TDVision: Um Módulo Computacional para Visualização de Dívidas Técnicas

Irvayne Ibiapina[1], **Ronivon Silva Dias**[1], **Otávio Cury Castro**[1], **Francisco Vanderson de Moura Alves**[1], **Pedro de Alcântara dos Santos Neto**[1]

[1]Universidade Federal do Piauí - UFPI - Brasil

irvaynematheus@gmail.com, ronivon@ufpi.edu.br, otaviocury.oc@gmail.com

vanderson.moura.vm@gmail.com, pasn@ufpi.edu.br

Abstract. *Quality is an essential feature of software. However, the quality of a software usually decreases during development. This occurs most often by actions taken by the development team that bring short-term benefits, but compromise the quality and evolution of long-term software. This phenomenon is known in the literature as Technical Debt. Because of this, in this paper presents a module called TDVision to identify and visualize technical debt present in software, in order to support management and quality inspection activities.*

Resumo. *A qualidade é uma característica imprescindível de um software. Entretanto, a qualidade de um software normalmente diminui durante seu desenvolvimento. Isso ocorre na maioria das vezes por atitudes tomadas pela equipe de desenvolvimento que trazem benefícios a curto prazo, mas comprometem a qualidade e evolução do software a longo prazo. Esse fenômeno é conhecido na literatura como Dívida Técnica. Por conta disso, neste trabalho é apresentado um módulo chamado TDVision para identificar e visualizar dívidas técnicas presentes em um software, de forma a apoiar atividades de gerenciamento e inspeção de qualidade.*

1. Introdução

A Engenharia de Software é uma área da computação que pode ser compreendida como a aplicação de uma abordagem sistemática e disciplinada que tem como objetivo a construção de um produto de software de qualidade [IEEE 1990]. Entretanto, a qualidade de um software normalmente diminui durante seu desenvolvimento em diversos aspectos que podem comprometer sua evolução e seu correto funcionamento [Parnas 1994].

Um fator que contribui para a diminuição da qualidade do software é conhecido como Dívida Técnica (DT) [Cunningham 1992]. A DT é uma alusão à dívida financeira e é ocasionada por escolhas que trazem benefícios a curto prazo, como maior agilidade e menor esforço, porém tornam-se mais custosas a longo prazo [Kruchten et al. 2012]. Os efeitos negativos da DT na evolução do software formam um ambiente favorável ao surgimento de defeitos, comprometem a qualidade e reduzem a manutenibilidade do código [Spínola et al. 1992]. Devido a esses efeitos, é crescente a atenção para o gerenciamento de DTs tanto na comunidade de desenvolvimento de software, como na comunidade científica.

Por conta disso, neste trabalho é apresentado um módulo computacional intitulado TDVision para monitorar DTs em projetos de software. Esse modulo possibilita

que gestores de projetos compreendam a situação do projeto no contexto de DTs, e assim, gerenciem essas pendências técnicas, e consequentemente contornem os problemas advindos da presença de DTs em projetos de software.

Este trabalho está organizado da seguinte forma: a Seção 2 apresenta conceitos relacionados a Dívida Técnica; a Seção 3 apresenta alguns estudos relacionados ao tema; na Seção 4 é apresentado o módulo que implementa a abordagem proposta neste trabalho; na Seção 5 é apresentado um estudo de viabilidade do TDVision em um projeto real de grande porte; Por fim, a Seção 6 apresenta as conclusões e trabalhos futuros.

2. Dívida Técnica

Dívida Técnica (DT) representa uma metáfora que se refere às consequências do desenvolvimento de software deficiente, imaturo ou impróprio [Cunningham 1992]. Alguns autores caracterizam as DTs como sendo qualquer parte do software atual que é considerado sub ótima de uma perspectiva técnica [E. Tom and Vidgen 2013].

Em seu trabalho, [Alves et al. 2016] sintetiza algumas dimensões da Dívida Técnica, categorizando-as em 15 tipos: Arquitetura, Construção, Projeto, Código, Defeito, Teste, Documentação, Infraestrutura, Pessoas, Processos, Requisitos, Serviço, Automação de Teste, Usabilidade e Versionamento. Essas DTs podem ser caracterizados por indicadores identificados por meio de cálculo de métricas de código, detecção de *code smells* e/ou análise de comentários.

