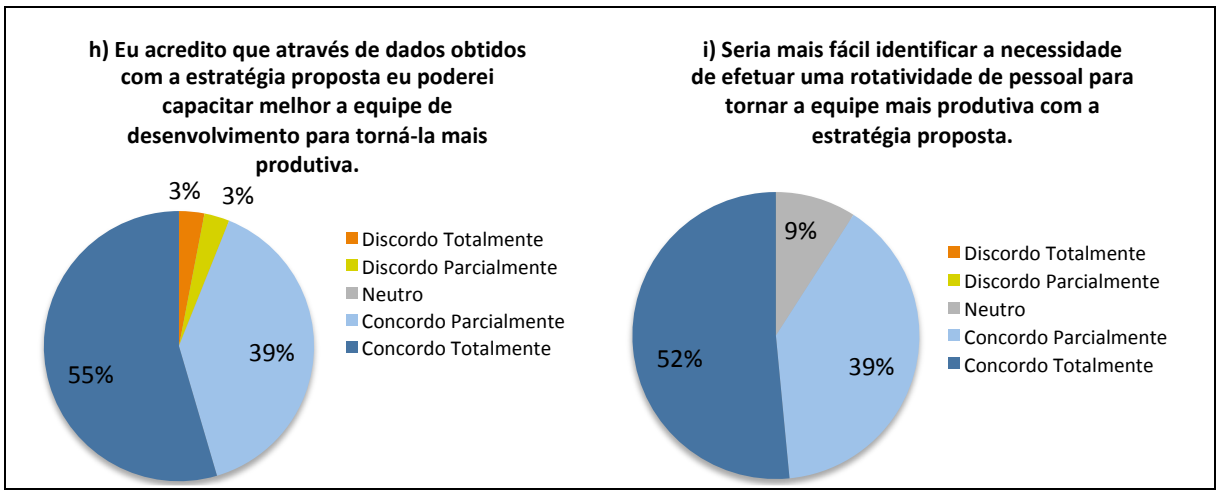
Todos os participantes concordaram com a afirmação **e) Usando a estratégia proposta, eu seria capaz de identificar quais são os desenvolvedores que mais contribuem com o surgimento da DT**. Este resultado foi bastante favorável a respeito da aceitação da estratégia.

Também não houve discordância com a afirmação **f) Usando a estratégia proposta, seria mais fácil de decidir quais são os itens de DT mais relevantes para pagamento**. 88% dos participantes concordaram enquanto apenas quatro participantes se manifestaram como neutro. Resultado semelhante foi obtido para a afirmação **g) Eu acredito que a estratégia proposta seria útil para a tomada de decisão na qualificação da equipe de desenvolvimento**. Apenas quatro participantes se mantiveram como neutro enquanto 88% se concordaram que a estratégia apoia no processo de tomada de decisão para a qualificação da equipe de desenvolvimento.

Dos 33 participantes apenas 1 discordou, sendo apenas em 2 das 7 questões. Esta informação indica que a estratégia é útil no processo de identificação da DT e na tomada de decisão na qualificação da equipe de desenvolvimento.

Em seguida, os participantes responderam a duas afirmações a respeito da produtividade na utilização da estratégia (apresentadas na seção 5.4), marcando a opção que melhor representava sua opinião conforme a seguinte escala: (1) Concordo totalmente; (2) Concordo parcialmente; (3) Neutro; (4) Discordo parcialmente; e (5) Discordo totalmente. Os resultados estão disponíveis na Figura 32. De acordo com as respostas coletadas, a maioria dos participantes concordaram com o ganho produtividade possibilitado através do uso da estratégia apresentada.

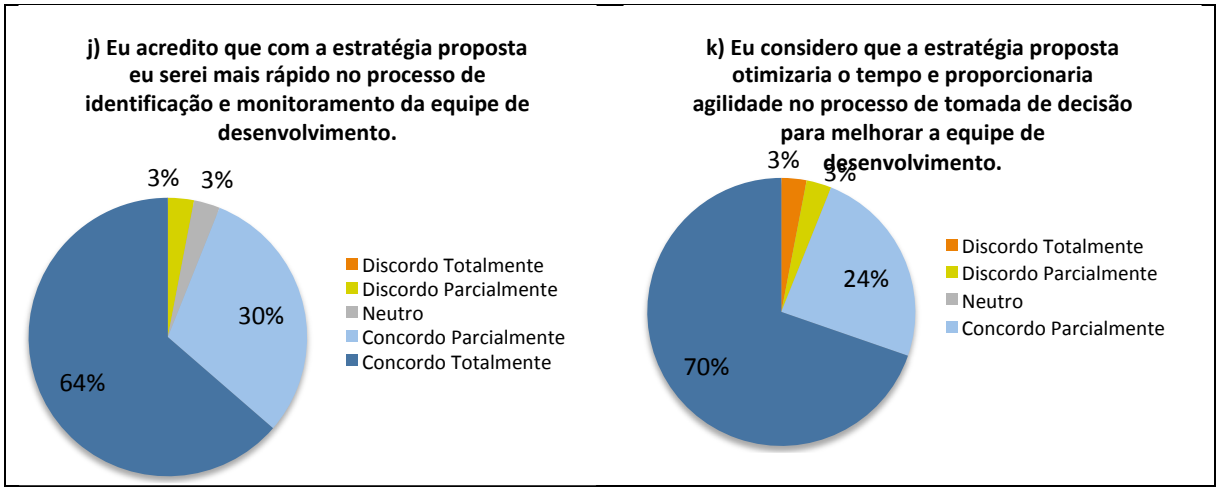
Figura 32 - Produtividade com a estratégia



A respeito da velocidade na utilização da estratégia, os participantes responderam a duas afirmações (apresentadas na seção 5.4), marcando a opção que melhor representava sua opinião conforme a seguinte escala: (1) Concordo totalmente; (2) Concordo parcialmente; (3) Neutro; (4) Discordo parcialmente; e (5) Discordo totalmente.

