Organização do Corpo de Conhecimento sobre Dívida Técnica: Tipos, Indicadores, Estratégias de Gerenciamento e Causas

Nicolli Souza Rios Alves[1,2] e Rodrigo Oliveira Spínola[2,3]

[1]Programa de Pós-Graduação em Ciência da Computação – Universidade Federal da Bahia (UFBA) – Salvador – BA – Brasil

[2]Programa de Pós-Graduação em Sistemas e Computação – Universidade Salvador (UNIFACS) – Salvador – BA – Brasil

[3]Centro de Projetos Fraunhofer para Engenharia de Software e Sistemas Salvador – BA – Brasil

nicollirioss@gmail.com, rodrigo.spinola@pro.unifacs.br

Abstract. *The identification and management of technical debt (TD) improve the software quality and the productivity of its development. However, before identifying or managing it, it is necessary to know its different types, the indicators of its presence, available techniques for its management, and the causes for its occurrence. The goal of this work is to organize a body of knowledge on TD considering this set of information. For this, we performed (i) a systematic mapping study of the literature that resulted in the analysis of 100 primary studies and (ii) an interview study that considered 30 units of analysis. The main results of this work are the organization of a body of knowledge on TD and its sharing through the TD Wiki infrastructure.*

Resumo. *A identificação e gerenciamento da dívida técnica (DT) resulta em maior qualidade e produtividade na construção do software. No entanto, antes de identificar ou gerenciar a dívida, é necessário conhecer seus diferentes tipos, os indicadores de sua presença, as técnicas para seu gerenciamento e as causas para a sua ocorrência. O objetivo deste trabalho é organizar um corpo de conhecimento sobre DT considerando esse conjunto de informações. Para isso, foram realizados um mapeamento sistemático da literatura que resultou na análise de 100 estudos primários e um estudo de entrevista que considerou 30 unidades de análise. Como resultado, o corpo de conhecimento foi organizado e compartilhado através da infraestrutura TD Wiki.*

1. Introdução

Durante o desenvolvimento, a qualidade do software diminui quando se considera aspectos como sua estrutura interna, adesão a normas, documentação e facilidade de entendimento para futuras manutenções [Lientz *et al.* 1978][Lehman e Belady 1985] [Parnas 1994]. Um motivo para isso acontecer é que as atividades de desenvolvimento são frequentemente realizadas sob forte restrição de tempo e recursos, sendo muitas vezes investida a quantidade mínima de esforço e tempo necessários para sua realização.

Esse cenário traz, por exemplo, impactos na produtividade uma vez que uma modificação em um software de baixa qualidade geralmente é mais difícil de ser realizada do que em um de alta qualidade [Lehman e Belady 1985].

Lidar com essa questão considerando o conceito de Dívida Técnica (DT) tem ajudado profissionais e pesquisadores a debaterem problemas associados ao desenvolvimento do software. DT descreve o efeito de artefatos imaturos no desenvolvimento do software, que traz um benefício a curto prazo para o projeto em termos de maior produtividade e menor esforço, mas que poderão ter de ser ajustados com juros (esforço extra necessário para ajustar o item da dívida ou realizar atividades de desenvolvimento em um software com sua qualidade interna degenerada) mais tarde [Seaman e Guo 2011][Kruchten *et al.* 2012]. A consequência desses problemas é observada em atrasos inesperados na realização de modificações necessárias e na dificuldade em atingir os critérios de qualidade acordados para o projeto [Spínola *et al.* 2013][Zazworka *et al.* 2013].

É comum que um projeto de software incorra em dívidas durante seu processo de desenvolvimento. No entanto, sua presença traz riscos para o projeto e dificulta sua gestão uma vez que os gerentes também terão que decidir se a dívida será paga e, em caso positivo, quanto dela deve ser paga e quando. A identificação, medição e gerenciamento da DT ajudaria os gestores a tomarem decisões fundamentadas, resultando em maior qualidade do software e maior produtividade na execução das atividades de desenvolvimento [Guo *et al.* 2014]. No entanto, antes de identificar, medir ou gerenciar a dívida, é necessário entender quais são os tipos de dívida que podem afetar os projetos, os indicadores de sua presença, as técnicas que têm sido propostas para o seu gerenciamento e o que leva equipes a incorrerem a dívida. A organização destas informações permitiria que equipes trabalhassem de forma mais controlada em atividades de prevenção, monitoramento e gerenciamento da dívida, além de estabelecer um arcabouço bem fundamentado sobre o qual novas pesquisas possam ser realizadas.

Este cenário motivou estes pesquisadores a organizarem um corpo de conhecimento sobre DT considerando em sua estrutura os seguintes itens: tipos de dívida, indicadores de sua presença, estratégias de gerenciamento e causas para a sua ocorrência. Assim, buscou-se responder as seguintes questões de pesquisa principais: **(RQ1)** Quais tipos de DT podem afetar projetos de software?; **(RQ2)** Quais são as estratégias que têm sido propostas para identificar ou gerenciar a DT em projetos de software?; **(RQ3)** Quais causas levam à ocorrência de DT?

Com a finalidade de atingir o objetivo do trabalho e responder às questões de pesquisa, uma metodologia foi definida baseada no paradigma da engenharia de software experimental [Basili 1992] e considerou a execução de dois estudos: (i) um mapeamento sistemático da literatura para responder às RQs 1 e 2, e (ii) um estudo de entrevista para responder à RQ3. Como resultado, este trabalho organizou os tipos de dívida em uma taxonomia [Alves *et at.* 2014], mapeou estratégias de identificação e gerenciamento da DT [Alves *et al.* 2016], identificou um conjunto de causas que levam equipes de desenvolvimento a incorrerem DT em seus projetos [Alves *et al.* 2017] e

organizou e compartilhou todo este conhecimento em TD Wiki[1] [Alves *et al.* 2015], de forma que as informações sejam facilmente acessadas por pesquisadores e profissionais.