Dentre as dívidas que podem surgir no código fonte do projeto, podemos destacar a Dívida de Código. Neste trabalho, a caracterização de Dívidas de Código é baseada no conjunto de 18 métricas de código e 7 *code smells*, todas fundamentadas em trabalhos presentes na literatura [Lanza and Marinescu 2007]. Além desses 7 tipos de *code smells*, este trabalho considera marcações no código com as *tags* TODO e FIXME como indicativos de DTs em comentários. Na Tabela 1 são apresentados os *code smells* considerados neste trabalho e sua respectiva descrição.

Tabela 1. Tipos de *Code Smell* considerados neste trabalho.

Code Smell	Descrição
God Class	Caracterizado por classes que centralizam a inteligência do sistema.
Brain Method	Caracterizado por métodos que centralizam as funcionalidades de uma classe.
Brain Class	Caracterizado por classes que acumulam uma quantidade excessiva de inteligência, geralmente possuindo vários Brain Method.
Data Class	Caracterizado por classes que apenas armazenam dados e não possuem funcionalidades complexas. Elas oferecem mais dados em vez de serviços.
Conditional Complexity	Caracterizado por métodos que possuem muitos comandos condicionais.
Long Method	Caracterizado por métodos que possuem muitas linhas de código.
Feature Envy	Caracterizado por métodos que acessam muitos dados de outras classes e pouca das suas.

3. Trabalhos Relacionados

Neste trabalho é apresentado um módulo intitulado TDVision para visualizar dívidas técnicas em projetos de software. O TDVision disponibiliza uma nova forma de visu-

alizar e gerenciar as pendências técnicas que possam vir a surgir em um software durante sua evolução. Entretanto, existem algumas ferramentas que auxiliam na identificação e monitoramento de DTs em projetos de software.

O *SonarQube* é uma das ferramentas mais conhecidas e utilizadas na indústria de software para realizar inspeção de qualidade de código [Campbell and Papapetrou 2013]. O objetivo dessa ferramenta é acompanhar a qualidade do código fonte. Ele é um software *open source* capaz de extrair diferentes tipos de informações sobre um sistema como: métricas de software, *code smells*, identificação de itens de dívida técnica, acoplamento entre as classes, problemas de vulnerabilidade e segurança, entre outros. O SonarQube pode ser encontrado no seguinte endereço: *http://www.sonarqube.org*.

Em [Mendes et al. 2015] é apresentada uma ferramenta, denominada *VisminerTD*, que permite a identificação automática e monitoramento da evolução de itens de dívida técnica em projetos de software. Ela utiliza mineração de repositórios de software para detectar 8 tipos de DT, e permitir a análise conjunta dos dados provenientes de um de métricas que analisam o código fonte e seus comentários, para apoiar a identificação. Para avaliar o *VisminerTD* foi realizado um estudo no projeto *JUnit* e foi possível obter indicativos iniciais sobre a viabilidade de uso da ferramenta proposta.

Este trabalho diferencia-se dos trabalhos citados anteriormente por propor um mecanismo que extrai dados de repositórios de código e gera informações acerca dos projetos em um ambiente Web sem a necessidade da integração com outros sistemas. Além disso, o TDVision exibe as informações de dívida técnica em uma menor granularidade, listando os problemas presentes em cada classe, e em seus respectivos métodos, facilitando o direcionamento na identificação de um determinado problema técnico.

4. Módulo TDVision

O TDVision é um módulo computacional que está incorporado a ferramenta CoDiVision [Moura et al. 2016] e tem como objetivo a visualização de pendências técnicas presentes no software. Com esse módulo, é possível identificar as DTs em classes e métodos de um projeto de software, e assim direcionar o usuário no momento da priorização correções das dívidas. O TDVision está disponível em um ambiente Web, e pode ser acessado por meio do seguinte endereço: `http://easii.ufpi.br/codivision/`.

O módulo TDVision utiliza-se da *API RepositoryMiner* [G et al. 2015] para identificar DTs que surgem no código fonte durante o desenvolvimento. Como apresentado na Seção 2, esses tipos de DTs presentes no código fonte são caracterizados pela existência de *code smells*. Portanto, é utilizado o recurso de extração e identificação de *code smells* disponível pela *API RepositoryMiner*.

Na Figura 1 é apresentada a arquitetura simplificada do TDVision. Ele utiliza de serviços existentes da CoDiVision para obter informações do projeto. Após isso, a *API RepositoryMiner* realiza o processo de identificação de *code smells*. Atualmente, são extraídos *code smells* de códigos desenvolvidos na linguagem Java.

5. Estudo de Viabilidade

Foi realizado um estudo de viabilidade para avaliar, sobre o ponto de vista operacional e técnico, a aplicabilidade do TDVision para o processo de desenvolvimento de software.