As respostas estão apresentadas na representação gráfica da Figura 33. Observa-se que na primeira afirmação, 94% dos participantes concordam com o ganho de velocidade ao utilizar a estratégia proposta e apenas um participante discordou parcialmente. Na afirmação seguinte, 94% dos participantes concordaram que a estratégia possibilita agilidade no processo de tomada de decisão para qualificar a equipe de desenvolvimento enquanto apenas dois participantes discordaram da afirmação.

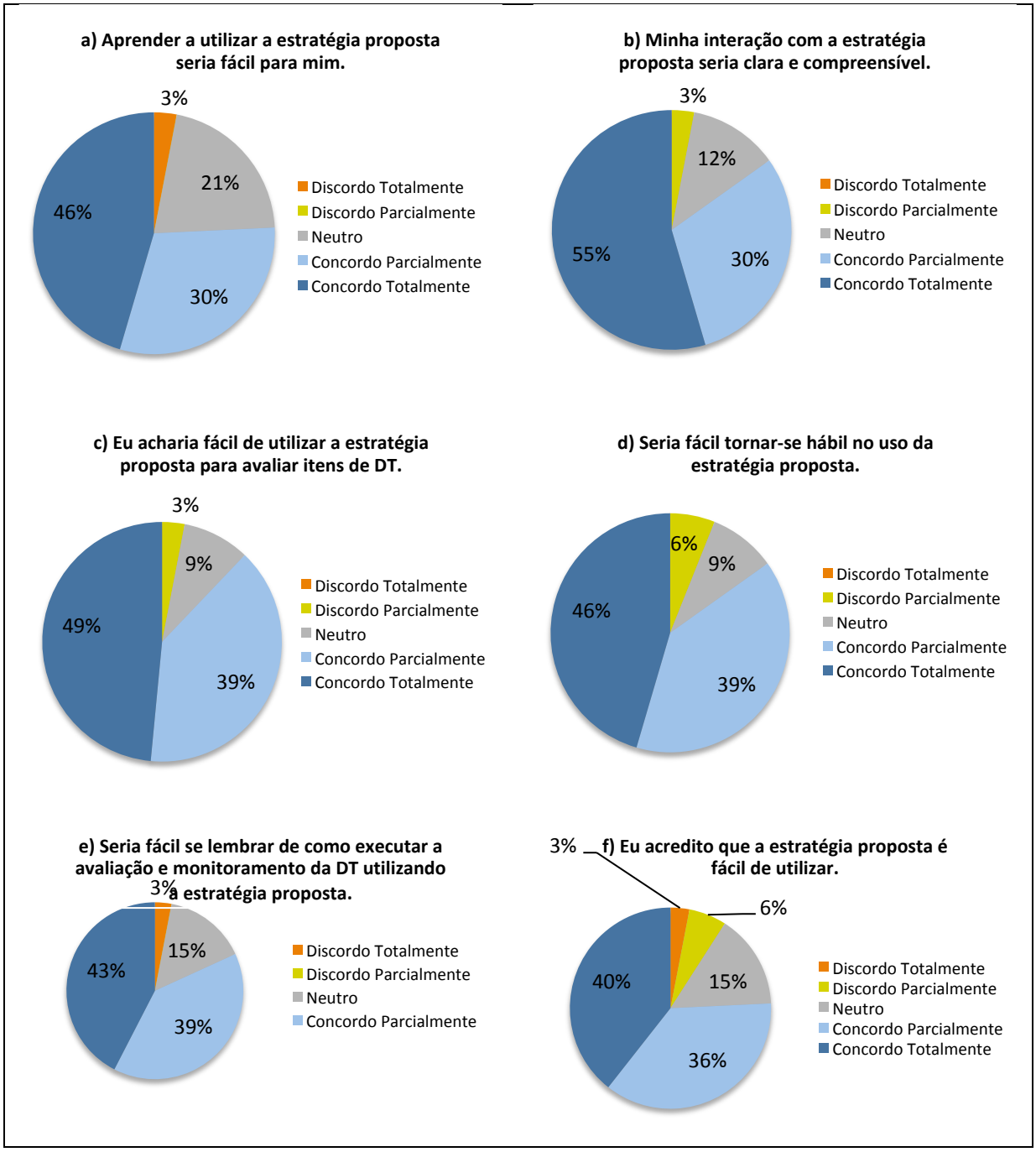
Figura 33 - Velocidade com a estratégia



5.5.2.2 Facilidade de uso

A respeito da facilidade de uso na utilização da estratégia, os participantes responderam a seis afirmações (apresentadas na seção 5.4), marcando a opção que melhor representava sua opinião conforme a seguinte escala: (1) Concordo totalmente; (2) Concordo parcialmente; (3) Neutro; (4) Discordo parcialmente; e (5) Discordo totalmente. Os resultados estão disponíveis na representação gráfica da Figura 34.