Além desta introdução, este artigo está estruturado em mais sete seções. Na Seção 2 é apresentada uma fundamentação teórica sobre DT. A Seção 3 apresenta a metodologia empregada e os resultados alcançados nesta pesquisa. Em seguida, a Seção 4 discute o objetivo, metodologia e resultados alcançados do mapeamento sistemático da literatura executado. A Seção 5 apresenta a taxonomia de tipos de DT organizada. Em seguida, a Seção 6 discute o objetivo, metodologia e resultados do estudo de entrevista realizado. A Seção 7 apresenta como o corpo de conhecimento organizado foi disponibilizado na infraestrutura TD Wiki. Finalmente, a Seção 8 apresenta as considerações finais deste trabalho.

2. Dívida Técnica

Desde sua citação inicial por Cunningham (1992), que utilizou o termo "*going into debt*" para explicar que uma pequena dívida pode acelerar o desenvolvimento de software, mas que cada minuto extra gasto com o código mal construído conta como juros sobre essa dívida, o conceito de DT tem se expandindo e abrange, atualmente, diferentes artefatos gerados ao longo do ciclo de desenvolvimento do software. Nesse sentido, pesquisas têm sido realizadas para tratar a DT sob diferentes perspectivas [Kruchten *et al.* 2012] [Zazworka *et al.* 2013] [Guo *et al.* 2014]: estratégias de gerenciamento, identificação, quantificação, aspectos financeiros, entre outros.

Existem algumas iniciativas com o objetivo de organizar os diferentes tipos de DT que podem afetar um projeto. Fowler (2003) classificou as dívidas considerando as características: imprudente/prudente e deliberado/inadvertido. Estas características permitem classificar a dívida quanto ao fato dela ter sido inserida de forma intencional ou não e, para ambos os casos, se ela pode ser considerada o resultado de uma falta de cuidado ou foi inserida com cautela. Próxima à classificação de Fowler, McConnell (2007) classificou a dívida em dois grupos: intencional e não intencional. A dívida não intencional ocorre devido a uma falta de atenção ou desconhecimento sobre a forma mais correta de desempenhar uma tarefa. Já a dívida intencional é originada de forma proativa por razões táticas ou estratégicas com o objetivo de cumprir o prazo de entrega.

Para que seja possível tornar a DT explícita e gerenciável, o primeiro passo é identificá-la. Estudos têm demonstrado que diferentes estratégias automatizadas podem ser utilizadas para apoiar a identificação da DT [Schumacher *et al.* 2010]. Também já se tem evidência de que mesmo utilizando soluções automatizadas, é importante considerar a percepção humana no processo de identificação uma vez que ela permite agregar informações contextuais a cada instância de DT encontrada e, dessa forma, aperfeiçoar a avaliação feita por ferramentas automatizadas [Zazworka *et al.* 2013].

Associada à atividade de identificação da dívida, estratégias de gerenciamento de DT também têm sido propostas [Guo *et al.* 2014]. Elas buscam identificar e monitorar os itens de DT inseridos no projeto de forma que eles estejam explícitos e sejam pagos no momento oportuno para evitar transtornos maiores.

[1] www.tdwiki.com

Entre as razões para incorrer em DT podem estar prazos muitos apertados, falta de recursos e metas com alto custo [Buschmann 2011]. Embora relevante para apoiar a gestão da DT sob uma perspectiva preventiva, a identificação de causas que levam à ocorrência da DT ainda é um assunto pouco explorado na literatura técnica.

3. Metodologia e Resultados da Pesquisa

A metodologia utilizada neste trabalho foi fundamentada no paradigma da Engenharia de Software Experimental [Basili 1992], que busca aprimorar a engenharia de software aplicando a abordagem científica na construção e/ou evolução de métodos, teorias e técnicas, sejam eles novos ou já existentes, para apoiar as atividades de desenvolvimento.

A Figura 1 apresenta as atividades realizadas, sendo que para cada uma delas são descritos os respectivos objetivos, resultados e publicações obtidas. Conforme pode ser observado, dois estudos experimentais (destacados em cinza escuro) foram executados: um mapeamento sistemático da literatura e um estudo de entrevista. O planejamento do mapeamento sistemático foi fundamento na revisão informal da literatura realizada na atividade 1 e, além de seus próprios resultados [Alves *et al.* 2016], possibilitou a definição de uma taxonomia de tipos de DT [Alves *et al.* 2014] na atividade 3. Já o estudo de entrevista executado na atividade 4 possibilitou investigar uma lacuna observada durante a execução do mapeamento: a identificação de causas que levam à ocorrência da DT [Alves *et al.* 2017]. Os resultados de todos esses estudos permitiram organizar um corpo de conhecimento sobre DT que foi estruturado e compartilhado na infraestrutura TD Wiki [Alves *et al.* 2015] elaborada na atividade 5.

Uma descrição mais aprofundada sobre as atividades 2, 3, 4 e 5 considerando seus objetivos, metodologia e resumo dos resultados alcançados será apresentada, respectivamente, nas Seções 4, 5, 6 e 7 deste artigo.

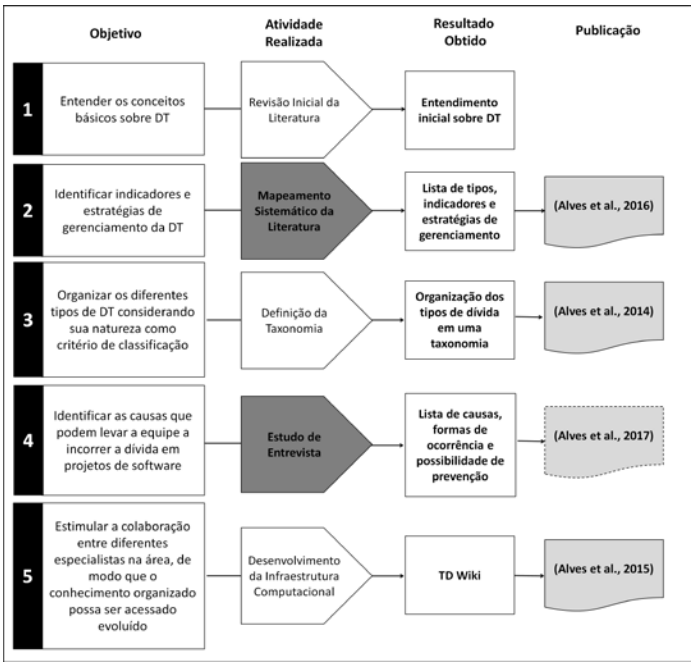

Figura 1. Metodologia e resultados da pesquisa.