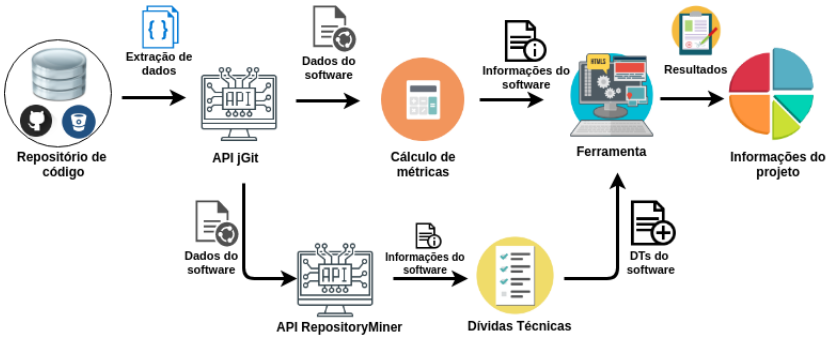

Figura 1. Arquitetura simplificada do TDVision.

Para isso, foi realizado a extração dos dados do Sistema Emissor de Relatório de Contas Anuais (SERCA). Esse software tem como objetivo facilitar na geração e gerenciamento de relatório das contas anuais do Tribunal de Contas do Estado do Piauí (TCE-PI).

Figura 2. Classes que possuem problemas de Dívida Técnica.

Na Figura 2 é apresentada as classes ordenadas de acordo com a criticidade dos arquivos em relação a presença de Dívidas Técnicas. Esse valor de criticidade é calculado utilizando um sistema de inferência *fuzzy* que considera características da classe, como o acoplamento, complexidade ciclomática e a quantidade de DTs. O objetivo desse mecanismo inteligente é facilitar a organização das classes, bem como apoiar no momento de priorizar uma determinada classe para pagamento de DTs.

Ao selecionar uma classe, é exibido a tela apresentada na Figura 3. Nela é mostrada os tipos de *code smells* identificados para a classe. Além disso, são apresentados os valores calculados nas métricas de código para a caracterização do determinado *code smell*. Também é apresentada o diretório da classe e se ele possui *code smells* de comentários.

Além das informações em uma determinada classe, o TDVision possibilita a visualização das pendências técnicas em uma menor granularidade, que são em métodos. Na Figura 4 é apresentada essa visualização. O usuário pode selecionar qual método ele

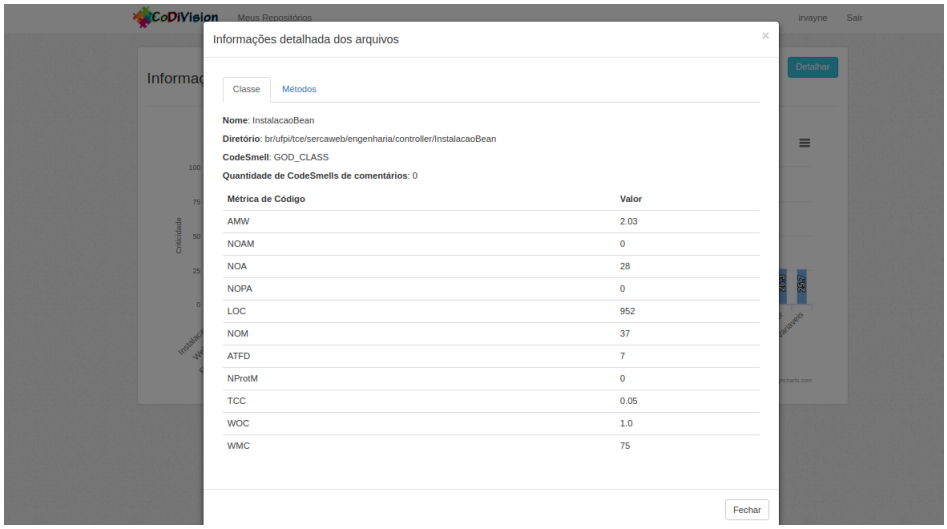

Figura 3. Dívidas Técnicas presentes na classe.

deseja visualizar as informações sobre DT, e então a TDVision disponibiliza os valores das métricas de código e *code smells*.