Figura 34 - Facilidade de uso com a estratégia



Na afirmação **a) Aprender a utilizar a estratégia proposta seria fácil para mim**, 76% dos participantes concordaram que seria fácil aprender a utilizar a estratégia proposta enquanto 21% se demonstraram como neutro e apenas um participante discordou parcialmente. Em seguida, na afirmação **b) Minha interação com a estratégia proposta seria clara e compreensível**, 85% dos participantes concordaram que a interação é clara e compreensível, 12% se manifestaram como neutro e apenas um participante discordou parcialmente.

Na afirmação **c) Eu acharia fácil de utilizar a estratégia proposta para avaliar itens de DT**, 88% dos participantes indicaram que a estratégia é fácil para avaliar itens de DT enquanto que 9% se apresentaram como neutro e apenas um participante discordou parcialmente. Em relação à afirmação **d) Seria fácil tornar-se hábil no uso da estratégia proposta**, 85% dos participantes afirmaram que seria fácil tornar-se hábil, 9% se apresentaram como neutro e apenas dois participantes discordaram parcialmente.

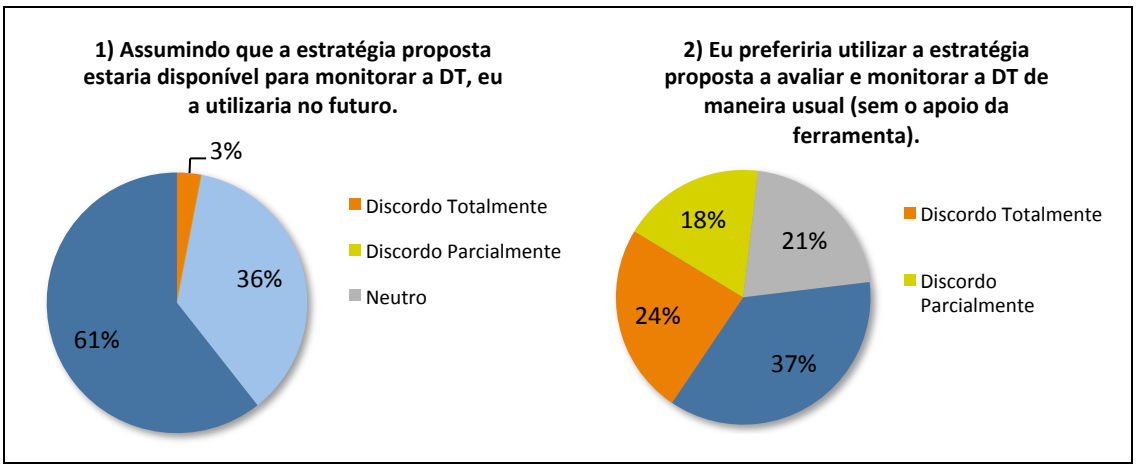
Na afirmação **e) Seria fácil se lembrar de como executar a avaliação e monitoramento da DT utilizando a estratégia proposta**, 82% dos participantes concordaram que seria fácil lembrar da execução da avaliação e monitoramento da DT enquanto 15% se manifestaram como neutro e apenas um participante discordou parcialmente.

Na afirmação **f) Eu acredito que a estratégia proposta é fácil de utilizar**, 76% dos participantes concordaram com a facilidade de uso da estratégia, 15% se apresentaram como neutro e três participantes discordaram da afirmação.

5.5.2.3 Possível uso futuro

Os participantes responderam a duas afirmações a respeito do possível uso futuro da estratégia (apresentadas na seção 5.4), marcando a opção que melhor representava sua opinião conforme a seguinte escala: (1) Concordo totalmente; (2) Concordo parcialmente; (3) Neutro; (4) Discordo parcialmente; e (5) Discordo totalmente. Os resultados estão disponíveis na Figura 35.

Figura 35 - Experiência dos participantes



De acordo com as respostas obtidas, na primeira afirmação, 97% dos participantes indicaram que utilizariam a estratégia proposta caso ela estivesse disponível. Na afirmação seguinte, 42% apontaram que preferiam realizar a atividade de análise sem o apoio da estratégia proposta. Embora os resultados indiquem que a estratégia tenha se apresentado como interessante, a justificativa deste número pode se dá pelo fato de que a estratégia era desconhecida pelos participantes e a compreensão em um contato inicial não tenha sido suficiente. Além disto, é preciso evoluir a estratégia para que possa ser utilizada posteriormente na indústria.

5.5.2.4 Pontos positivos e negativos

Os participantes do estudo também indicaram pontos positivos e negativos após a utilização da estratégia. Os pontos positivos indicados foram:

a) “Otimização do tempo/agilidade/produtividade” (18 participantes);
b) “Agilidade no processo de tomada de decisão para qualificação de equipe” (8 participantes);
c) “Gráficos e/ou tabelas ajudam na compreensão” (8 participantes);
d) “Usabilidade e boa interação homem-computador” (8 participantes);
e) “Resultados detalhados sobre colaboradores e DT” (7 participantes);
f) “Redução de erros futuros” (4 participantes);
g) “Qualificação da equipe de desenvolvimento” (3 participantes);
h) “Precisão” (2 participantes);
i) “Pontuar horas necessárias para corrigir problemas” (2 participantes);
j) “Classificação dos tipos DT por desenvolvedor” (1 participante);
k) “Motivação da equipe de desenvolvimento” (1 participante);

l) “Orientações sobre as práticas de desenvolvimento” (1 participante); e
m) “Planejamento de testes” (1 participante).

É possível identificar alguns benefícios esperados pela utilização da estratégia e que foram confirmados pelos participantes, tais como a otimização do tempo e agilidade na tomada de decisão para qualificação da equipe de desenvolvimento. Outros pontos que foram citados com maior frequência foram a facilidade de uso e possibilidade de redução de erros futuros.