4. Identificação e Gerenciamento da Dívida Técnica: Um Mapeamento Sistemático da Literatura

4.1 Objetivo

Esta seção apresenta os resultados de um mapeamento sistemático da literatura conduzido com o objetivo de responder a seguinte questão de pesquisa: *"Quais são as estratégias que têm sido propostas para identificar ou gerenciar a DT em projetos de software?"*. As seguintes questões complementares foram derivadas da questão principal:

- **(Q1)** Quais são os tipos de DT?
- **(Q2)** Quais são as estratégias propostas para identificar a DT?
  - **(Q2.1)** Quais avaliações experimentais foram realizadas?
  - **(Q2.2)** Quais são as fontes de dados e artefatos que têm sido propostos para identificar a DT?
  - **(Q2.3)** Quais técnicas de visualização de software têm sido propostas para apoiar a identificação de DT?
- **(Q3)** Quais estratégias têm sido propostas para apoiar o gerenciamento da DT?
  - **(Q3.1)** Quais avaliações experimentais foram realizadas?
  - **(Q3.2)** Quais técnicas de visualização de software têm sido propostas para apoiar o gerenciamento da DT?

Ao responder estas questões, foram identificados os tipos de DT, os indicadores de sua existência em projetos e as estratégias que têm sido desenvolvidas para apoiar a gestão dessa dívida. Além disso, o grau de maturidade das propostas existentes foi investigado através de uma análise das avaliações experimentais que foram realizadas. Também foi pesquisado como recursos de visualização de software têm sido utilizados para apoiar a identificação e monitoramento da DT, identificando quais metáforas visuais têm sido propostas e quais são as plataformas que estão sendo usadas para apresentar os diferentes tipos de dívida.

4.2 Metodologia

O mapeamento sistemático da literatura fornece um meio para realizar revisões de literatura abrangentes e imparciais, proporcionando valor científico. Ele é um tipo de estudo secundário que tem como objetivo caracterizar uma determinada área de pesquisa através de um procedimento sistemático que visa identificar a extensão e a natureza dos principais estudos disponíveis na área [Budgen *et al.* 2008]. No mapeamento realizado neste trabalho, seguiu-se um processo baseado no definido por Peterson *et al.* (2008), que considera as seguintes atividades: definição das questões de pesquisa, definição das fontes de dados, condução da busca, seleção dos estudos, extração de dados e mapeamento dos resultados. Maiores detalhes sobre o planejamento do estudo podem ser encontrados em [Alves *et al*. 2016].

A identificação e a filtragem dos artigos foram realizadas em cinco etapas (Figura 2). A primeira consistiu na busca por artigos em cada uma das bibliotecas digitais selecionadas para este estudo (ACM Digital Library, IEEE Xplore, Science Direct, Engenharia Village, Springer Link, Scopus, CiteSeer e DBLP). No segundo passo, um pesquisador realizou a primeira filtragem utilizando os critérios de exclusão.

Esse processo resultou na exclusão de todos os relatórios de workshops/conferências, apresentações em Power Point e artigos duplicados.

Na terceira etapa, cada artigo foi analisado por dois pesquisadores e, no caso de conflito, um terceiro analisaria e tomaria a decisão de incluir ou não o estudo. A filtragem ocorreu através da leitura dos títulos, abstracts e introdução usando os critérios de inclusão e exclusão. Após esta etapa, 100 estudos foram selecionados para o próximo estágio de filtragem. Na quarta etapa, o último filtro foi aplicado. Desta vez, os trabalhos selecionados foram lidos na íntegra. Ao final desta fase, mais 12 artigos foram excluídos por não possuírem informações suficientes sobre identificação ou gerenciamento de DT. Finalmente, na quinta etapa foi aplicado o *snowballing*, verificando as referências de cada estudo selecionado. Ao final, 100 estudos foram selecionados para extrair as informações e responder às questões de pesquisa.

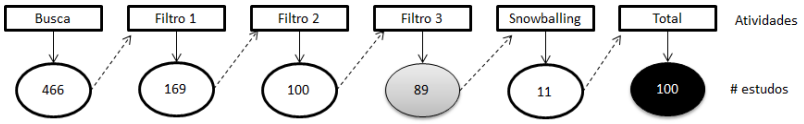

Figura 2. Processo de filtragem dos artigos

4.3 Resumo dos Resultados

A Figura 3 representa uma visão consolidada do estudo realizado considerando a relação entre o número de artigos analisados por assunto (indicadores, artefatos, fontes de dados, estratégia de gerenciamento, visualização de software), os tipos de DT identificados e a quantidade de estudos experimentais realizados. Pode-se observar que a maioria dos estudos lida com DT no nível de código fonte, ou seja, dívida de projeto, defeito, código e arquitetura (área superior esquerda do gráfico de bolhas).

Também é possível observar a existência de tipos de dívida (ex.: requisitos, código, teste) ao longo de todo o ciclo de desenvolvimento do software. Assim, garantir a qualidade do código fonte não é a única maneira de melhorar a qualidade do projeto. Apesar disso, grande parte dos estudos identificados se concentram em analisar os problemas existentes no código. Essa ênfase pode ser explicada pelo fato de que já há um conjunto considerável de métricas e ferramentas que permitem extrair informações do código que podem ser utilizadas como indicadores da presença da dívida.