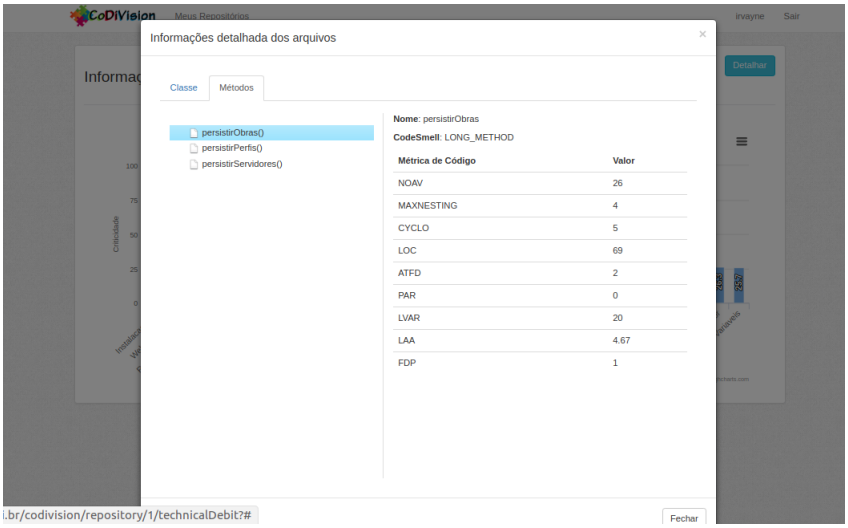

Figura 4. Dívidas Técnicas presentes no método.

6. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentado o TDVision, um módulo computacional para identificar e monitorar as dívidas técnicas presentes no código fonte de um software. Com o TDVision, gestores de projetos e desenvolvedores podem identificar quais pendências técnicas foram inseridas durante o desenvolvimento, e assim propor estratégias para mitigar os problemas advindos mau gerenciamento de DTs.

Foi realizado um estudo de viabilidade para verificar a aplicabilidade do TDVision em um contexto real. Foram extraídos dados de um projeto de grande porte, e com isso, foi possível identificar seus problemas de dívidas técnicas em nível de classe e de método.

Como trabalhos futuros pretendemos realizar um estudo de caso para avaliar em médio/longo prazo os efeitos da TDVision no ciclo de vida do software. Além disso, pretendemos expandir o suporte a extração de dívidas em outras linguagens e a possibilidade de diferentes formas de visualização das informações.

Referências

Alves, N. S., Mendes, T. S., de Mendonça, M. G., Spínola, R. O., Shull, F., and Seaman, C. (2016). Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, 70:100–121.

Campbell, G. and Papapetrou, P. P. (2013). *SonarQube in action*. Manning Publications Co.

Cunningham, W. (1992). The wycash portfolio management system. In *Addendum to the Proceedings on Object-oriented Programming Systems, Languages, and Applications (Addendum)*, OOPSLA '92, pages 29–30, New York, NY, USA. ACM.

E. Tom, A. A. and Vidgen, R. (2013). An exploration of technical debt. *Journal of Systems and Software*.

G, F., Novais, Mendes, T. S., Gonçalves, Renato Novais, R., Spınola, R. O., Mendonça, M., and Salvador, B. (2015). Repositoryminer- : uma ferramenta extensível de mineração de repositórios de software para identificação automática de dívidas técnicas.

IEEE (1990). Ieee standard glossary of software engineering terminology. *IEEE Standard 610.12*.

Kruchten, P., Nord, R. L., and Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *Ieee software*, 29(6):18–21.

Lanza, M. and Marinescu, R. (2007). *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. Springer Science & Business Media.

Mendes, T. S., Gonçalves, D. P., Gomes, F. G., Novais, R., Spınola, R. O., Mendonça, M., and Salvador, B. (2015). Visminertd: Uma ferramenta para identificaç ao automática e monitoramento interativo de dıvida técnica.

Moura, F., Lira, W., Ibiapina, I., and Neto, P. (2016). Codivision: Uma ferramenta para mapear a divisão do conhecimento entre os desenvolvedores a partir da análise de repositório de código. *Congresso Brasileiro de Software - Cbsoft*.

N. Brown, Y. Cai, e. a. (2010). Managing technical debt in software-reliant systems. *Proceedings of the FSE/SDP workshop on Future of software engineering research*.

Parnas, D. L. (1994). Software aging. In *Proceedings of the 16th International Conference on Software Engineering*, ICSE '94, pages 279–287, Los Alamitos, CA, USA. IEEE Computer Society Press.

Spínola, R., e. a. M. T. S., Gonçalves, D. P., and Gomes (1992). The wycash portfolio management system. *ACM SIGPLAN OOPS Messenger (Vol. 4, No. 2). ACM. pp. 29-30*.