Em contrapartida, os participantes indicaram os seguintes pontos negativos:

a) “Tempo para utilização” (3 participantes);
b) “Acompanhamento com o desenvolvedor para parametrizar a DT” (3 participantes);
c) “Interface em outros idiomas” (1 participante);
d) “Pré-requisitos” (1 participante);
e) “Desconforto ao estar sendo monitorado constantemente” (1 participante); e
f) “Dificuldade para atuar em equipe distante/remota” (1 participante).

O “tempo para utilização” pode-se referir ao fato de que os participantes ainda não conheciam a ferramenta e o tempo dedicado para realização do estudo não tenha sido suficiente para assimilar confortavelmente todo o conteúdo apresentado. Acredita-se que com um treinamento e outras oportunidades de utilização seja possível sanar esse ponto.

Outra questão apresentada foi o “acompanhamento com o desenvolvedor para parametrizar a DT”. Este item trata da necessidade de, junto ao desenvolvedor, obter informações sobre o tempo necessário para efetuar o pagamento do item de DT. O desenvolvedor responsável pelo surgimento da DT é o profissional que provavelmente tem mais conhecimento sobre esse trecho de código e que pode apontar informações importantes como se a DT foi intencional, além do tempo necessário para eliminá-la do projeto.

O ponto “pré-requisitos” trata das configurações necessárias para executar o projeto, apresentadas no capítulo 5. O ponto “desconforto ao estar sendo monitorado constantemente” refere-se a um possível sentimento pessoal de um membro da equipe de desenvolvimento que pode se sentir desconfortável em estar sendo monitorado.

Para finalizar, as seguintes sugestões foram apresentadas pelos participantes:

a) “Distribuição em outras línguas”;
b) “Teste com algumas perguntas para os desenvolvedores, construindo uma característica deles”;
c) “Inserir a quantidade de horas realmente necessárias para pagamento do item de dívida”;
d) “Poderia automatizar para o VISMINERTD enviar algum alerta para os que contribuem informando sobre os itens de dívida identificados”;
e) “Um "?" ao lado de cada informação explicando do que se trata cada tópico”;
f) “Acredito que o armazenamento das informações seria interessante para outras equipes usarem como parâmetro”; e
g) “Adicionar a quantidade de código escrito e a complexidade do código, pois nem todas as partes dos códigos são iguais”.

A sugestão de dar suporte a outros idiomas é uma atualização simples de ser realizada na ferramenta. A ferramenta foi disponibilizada em inglês para tentar reduzir barreiras em seu uso, uma vez que ela foi planejada como software livre e já se encontra disponível para a comunidade utilizar e contribuir com evoluções.

A sugestão sobre considerar características dos desenvolvedores pode trazer mais informações sobre os desenvolvedores. É uma informação complementar que pode ser útil na evolução do software. O item sobre a quantidade total de horas sugere a utilização de métricas como ponto de função. É importante lembrar que a ferramenta permite anotações individuais para cada item de DT, sendo assim possível manter este tipo de informação e consulta-la em momentos posteriores.

Sobre a sugestão de enviar alertas automáticos, é algo possível de ser implementado em outras versões, mas deve ser conduzido com cautela. Além de aumentar os pré-requisitos (como serviço de envio de e-mail e configuração do mesmo), é importante que haja um responsável em gerenciar a comunicação e que este tenha o controle sobre a sua distribuição.

Sobre as duas últimas sugestões, as funcionalidades já estão disponíveis na versão apresentada da estratégia. Todas as interações são persistidas e, ao utilizar o sistema em outro momento, as informações são restauradas.

5.6 DISCUSSÃO

A realização deste estudo forneceu indícios sobre o uso da estratégia proposta e seus resultados apontam que o VisminerTD pode ser útil para apoiar as atividades de monitoramento da dívida técnica (97% dos participantes indicaram que fariam uso da estratégia).

Os resultados obtidos apontam também que alguns dos benefícios esperados foram atingidos, uma vez que mais de 88% dos participantes do estudo estão de acordo que a estratégia proposta é útil, fácil de utilizar e obtém-se maior produtividade e agilidade no processo de monitoramento dos desenvolvedores, tornando possível a tomada de decisão no processo de aperfeiçoamento da equipe.

Apesar do VisminerTD ter se mostrado um apoio ferramental útil e interessante para obter informações sobre os desenvolvedores, os resultados do estudo também indicaram que a estratégia ainda não está preparada para ser utilizada na indústria. Esse é um resultado esperado, uma vez que a tecnologia está em seus estágios iniciais de desenvolvimento.

5.7 AMEAÇAS À VALIDADE

Os participantes do estudo foram escolhidos por conveniência e são alunos de graduação de cursos na área de Tecnologia da Informação. Entretanto, parte deles possui experiência em desenvolvimento de software na indústria, diminuindo tal ameaça. Desta forma, assumindo ainda que os resultados não sejam generalizáveis, eles fornecem indícios iniciais válidos sobre o objetivo do estudo.

Além disto, os itens de DT utilizados para análise no estudo, indicados na seção 5.3, fazem parte de uma amostra pequena em relação à quantidade real. Esta ameaça foi assumida para viabilizar a aplicação deste estudo, em um cenário onde não havia muitas horas disponíveis com os participantes. Entretanto, os autores do trabalho acreditam que quanto mais itens de DT forem analisados, a estratégia terá mais utilidade quando comparada à análise manual.

5.8 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado um estudo da estratégia proposta neste trabalho com o objetivo de avaliar sua viabilidade de uso. A avaliação foi realizada através do planejamento e execução de um estudo de caso envolvendo 33 participantes.