Este trabalho também identificou vários estudos sobre estratégias de gerenciamento da DT. No entanto, embora diferentes estratégias tenham sido identificadas, apenas cinco delas (Abordagem de Portfólio, Análise de Custo-Benefício, Processo Analítico Hierárquico, Cálculo do DT-principal, e Marcação das dependências e problemas de código) foram citadas em mais de dois trabalhos e somente algumas delas foram avaliadas. Isso mostra que a maioria dos autores propuseram novas estratégias, mas poucos estão realizando estudos para avaliar a sua aplicabilidade.

Também é possível observar na Figura 3 que ainda são poucos os estudos realizados sobre o uso de técnicas de visualização de software para apoiar as atividades relacionadas à DT. Além disso, quando se observa o lado direito do gráfico, onde os estudos experimentais identificados são apresentados, é possível notar que o conhecimento sobre os benefícios e limitações do que tem sido proposto pela comunidade de pesquisa ainda é limitado.

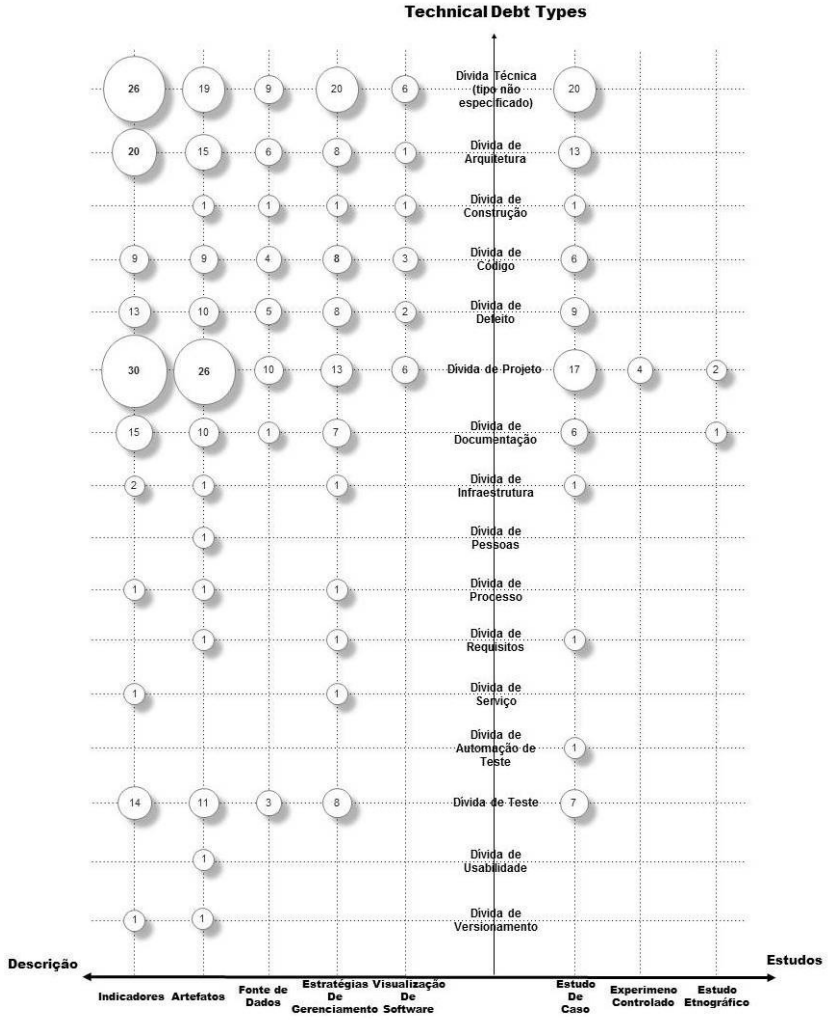

Figura 3. Visualização dos resultados do mapeamento sistemático

Este mapeamento sistemático buscou, principalmente, orientar futuras pesquisas sobre identificação e gerenciamento de DT. Entretanto, seus resultados também têm implicações importantes para profissionais, particularmente para aqueles que buscam na literatura orientação sobre como gerenciar DT em projetos reais:

- DT pode ser encontrada em diferentes artefatos. Assim, várias estratégias devem ser empregadas se o objetivo é encontrar todos os tipos de DT que podem trazer um impacto negativo para o projeto;
- Existem diferentes indicadores para cada tipo de DT. Assim, os desenvolvedores possuem opções para definir uma estratégia para identificar e rastrear a DT em seus projetos, e deveriam definir critérios para a escolha dos mais adequados;
- A maioria das pesquisas sobre identificação de DT é relacionada a dívida de código. Isto poderia sugerir que se concentrar em atividades de identificação de DT considerando, inicialmente, a dívida relacionada com código faria sentido. No entanto, este tipo de decisão deve ser assumido como sendo um risco porque a dívida pode estar oculta no projeto de formas diferentes e aquelas não relacionadas a código também podem trazer impactos negativos significativos. Assim, sugere-se evitar limitar o escopo da identificação da dívida àquelas

relacionadas apenas a código;

- Foram identificadas diferentes estratégias de gestão de DT. A maioria delas ainda requer uma investigação mais aprofundada e avaliação experimental. No entanto, elas podem ser um ponto de partida para a customização ou a definição de uma estratégia de gerenciamento de DT em um projeto de software real.

Para pesquisadores, os resultados do estudo trazem as seguintes implicações:

- Existem diferentes tipos de DT e alguns indicadores para cada um deles, mas não foi identificada nenhuma evidência sobre como utilizar este conjunto de informações para orientar iniciativas de identificação de DT em situações reais. Apesar dos progressos em diferentes áreas da DT, ainda há uma necessidade de analisar o todo e investigar abordagens holísticas para gerir de forma eficaz a DT na indústria de software;
- Poucos estudos experimentais foram realizados em ambientes reais. Este é um indicador de que, em algumas áreas, ainda não é possível entender completamente todos os custos ou benefícios dos indicadores de DT e estratégias de gerenciamento propostas. Algumas delas foram apenas citadas;
- A pesquisa sobre DT é altamente concentrada em alguns tipos de dívida (projeto, arquitetura, código e defeito). Isto indica uma lacuna que pode ser explorada nos próximos anos.

Os resultados obtidos também indicaram que DT é uma área de pesquisa ativa e produtiva, que continua a crescer e que ainda necessita de maior maturidade em termos de conceitos consolidados e novas avaliações experimentais.

5. Taxonomia de Tipos de Dívida Técnica

5.1 Objetivo

Esta seção apresenta a definição de uma taxonomia para organizar os diferentes tipos de DT. Depois de definida, a taxonomia foi avaliada em duas etapas. Na primeira delas, critérios de qualidade adaptados de Gruber (1995) foram considerados. Em um segundo momento, um especialista da área, que não participou do desenvolvimento da taxonomia, fez uma avaliação do conhecimento organizado.

5.2 Metodologia

A construção de taxonomias não é uma tarefa simples e deve ser apoiada por práticas da engenharia de software. Neste trabalho foi utilizado um processo estruturado para guiar a sua definição. O processo foi uma adaptação da abordagem SABiO definida por Falbo *et al.* (1998) para apoiar a construção de ontologias. Embora existam abordagens específicas para apoiar a definição de taxonomias, SABiO foi escolhida por já ter sido utilizada no apoio à definição de diferentes vocabulários de conhecimento. As seguintes atividades foram desempenhadas:

- **Identificação de Propósitos e Especificação de Requisitos:** neste trabalho, o propósito da taxonomia é organizar os diferentes tipos de DT considerando sua natureza como critério de classificação;
- **Captura e Formalização da Taxonomia:** os tipos de DT definidos na taxonomia, assim como suas definições, foram identificados a partir do resultado

do mapeamento sistemático apresentado na seção 4. Associadas a cada um dos tipos, outras informações também foram especificadas: indicadores que podem ser utilizados para identificar a DT em projetos e referências de onde as informações foram extraídas. Uma vez capturados os conceitos que definiam a taxonomia, realizou-se sua formalização na linguagem OWL.

- **Avaliação da Taxonomia:** a taxonomia definida foi avaliada em duas etapas:
  - **Avaliação a partir dos critérios de qualidade [Gruber 1995]:** para a primeira etapa, a taxonomia foi enviada para dois pesquisadores do grupo de pesquisa sobre dívida técnica TDResearchTeam[2]. Os pesquisadores foram convidados a avaliá-la considerando os seguintes critérios: Clareza, Extensibilidade, Viés de codificação mínimo e Compromissos mínimos. No geral, alguns pequenos ajustes foram solicitados de forma a tornar mais clara a diferença entre alguns tipos de DT e melhorar a definição de alguns exemplos;
  - **Avaliação por um especialista na área:** A segunda etapa da avaliação considerou as recomendações de Gruber (1995), que indica que representações de conhecimento devem ser embasadas no consenso de um grupo de especialistas da área. Neste trabalho, esta atividade foi desempenhada por um especialista na área de DT. Para isso, os tipos identificados e suas respectivas definições foram organizados em um formulário e enviados para avaliação. O especialista que participou desta etapa é um pesquisador sênior sendo, atualmente, um dos mais atuantes na área de DT e estudos qualitativos em engenharia de software experimental. Como resultado da avaliação, um conjunto de ajustes foi sugerido no sentido de tornar mais claras algumas definições que, em alguns casos, estavam descritas sob diferentes perspectivas. Além disso, foi indicado também que os tipos identificados faziam sentido, nenhum novo tipo foi sugerido.

5.3 Resumo dos Resultados

Os tipos de DT estruturados na taxonomia foram [Alves *et al.* 2014]:

- **Dívida de Arquitetura:** refere-se a problemas encontrados na arquitetura do software que podem afetar os requisitos arquiteturais do projeto;
- **Dívida de Automação de Testes:** refere-se ao trabalho envolvido na automatização de testes de funcionalidades previamente desenvolvidas para apoiar a integração contínua e ciclos mais rápidos de desenvolvimento;
- **Dívida de *Build* (Construção):** refere-se a questões que tornam a tarefa de compilação mais difícil e desnecessariamente demorada;
- **Dívida de Código:** refere-se a problemas encontrados no código que podem afetar negativamente a legibilidade do código tornando-o mais difícil de manter;
- **Dívida de Defeito:** refere-se a defeitos conhecidos que o comitê de controle de configuração concorda que devem ser reparados, mas que devido a prioridades concorrentes e recursos limitados, têm de ser adiados para mais tarde;
- **Dívida de Documentação:** refere-se a problemas encontrados na documentação

[2] www.tdresearchteam.com

como documentação inexistente, inadequada ou incompleta;

- **Dívida de Infraestrutura:** refere-se a questões de infraestrutura que podem atrasar ou impedir algumas atividades de desenvolvimento;
- **Dívida de Pessoas:** refere-se a questões relacionadas a pessoas que podem atrasar ou impedir a realização de algumas atividades de desenvolvimento;
- **Dívida de Processo:** refere-se a processos ineficientes;
- **Dívida de Projeto:** pode ser descoberta através da análise do código fonte e identificação de violações de princípios da orientação a objetos;
- **Dívida de Requisitos:** refere-se a decisões sobre quais requisitos a equipe de desenvolvimento precisa implementar ou como implementá-los;
- **Dívida de Serviço:** refere-se à seleção e substituição inadequada de serviços *web* que levam à incompatibilidade entre as características do serviço e os requisitos da aplicação;
- **Dívida de Teste:** refere-se a problemas (ex.: baixa cobertura dos testes) encontrados em atividades de teste que podem afetar sua qualidade;
- **Dívida de Usabilidade:** refere-se a decisões inadequadas de usabilidade que terão de ser ajustadas mais tarde;
- **Dívida de Versionamento**: refere-se a problemas de versionamento do código fonte, como uso desnecessário de *forks.*

A taxonomia definida permite o compartilhamento de um vocabulário comum para comunidade de pesquisa em DT e profissionais da área. Embora ela seja uma tentativa inicial de organizar os tipos de dívida, a lista de tipos pode não ser completa, podendo levar à necessidade de exclusão de alguns deles ou inclusão novos tipos. Para obter uma taxonomia melhor definida, a colaboração entre diferentes especialistas precisa ser estimulada, de forma que que eles participem de seu desenvolvimento. Para isso, foi desenvolvida uma infraestrutura (apresentada na Seção 7) para permitir o compartilhamento e evolução colaborativa desse conhecimento.