Os resultados obtidos indicaram que a estratégia apresentada é capaz de apoiar o monitoramento de como os membros da equipe de desenvolvimento contribuem com o surgimento da DT, no contexto de projetos de desenvolvimento de software, proporcionando maior utilidade, facilidade de uso, agilidade e produtividade no processo de tomada de decisão para proporcionar a melhoria da equipe de desenvolvimento. Além disto, foi possível obter informações sobre pontos negativos e sugestões de melhorias.

O próximo capítulo apresenta as considerações finais deste trabalho, descrevendo as contribuições, limitações e possíveis trabalhos futuros.

Capítulo 6 – Considerações finais

Este capítulo apresenta as considerações finais da pesquisa, suas contribuições, limitações e trabalhos futuros a serem realizados para a continuidade deste trabalho.

6.1 CONSIDERAÇÕES FINAIS

Dívida técnica descreve os custos que podem existir ao se optar por um caminho mais rápido e fácil no desenvolvimento e evolução do software, no intuito de obter tempo em troca do emprego de boas práticas de construção de software. Em ambientes de desenvolvimento de software, é comum a escolha desses atalhos para obedecer a prazos geralmente curtos.

Neste contexto, este trabalho abordou a importância do monitoramento da DT sob a perspectiva do entendimento sobre como os membros do time contribuem para a ocorrência ou eliminação de itens de DT. Foi proposta uma estratégia para qualificar os desenvolvedores de acordo com seu histórico de colaboração no projeto. Em complemento, foi planejado e desenvolvido o software VisminerTD para apoiar o uso estratégia proposta.

Para fundamentar a construção da estratégia, foi realizada uma revisão na literatura para identificar conceitos sobre dívida técnica (Capítulo 2) e os desafios sobre gerenciamento de equipes de desenvolvimento (Capítulo 3). A partir destas informações, foram conduzidas duas atividades, uma para identificar critérios para a construção da estratégia (Capítulo 4) e outra para avaliar a possibilidade de aplicação da estratégia proposta (Capítulo 5).

6.2 RESULTADOS E CONTRIBUIÇÕES OBTIDAS

Com este trabalho, os seguintes resultados e contribuições foram obtidos:

1. **Identificação de indicadores que podem apoiar a tomada de decisão em relação à qualificação da equipe de desenvolvimento:** através de busca na literatura técnica, foram identificados indicadores sobre DT presentes em código fonte e contribuição de desenvolvedores (Capítulos 2 e 3).
2. **Desenvolvimento de uma estratégia para apoiar a qualificação da equipe de desenvolvimento:** através de indicadores obtidos no item anterior, foi possível estabelecer uma estratégia que possibilita a obtenção de dados que fundamenta o monitoramento da DT sob a perspectiva do entendimento sobre

como os membros do time contribuem para a ocorrência ou eliminação de itens de DT;

3. **Ferramenta computacional para apoiar a utilização da estratégia:** foi desenvolvido o VisminerTD, que possibilita o monitoramento da evolução da DT;
4. **Avaliação da estratégia:** um estudo de caso foi planejado e executado com o objetivo de avaliar a viabilidade de uso da abordagem proposta.

6.3 LIMITAÇÕES

As principais limitações deste trabalho estão relacionadas às atividades executadas e apresentadas nos Capítulos 2, 3 e 4:

1. **Utilização do Mapeamento Sistemático:** embora tenha-se utilizado o mapeamento sistemático realizado por Alves *et al.* (2016) na busca de trabalhos que tratassem do gerenciamento de DT, não é possível garantir que todos os estudos primários relevantes tenham sido considerados;
2. **Perfil dos participantes do estudo:** os participantes do estudo foram escolhidos por conveniência, sendo composto por discentes da área de engenharia de software. Alguns indicaram não possuir conhecimento na área, mas, mesmo que os resultados não possam ser generalizados, eles revelam indicadores iniciais válidos sobre as questões analisadas.
3. **Estudo de caso:** o projeto real apresenta 1.442 *commits* relacionados ao total de 105 desenvolvedores. Porém, com o objetivo de viabilizar o planejamento e execução do estudo, foi utilizada uma amostra contendo 20 *commits* envolvendo quatro desenvolvedores. Esta quantidade pode ser considerada como pequena. Para lidar com esta limitação, replicações do experimento podem ser executadas utilizando uma amostra maior para obtenção de resultados mais conclusivos.

6.4 TRABALHOS FUTUROS

Ainda há muito a ser pesquisado e explorado em relação a DT e desenvolvedores. Considerando este trabalho como ponto de partida, algumas perspectivas futuras de trabalhos são:

a) Replicar o estudo de caso apresentado no Capítulo 5 para avaliar, na indústria, a percepção dos usuários quanto à utilização da estratégia proposta;
b) Em relação ao VisminerTD, pretende-se planejar e executar as seguintes melhorias sugeridas pelos participantes:
    - Adicionar opções de ajuda na interface para exibir informações;
    - Em cada item de DT, informar a quantidade de linhas adicionadas e/ou removidas;
    - Disponibilizar em outros idiomas, além do inglês.
c) Sobre o Visminer, pretende-se evoluir a ferramenta para atender a mais funcionalidades sobre indicadores.

d) Investigar como as informações disponibilizadas pela estratégia proposta podem ser úteis no sentido de fundamentar tomadas de decisão referentes à qualificação da equipe de desenvolvimento.

Referências

ALVES, N. S.R. et al. Identification and management of technical debt: A systematic mapping study. **Information and Software Technology**, v. 70, p. 100-121, 2016.

AMANATIDIS, T. et al. Who is producing more technical debt?: a personalized assessment of TD principal. In: INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT (MTD' 17), 9., 2017. **Proceedings…** 2017. p. 4.

CATALDO, M.; HERBSLEB, J. D. Coordination breakdowns and their impact on development productivity and software failures. **IEEE Transactions on Software Engineering**, v. 39, n. 3, p. 343-360, 2013.

CUNNINGHAM, W. The WyCash portfolio management system. In: ADDENDUM TO THE PROCEEDINGS ON OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES, AND APPLICATIONS, 1993. **Proceedings…** 1993. p. 29-30.

CURTIS, B.; SAPPIDI, J.; SZYNKARSKI, A. Estimating the size, cost, and types of technical debt. In: INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT, 3., 2012. **Proceedings…** 2012. p. 49-53.

DAVIS, F. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS Quarterly,** v.13, n.3, p. 319–340, 1989.

DEMARCO, T. **The deadline:** a novel about project management. [S.l.]: [s.n.], 1997.

FINDBUGS. [Homepage]. 2017. Disponível em <http://findbugs.soureforge.net>. Acesso em: 19 jan. 2017.

FOWLER, M. **TechnicalDebt.** 2004. Disponível em: <http://www.martinfowler.com/bliki/TechnicalDebt.html> Acesso em: 19 jan. 2017.

FOWLER, M. **Technical debt quadrant.** 2009. Disponível em: <http://www.martinfowler.com/bliki/TechnicalDebtQuadrant.html>. Acesso em: 19 jan. 2017.

FRANÇA, A. et al. A qualitative research on software projects team building. In: CONTECSI INTERNATIONAL CONFERENCE ON TECHNOLOGY AND INFORMATION SYSTEMS, 5., 2008, São Paulo. **Anais…** 2008.

GITHUB. [Homepage]. 2017. Disponível em: <https://github.com/search>, <https://github.com/search/advanced> e <https://help.github.com/articles/fork-a-repo> Acesso em: 19 jan. 2017.

GOUSIOS, G.; KALLIAMVAKOU, E.; SPINELLIS, D. Measuring developer contribution from software repository data. In: MSR ’08: INTERNATIONAL WORKING CONFERENCE ON MINING SOFTWARE REPOSITORIES, 2008. **Proceedings…** 2008. p. 129–132.

HOVEMEYER, D.; PUGH, W. Finding bugs is easy. In: ACM SIGPLAN Notices, 2004. pp. 92–106.

KALLIAMVAKOU, E.; GOUSIOS, G.; BLINCOE, K. The Promises and perils of mining github. In: 1 WORKING CONFERENCE ON MINING SOFTWARE REPOSITORIES, 11., 2014. **Proceedings…** 2014. p. 92-101.

KHATUN, A.; SAKIBY, K. A bug assignment technique based on bug fixing expertise and source commit recency of developers. In: COMPUTER AND INFORMATION TECHNOLOGY (ICCIT), 2016 INTERNATIONAL CONFERENCE ON, 19., 2016. **Proceedings…** 2016. p. 592-597

KRUCHTEN, P.; NORD, R.; OZKAYA, I. Technical debt: From metaphor to theory and practice. **Ieee software**, v. 29, n. 6, p. 18-21, 2012.

LANZA, M.; MARINESCU, R. **Object-oriented metrics in practice:** using software metrics to characterize, evaluate, and improve the design of object-oriented systems.[S.l.]: Springer Science & Business Media, 2006.

LATOZA T. D.; VENOLIA G.; DELINE R., Maintaining mental models: a study of developer work habits. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 28., 2006. **Proceedings...** 2006. p. 492–501.

LETOUZEY, J.; ILKIEWICZ, M. Managing technical debt with the SQALE method. **IEEE Software**, v. 29, n. 6, p. 44-51, 2012.

LETOUZEY, J.-L. The SQALE method for evaluating Technical Debt. In: INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT (MTD), 3., 2012. **Proceedings...** 2012.

LIM, E.; TAKSANDE, N.; SEAMAN, C. A balancing act: What software practitioners have to say about technical debt. **IEEE software**, v. 29, n. 6, p. 22-27, 2012.

MCCONNELL, S. Technical Debt - 10x Software Development. 2007. Disponível em: <http:// http://www.construx.com/10x_Software_Development/Technical_Debt> Acesso em: 19 jan. 2017.

NUGROHO, A.; VISSER, J.; KUIPERS, T. An empirical model of technical debt and interest. In: WORKSHOP ON MANAGING TECHNICAL DEBT, 2., 2011. **Proceedings…** 2011. p. 1-8.

OLSINA, L. et al. Specifying quality characteristics and attributes for websites. In: WEB ENGINEERING: managing diversity & complexity of web application development. Heidelberg: Springer, 2008. p. 266-278.

OSTRAND T. J.; WEYUKER E. J.; BELL R. M. Programmer-based fault prediction. In: INTERNATIONAL CONFERENCE ON PREDICTIVE MODELS IN SOFTWARE ENGINEERING, 6., 2010. **Proceedings…** 2010. p. 19.

PMD. [Homepage]. 2017. Disponível em: <https://pmd.github.io> Acesso em: 19 jan. 2017.

POWER, K. Understanding the impact of technical debt on the capacity and velocity of teams and organizations: Viewing team and organization capacity as a portfolio of real options. In: MANAGING TECHNICAL DEBT (MTD), 2013 INTERNATIONAL WORKSHOP ON, 4., 2013. **Proceedings…** 2013. p. 28-31.