6. Causas que Levam à Inserção da DT: Um Estudo de Entrevista

6.1 Objetivo

O estudo de entrevista foi conduzido com o objetivo de **analisar** os diferentes tipos de DT, **com o propósito de** caracterizar, **com respeito** às causas que levam a sua inserção, **no contexto de** projetos de desenvolvimento de software **sob o ponto de vista de** profissionais da área [Alves *et al.* 2017]. Alinhado a este objetivo, foram definidas as seguintes questões de pesquisa: (Q1) Quais causas levam à ocorrência de dívida técnica?; (Q2) As causas ocorrem de forma encadeada ou isolada?; (Q3) A dívida técnica pode ser evitada?; (Q4) Em termos de esforço, é melhor evitar a dívida ou incorrê-la para pagar depois?

6.2 Metodologia

O estudo de entrevista foi planejado e executado considerando profissionais da área. A escolha dos participantes foi realizada por conveniência e considerou dois parceiros da indústria: Centro de Projetos Fraunhofer para Engenharia de Software e Sistemas e SENAI Cimatec. Ao total, 10 engenheiros de software foram convidados a participar do estudo via e-mail e foram entrevistados presencialmente.

Antes de realizar a entrevista, um formulário de caracterização e consentimento foi preenchido pelos participantes do estudo. O resultado dessa caracterização foi utilizado para apoiar a escolha dos tipos de dívida sobre os quais cada participante deveria responder. As entrevistas foram realizadas individualmente, sendo que cada entrevistado respondeu às perguntas sobre três tipos de dívida. Como cada entrevistado respondeu a cada pergunta três vezes (cada uma para um tipo de dívida) e o estudo contou com a participação de 10 engenheiros de software, ao final, 30 respostas foram obtidas para cada questão realizada.

No início de cada entrevista foi realizada uma breve apresentação do tema DT, que durou cerca de 10 minutos. Em seguida, para cada tipo de dívida definido para o entrevistado, os passos apresentados a seguir foram considerados:

1. A definição do tipo de dívida selecionado foi apresentada;
2. Foi pedido ao entrevistado que ele fornecesse um exemplo, com base em sua experiência, do tipo de dívida que estava sendo discutido, para confirmar o entendimento do entrevistado sobre o tipo considerado;
    2.1. Se o entrevistado desse um exemplo que não fosse do tipo de dívida em discussão, seria pedido um segundo exemplo explicando novamente a definição daquele tipo;
        2.1.1. Se o entrevistado não conseguisse pensar em um exemplo sobre o tipo de dívida escolhido, este tipo deveria ser substituído por um outro tipo.
    2.2. Se o entrevistado apresentasse um exemplo que representasse o tipo de dívida discutido, a entrevista prosseguia para o próximo passo;
3. Foi perguntado ao entrevistado o que levou a equipe de desenvolvimento a inserir o item de dívida descrito em seu exemplo;
    3.1. Em seguida, foi perguntado se algum outro motivo teria levado ou contribuído para o motivo citado. Esta pergunta foi realizada repetidas vezes de forma que detalhes mais específicos sobre o que poderia ter levado a equipe a incorrer na dívida fosse identificado (o objetivo aqui foi fugir do óbvio e tentar entender se as causas para a inserção da dívida ocorrem em cadeia);
4. Foi perguntado ao entrevistado o que, na opinião dele, teria evitado a DT;
    4.1.Em seguida, foi perguntado se valeria a pena ter investido na prevenção da dívida considerando o que ele indicou na pergunta anterior ou se fazer isso seria mais caro do que incorrer na própria dívida.

As entrevistas foram gravadas com autorização dos participantes. Depois de realizadas, elas foram transcritas para que a análise dos dados pudesse ser realizada. Os textos transcritos das entrevistas passaram por um processo de codificação e, então, as evidências coletadas foram avaliadas frente às questões de pesquisa.

A codificação foi realizada por um pesquisador e, depois, avaliada por um segundo de acordo com o esquema de codificação definido. Este esquema partiu de um conjunto inicial de códigos e evoluiu à medida que a análise dos dados foi realizada. Os resultados do estudo foram formulados com base na síntese dos fragmentos codificados de cada questão. Assim, os resultados foram fundamentados nos dados do estudo, em vez de formulados anteriormente e validados através dos dados.

6.3 Resumo dos Resultados

Ao total foram identificadas 84 causas distintas (Q1). Destas, as mais citadas foram prazo (13 citações), custo (5 citações), documentação inexistente (4 citações) e alta rotatividade na equipe (4 citações). Foi possível observar também que a maioria das causas estavam associadas a um tipo de dívida. A presença de causas compartilhadas e específicas indica que ações para prevenir a dívida podem ter um efeito específico ou contribuírem para o tratamento de diferentes tipos de dívida.

Em relação à Q2, no geral, os resultados indicaram que uma série de fatores leva à presença da dívida no projeto. Embora as causas possam afetar o projeto de forma isolada, quando ocorrem em cadeia a consequência é potencializada. Além disso, as causas podem possuir pesos diferentes na contribuição para a ocorrência da dívida. Para questão de pesquisa Q3, os entrevistados indicaram em sua maioria que a DT poderia ser evitada. Apenas dois participantes indicaram que a dívida não poderia ser evitada. Para aqueles que indicaram que a dívida não poderia ser evitada totalmente, o motivo estava associado à existência de uma causa que não poderia ser tratada.

Por fim, em relação à Q4, os resultados indicaram que é melhor e mais barato investir na prevenção da dívida do que incorrê-la e pagá-la depois. Foi possível observar também que alguns fatores pesam positivamente na decisão de prevenir a dívida: custo da correção tardia dos problemas inseridos no projeto, nível de retrabalhado necessário para eliminar a dívida e a motivação da equipe. Os dois últimos fatores impactam diretamente na produtividade da equipe de desenvolvimento. Por outro lado, podem existir situações em que é melhor incorrer a dívida e pagá-la depois, como o que ocorre quando o custo de evitar a dívida é maior do que o de inseri-la.

7. TD Wiki: Compartilhamento e Evolução do Conhecimento sobre DT

TD Wiki é uma infraestrutura computacional de apoio ao compartilhamento e evolução colaborativa do conhecimento sobre DT disponibilizada na web [Alves *et al.* 2015]. No contexto deste trabalho, as seguintes informações foram organizadas para compor a base de conhecimento: tipos de dívida técnica, indicadores, causas e estudos de avaliação. Existem duas expectativas principais para esta infraestrutura: (1) criar um canal para que o assunto DT seja efetivamente tratado na indústria de software através do compartilhamento das informações em um formato adequado para uso por profissionais; (2) permitir a evolução do conhecimento de forma colaborativa.

Em TD Wiki, o conhecimento foi organizado em duas camadas: visão geral e visão detalhada. A camada de visão geral está representada na página inicial (Figura 4 – imagem superior), na qual são apresentadas uma breve descrição de cada tipo de dívida e alguns de seus indicadores. Para isso, os tipos de dívida foram representados em uma árvore (quanto mais espessas forem as linhas, mas evidências já foram coletadas sobre aquele tipo de dívida). É a partir dessa árvore que o usuário pode obter mais detalhes sobre cada tipo.

Já na camada de visão detalhada (Figura 4 – imagem inferior), o usuário tem acesso aos indicadores conhecidos de cada tipo de dívida, as causas que podem levar à sua ocorrência, estudos de avaliação realizados e algumas referências. Nesta página, os usuários também podem indicar que tipo de DT eles estão trabalhando e quais

indicadores são mais adequados para a identificação daquele tipo. Também é possível indicar quais são as principais causas que contribuem para que um determinado tipo de dívida seja inserido no projeto. Essas informações coletadas dos usuários são utilizadas pela infraestrutura para representar a relevância dos indicadores e causas.

Além do acesso às informações, TD Wiki também permite a evolução colaborativa do conhecimento, possibilitando que usuários registrados adicionem novos tipos de dívida, indicadores, causas e referências. Para cada uma dessas funcionalidades, um grupo de três moderadores avalia as informações fornecidas. Se aprovada, a nova informação é incorporada à base de conhecimento e disponibilizada para visualização.

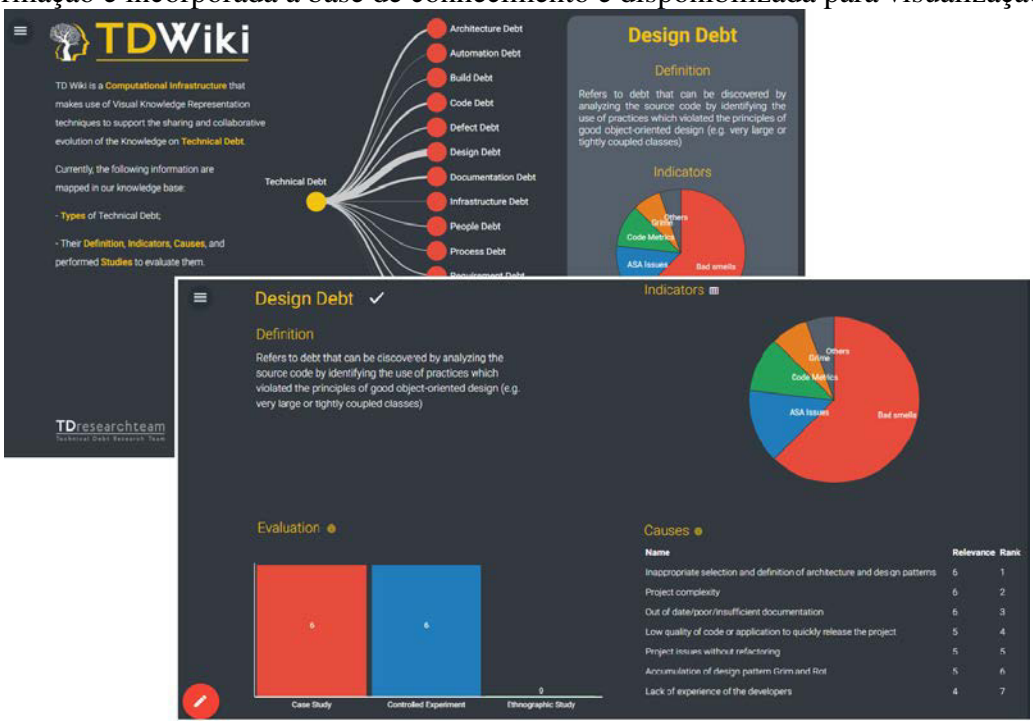

Figura 4. TD Wiki

8. Considerações Finais

Este trabalho organizou um corpo de conhecimento sobre DT considerando seus tipos, indicadores, estratégias de gerenciamento e causas para sua ocorrência. Em complemento, as informações organizadas foram disponibilizadas através da infraestrutura TD Wiki. Para alcançar estes resultados, a pesquisa seguiu duas linhas de trabalho: (1) realização de um mapeamento sistemático da literatura complementado por um estudo de entrevista para a coleta de evidências; (2) estruturação das informações identificadas em uma taxonomia de tipos de dívida e em uma infraestrutura que apoia o compartilhamento e evolução colaborativa do conhecimento.

Ao analisar as questões de pesquisa principais definidas para este trabalho e descritas na Seção 1, os seguintes resultados e contribuições foram obtidos: (i) Organização de um corpo de conhecimento sobre dívida técnica [Alves *et al.* 2016], (ii) Estruturação do corpo de conhecimento organizado [Alves *et al.* 2014], (iii) Identificação de causas que levam equipes de desenvolvimento a inserirem a dívida em seus projetos [Alves *et al.* 2017], e (iv) Compartilhamento do corpo de conhecimento organizado [Alves *et al.* 2015]. Em conjunto, estes resultados contribuem para a evolução do *Technical Debt Landscape* [Izurieta *et al.* 2012] [Kruchten *et al.* 2012].