QIU, Y. et al. An empirical study of developer quality. In: INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY, RELIABILITY AND SECURITY - COMPANION (QRS-C), 2015, Vancouver, BC, Canada. **Proceedings…** 2015. p. 202-9.

SARDINHA, F.; COSTA, C. Training and interface features in technology acceptance. In: WORKSHOP ON OPEN SOURCE AND DESIGN OF COMMUNICATION, 2011. **Proceedings…** 2011.p. 55–60.

SEAMAN C.; GUO, Y. Measuring and monitoring technical debt, advances. **Computers,** v. 82, 2011.

SILLITTI, A. et al. Collecting, integrating and analyzing software metrics and personal software process data. In: EUROMICRO CONFERENCE, 29., 2003. **Proceedings...** 2003. p. 336-342.

SONARQUBE. [Homepage]. 2017. Disponível em: <http://www.sonarqube.org>. Acesso em: 19 jan. 2017.

TSAY, J.; DABBISH, L.; HERBSLEB, J. Influence of social and technical factors for evaluating contribution in github. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING - ICSE, 36., 2014. **Proceedings…** 2014.

VASILESCU, B. et al. Gender and tenure diversity in GitHub teams. In: ANNUAL ACM CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 33., 2015. **Proceedings**... 2015. p.3789-3798.

WAGSTROM, P. A. Vertical communication in open software engineering communities. 2009 (Dissertation)- Carnegie Institute of Technology and School of Computer Science. Pittsburgh, 2009.

WU, Y. et al. The influence of developer quality on software fault-proneness prediction. In: INTERNATIONAL CONFERENCE ON, 8., 2014. **Proceedings**... 2014. pp. 11-19.

ZHOU, M.; MOCKUS, A. Mining micro-practices from operational data. In: ACM SIGSOFT INTERNATIONAL SYMPOSIUM ON FOUNDATIONS OF SOFTWARE ENGINEERING, 22., 2014. **Proceedings...** 2014. p. 845-848.

Apêndice A - Instrumentos utilizados no estudo apresentado no Capítulo 5

A.1 Formulário do Termo de Consentimento

TDresearchteam
Technical Debt Research Team

Estudo da aplicação da estratégia de monitoramento da contribuição de equipes de desenvolvimento na evolução DT.

Termo de Consentimento

As informações contidas neste formulário visam afirmar acordo por escrito, mediante o qual o profissional autoriza sua participação no experimento, com pleno conhecimento da natureza dos procedimentos a que se submeterá para ser participante do estudo e com capacidade de livre arbítrio e sem qualquer coação.

Esta participação é voluntária e o sujeito deste experimento tem a liberdade para retirar seu consentimento a qualquer momento e deixar de participar do estudo, sem qualquer prejuízo ao atendimento a que está sendo ou será submetido.

I. TEMA

Dívida Técnica.

II. OBJETIVOS DO ESTUDO

Analisar a estratégia de monitoramento de itens de DT proposta, **com o objetivo de** caracterizar **com respeito a** sua contribuição em relação à qualificação da equipe de desenvolvimento, **sob o ponto de vista de** estudantes de graduação cursando a disciplina de engenharia de software **no contexto de** projetos de desenvolvimento de software.

III. INSTITUIÇÃO RESPONSÁVEL

Universidade Salvador - UNIFACS.

IV. PESQUISADOR

Sandro Luiz Abreu Campos dos Santos, aluno do curso Mestrado Acadêmico em Sistemas e Computação, orientado pelo Prof. Dr. Rodrigo Oliveira Spínola, na instituição Universidade Salvador - UNIFACS.

V. CONFIDENCIALIDADE

Toda informação coletada neste estudo é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a

manter sigilo das tarefas solicitadas e dos documentos apresentados e que fazem parte dos experimentos.

VI. CONSENTIMENTO

Eu, _____,
certifico que, tendo lido as informações acima e suficientemente esclarecido de todos os itens, estou plenamente de acordo com a realização do experimento. Assim, eu autorizo a execução do trabalho de pesquisa exposto acima.

Assinatura do Participante:

_____.

A.2 Formulário de caracterização do participante

Estudo da aplicação da estratégia de monitoramento da contribuição de equipes de desenvolvimento na evolucão DT.

Caracterização do Participante

Nome:_____ Nível: _____

Experiência com desenvolvimento de software

1) **Qual a sua experiência com desenvolvimento de software? (Marque os itens que melhor se aplicam):**

( ) Nunca desenvolvi software.
( ) Desenvolvi software apenas para uso próprio.
( ) Desenvolvi software como parte de uma equipe, relacionado a um curso.
( ) Desenvolvi software como parte de uma equipe, na indústria.

2) **Qual o número de anos de experiência relevante em desenvolvimento de software (Ex.: "Eu trabalhei por 10 anos como gerente de projeto de software na indústria").**

_____
_____
_____

3) **Informe o grau de sua experiência sobre desenvolvimento de software seguindo a escala de 5 pontos abaixo:**

(1) Nenhum
(2) Estudado em aula
(3) Praticado em projetos em sala de aula
(4) Utilizado em projetos pessoais
(5) Utilizado em projetos na indústria

Experiência com gerenciamento de projetos de software?	1	2	3	4	5
Experiência com acompanhamento e correção de defeitos do software?	1	2	3	4	5
Experiência com manutenção de software?	1	2	3	4	5
Experiência com arquitetura de software?	1	2	3	4	5
Experiência com projeto de software?	1	2	3	4	5
Experiência com documentação de software?	1	2	3	4	5
Experiência com especificação de requisitos de software?	1	2	3	4	5
Experiência com codificação?	1	2	3	4	5
Experiência com testes de software?	1	2	3	4	5

Conhecimento sobre Dívida Técnica

4) **Classifique o seu conhecimento sobre Dívida Técnica:**
   ( ) Excelente
   ( ) Bom
   ( ) Baixo
   ( ) Não tenho conhecimento

A.3 Formulário de avaliação da estratégia proposta

TDresearchteam
Technical Debt Research Team

Estudo da aplicação da estratégia de monitoramento da contribuição de equipes de desenvolvimento na evolução da DT.

Formulário de avaliação

Instrução: Esta é a terceira etapa do estudo de viabilidade da aplicação da estratégia de monitoramento da contribuição de desenvolvedores no surgimento da dívida técnica. O objetivo da atividade a ser realizada é coletar suas percepções e considerações sobre a execução da atividade de monitoramento da DT com a utilização do VisminerTD, a partir do preenchimento das questões listadas abaixo.

1) Em relação à **utilidade** da estratégia de monitoramento da DT com o apoio ferramental VisminerTD, para cada item marque a opção que melhor representa seu ponto de vista:

(1) Concordo Totalmente	(2) Concordo Parcialmente	(3) Neutro	(4) Discordo Parcialmente	(5) Discordo Totalmente

Item	Resposta				
	1	2	3	4	5
Usando a estratégia proposta, eu seria capaz de identificar e avaliar itens de DT mais rapidamente.					
Usando a estratégia proposta, eu iria melhorar o meu desempenho na avaliação de itens de DT.					
Usando a estratégia proposta, eu iria melhorar a minha eficácia em avaliar itens de DT.					
Usando a estratégia proposta, eu seria capaz de identificar quais são todos os desenvolvedores que trabalharam na evolução do software.					
Usando a estratégia proposta, eu seria capaz de identificar quais são os desenvolvedores que mais contribuem com o surgimento da DT.					
Usando a estratégia proposta, seria mais fácil de decidir quais são os itens de DT mais relevantes para pagamento.					
Eu acredito que a estratégia proposta seria útil para a tomada de decisão na qualificação da equipe de desenvolvimento.					

2) Em relação à **facilidade de uso** da estratégia de monitoramento da DT com o apoio ferramental VisminerTD, para cada item marque a opção que melhor representa o seu ponto de vista:

(1) Concordo Totalmente	(2) Concordo Parcialmente	(3) Neutro	(4) Discordo Parcialmente	(5) Discordo Totalmente

Item	Resposta				
	1	2	3	4	5
Aprender a utilizar a estratégia proposta seria fácil para mim.					
Minha interação com a estratégia proposta seria clara e compreensível.					
Eu acharia fácil de utilizar a estratégia proposta para avaliar itens de DT.					
Seria fácil tornar-se hábil no uso da estratégia proposta.					
Seria fácil se lembrar de como executar a avaliação e monitoramento da DT utilizando a estratégia proposta.					
Eu acredito que a estratégia proposta é fácil de utilizar.					

3) Em relação à **produtividade** da estratégia de monitoramento da DT com o apoio ferramental VisminerTD, para cada item marque a opção que melhor representa o seu ponto de vista:

(1) Concordo Totalmente	(2) Concordo Parcialmente	(3) Neutro	(4) Discordo Parcialmente	(5) Discordo Totalmente

Item	Resposta				
	1	2	3	4	5
Eu acredito que através de dados obtidos com a estratégia proposta eu poderei capacitar melhor a equipe de desenvolvimento para torná-la mais produtiva.					
Seria mais fácil identificar a necessidade de efetuar uma rotatividade de pessoal para tornar a equipe mais produtiva com a estratégia proposta.					

4) Em relação a **velocidade** da estratégia de monitoramento da DT, com o VisminerTD, para cada item marque a opção que melhor representa o seu ponto de vista:

(1) Concordo Totalmente	(2) Concordo Parcialmente	(3) Neutro	(4) Discordo Parcialmente	(5) Discordo Totalmente

Item	Resposta				
	1	2	3	4	5
Eu acredito que com a estratégia proposta eu serei mais rápido no processo de identificação e monitoramento da equipe de desenvolvimento.					
Eu considero que a estratégia proposta otimizaria o tempo e proporcionaria agilidade no processo de tomada de decisão para melhorar a equipe de desenvolvimento.					

5) Em relação a um **possível uso futuro** da estratégia de monitoramento da DT, com o VisminerTD, para cada item marque a opção que melhor representa o seu ponto de vista:

(1) Concordo Totalmente	(2) Concordo Parcialmente	(3) Neutro	(4) Discordo Parcialmente	(5) Discordo Totalmente

Item	Resposta				
	1	2	3	4	5
Assumindo que a estratégia proposta estaria disponível para monitorar a DT, eu a utilizaria no futuro.					
Eu preferiria utilizar a estratégia proposta a avaliar e monitorar a DT de maneira usual (sem o apoio da ferramenta).					

6) De acordo com sua opinião, liste os aspectos positivos da utilização da estratégia de monitoramento de equipe do VisminerTD:

7) De acordo com sua opinião, liste os aspectos negativos da utilização da estratégia de monitoramento de equipe do VisminerTD:

8) Você possui alguma sugestão para melhoria da estratégia de monitoramento de equipe com o VisminerTD? Em caso de positivo, por favor, especifique-a:

( ) Sim ( ) Não