Ainda há muito a ser pesquisado e explorado sobre DT. Considerando este trabalho como ponto de partida, algumas perspectivas futuras de trabalho são: (i) ainda não se sabe como as causas identificadas estão relacionadas a indicadores de presença da dívida. Algumas relações entre causas e indicadores parecem ser justificadas logicamente, por exemplo: falta de experiência dos desenvolvedores e violação de modularidade; falta de experiência dos desenvolvedores e dependências estruturais. No entanto, uma investigação aprofundada do tipo de relação (causa - consequência) ainda pode ser considerada uma área em aberto; (ii) realizar avaliação experimental da infraestrutura TD Wiki de modo que sua aplicabilidade seja avaliada, e; (iii) replicar o estudo de entrevista executado. Essa replicação permitirá que os resultados sejam mais generalizáveis e permitirá que análises específicas (por exemplo, por tipo de dívida) sejam realizadas.

Atualmente, os autores deste trabalho estão envolvidos em um projeto de pesquisa que busca investigar causas e efeitos que a dívida pode trazer em projetos de software. O trabalho tem sido realizado no contexto da pesquisa de doutorado da primeira autora deste artigo.

7. Agradecimentos

Agradecemos à CAPES e ao CNPq (Universal 458261/2014-9) pelo apoio financeiro para a realização deste trabalho. Agradecemos também aos demais membros do TDResearchTeam.com que contribuíram com esta pesquisa.

Referências

Alves, N.S.R., Araújo, R.S. and Spínola, R.O. (2015) A Collaborative Computational Infrastructure for Supporting Technical Debt Knowledge Sharing and Evolution. In: Americas Conference on Information Systems, Puerto Rico.

Alves, N.S.R., Mendes, T.S., 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. DOI: https://doi.org/10.1016/j.infsof.2015.10.008

Alves, N.S.R., Spínola, R.O., Mendonça, M.G. and Seaman, C. (2017) A study of factors that lead development teams to incur technical debt in software projects. Submetido em abril de 2017 para o Empirical Software Engineering Journal.

Alves, N.S.R., Ribeiro, L. F., Caires, V., Mendes, T.S. and Spínola, R.O. (2014) "Towards an Ontology of Terms on Technical Debt". In Proc. of the Sixth Int. Work. on Managing Technical Debt. IEEE Comp Society, Washington, USA, 1-7. DOI: 10.1109/MTD.2014.9

Basili, V.R. (1992) The Experimental Paradigm in Software Engineering. In Proc. of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions. Springer-Verlag, London, UK, UK, 3-12.

Budgen, D., Turner, M., Brereton, P. and Kitchenham, B. (2008) Using Mapping Studies in Software Engineering. In the Proceedings of PPIG Psychology of Programming Interest Group, Lancaster University, UK, pp. 195–204.

Buschmann, F. (2011) “To pay or not to pay technical debt,” IEEE Software, v. 28, n. 6, p. 29-31.

Cunningham, W. (1992) The WyCash portfolio management system. ACM SIGPLAN OOPS Messenger, 4(2), 29-30.

Falbo, R.A., Menezes, C.S. and Rocha, A.R.C. (1998) A systematic approach for building ontologies. In Ibero-American Conference on Artificial Intelligence (pp. 349-360). Springer Berlin Heidelberg.

Fowler, M. (2003) Technical Debt. Available: http://www.martinfowler.com/bliki/TechnicalDebt.html

Gruber, T.R. (1995) “Toward Principles For The Design Of Ontologies Used For Knowledge Sharing,” Int. Journal Human-Computer Studies, 43(5/6), p. 907-928.

Guo, Y., Spínola, R.O. and Seaman, C. (2014) Exploring the costs of technical debt management – a case study, Empirical Software Engineering, 1-24.

Izurieta, C., Vetro, A., Zazworka, N., Cai, Y., Seaman, C. and Shull, F. (2012) Organizing the technical debt landscape, in Third International Workshop on Managing Technical Debt (MTD), pp. 23-26.

Kruchten, P., Nord, R. and Ozkaya, I. (2012) Technical Debt: From Metaphor to Theory and Practice, Software, IEEE 29(6), 18-21.

Lehman, M.M. and Belady, L.A. (1985) Program evolution: processes of software change. Academic Press Professional, Inc.

Lientz, B.P., Swanson, E.B. and Tompkins, G.E. (1978) Characteristics of application software maintenance. Communications of the ACM, 21(6), 466-471.

Mcconnell, S. (2007) Technical Debt. 10x Software Development [Blog]. Available at: http://blogs.construx.com/blogs/stevemcc/archive/2007/11/01/technical-debt-2.aspx.

Parnas, D.L. (1994) Software aging. In Proceedings of the 16th international conference on Software engineering (pp. 279-287). IEEE Computer Society Press.

Petersen, K., Feldt, R., Mujtaba, S. and Mattson, M. (2008) Systematic mapping studies in software engineering. In the 12th International Conference on Evaluation and Assessment in Software Engineering, University of Bari, Italy.

Schumacher, J.; Zazworka, N.; Shull, F.; Seaman, C. and Shaw, M. (2010) Building empirical support for automated code smell detection, ESEM’10: Proceedings of the 2010 ACM-IEEE Int. Symp. on Empirical Software Engineering and Measurement.

Seaman, C. and Guo, Y. (2011) Measuring and Monitoring Technical Debt, Advances in Computers 82, 25-46.

Spínola, R., Zazworka, N., Vetro`, A., Seaman, C. and Shull, F. (2013) Investigating technical debt folklore: Shedding some light on technical debt opinion, in Managing Technical Debt (MTD), 2013 4th International Workshop on, pp. 1-7.

Zazworka, N., Spínola, R.O., Vetro, A., Shull, F. and Seaman, C. (2013) A case study on effectively identifying technical debt, EASE 13: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering.