MESTRADO EM SISTEMAS E COMPUTAÇÃO

RICARDO AZEVEDO PORTO

UM MAPA TEMÁTICO SOBRE DÍVIDA TÉCNICA DE PROJETO

Salvador
2018

RICARDO AZEVEDO PORTO

UM MAPA TEMÁTICO SOBRE DÍVIDA TÉCNICA DE PROJETO

Dissertação apresentada ao programa de Mestrado em Sistemas e Computação da. UNIFACS Universidade Salvador, Laureate International Universities, como requisito parcial para obtenção do grau de Mestre.

Orientador: Prof. Dr. Rodrigo Oliveira Spínola.

Salvador
2018

Ficha catalográfica elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador, Laureate International Universities.

Porto, Ricardo Azevedo

Um mapa temático sobre dívida técnica de projeto./ Ricardo Azevedo Porto.- Salvador, 2018.

92 f.: il.

Dissertação apresentada ao programa de Mestrado em Sistemas e Computação da. UNIFACS Universidade Salvador, Laureate International Universities, como requisito parcial para obtenção do grau de Mestre.

Orientador: Prof. Dr. Rodrigo Oliveira Spínola.

1. Desenvolvimento de *software*. 2. *M*apa temático. 3. Dívida técnica. I. Spinola, Rodrigo Oliveira, orient. II. Título.

CDD.004.6

RICARDO AZEVEDO PORTO

UM MAPA TEMÁTICO SOBRE DÍVIDA TÉCNICA DE PROJETO

Dissertação aprovada como requisito final para obtenção do grau de Mestre em Sistemas e Computação, 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
UNIFACS Universidade Salvador, Laureate International Universities

Sérgio Martins Fernandes _____
Doutor em Ciências da Computação pela Universidade de São Paulo - USP
UNIFACS Universidade Salvador, Laureate International Universities

Mário André de Freitas Farias _____
Doutor em Ciência da Computação - Ufba - Unifacs pela Universidade Federal da Bahia
Instituto Federal de Sergipe – IFS

Salvador, de de 2018.

"Cada dia que amanhece assemelha-se a uma página em branco, na qual gravamos os nossos pensamentos, ações e atitudes. Na essência, cada dia é a preparação de nosso próprio amanhã."

Chico Xavier

AGRADECIMENTOS

Primeiramente, a Deus! Pela oportunidade, força e persistência durante esta jornada. Agradeço à minha família, a meu pai Roberto Porto (In memoriam), minha mãe Maria Edinalva Azevedo Porto e, aos meus irmãos, Robson, Dayse e Roberto pelo companheirismo e o amor de vocês, com vocês me sinto fortalecido para enfrentar os desafios da vida com dignidade e humildade.

Ao meu orientador Profº Dr. Rodrigo Spínola pela paciência, disponibilidade e pelas contribuições realizadas ao longo desta pesquisa. Por fim, aos meus amigos pelo incentivo dado durante esta jornada.

RESUMO

Dívida técnica (DT) descreve a relação entre os retornos a curto prazo, obtidos através do adiamento de alguma atividade relacionada ao processo de desenvolvimento de *software*, e as consequências a longo prazo deste adiamento. Sendo assim, DT é um termo utilizado para descrever situações nas quais a equipe de desenvolvimento adota medidas que impactam na qualidade do *software* com o objetivo de atender a uma necessidade momentânea, tal como a liberação oportuna do *software*. Essa liberação oportuna, a princípio, pode parecer rentável com relação ao tempo e aumento de produtividade da equipe de desenvolvimento, pois resultará em uma entrega de *software* mais rápida. O conceito de DT tem despertado interesse na comunidade de engenharia de software e, com isso, diversos estudos secundários têm sido realizados na área. Esses estudos são relevantes para a área uma vez que são abrangentes, permitindo uma visão holística sobre os trabalhos que têm sido realizados. No entanto, nenhum deles realizou a aplicação de síntese temática com o objetivo de investigar os resultados existentes da área provenientes da execução de estudos experimentais. Neste contexto, o objetivo desta pesquisa é aplicar síntese temática para avaliar as evidências provenientes de avaliações experimentais identificadas em artigos da área de DT com foco em dívida de projeto. Para isso, uma revisão sistemática da literatura foi executada. Os resultados permitiram realizar o mapeamento e caracterização dos estudos experimentais primários que foram realizados na área de DT e a elaboração de um mapa temático sobre a dívida técnica de projeto, que é o tipo de dívida que tem sido mais discutido na literatura técnica.

Palavras chaves: Dívida Técnica. Dívida Técnica de Projeto. Revisão Sistemática da Literatura. Síntese Temática. Mapa Temático.

ABSTRACT

Technical Debt (TD) describes relationship between short-term returns obtained throw postponing some activity related to software development process, and the long-term consequences of this postponement. Thus TD is a term used to describe situations in which the development team adopts measures that impact the quality of the software in order to meet a momentary need, such as the timely release of the software. This timely release may at first seem cost-effective with respect to the time and productivity gains of development team as it will result in faster software delivery. The concept of TD has attracted interest in the software engineering community and, thereby, several secondary studies have been carried out in the area. These studies are relevant to the area since they are comprehensive, allowing a holistic view on the work that has been undertaken. However, none of them performed the application of thematic synthesis with the objective of investigating the existing results of the area from the execution of experimental studies. In this context, the objective of this research is to apply thematic synthesis to evaluate the evidence from experimental evaluations identified in articles in the TD area with focus on project debt. For this, a systematic review of the literature was performed. The results allowed the mapping and characterization of the primary experimental studies that were carried out in the TD area and the elaboration of a thematic map about technical project debt, which is the type of debt that has been most discussed in the technical literature.

Keywords: Technical Debt. Technical Project Debt. Systematic Review of Literature. Thematic Synthesis. Thematic map.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

DT	Dívida Técnica
DTSA	DT Self-Admitted
ESBE	Engenharia de Software baseada em Evidências
MS	Mapeamento Sistemático
RA	Requisitos Ágeis
RL	Revisão da Literatura
RS	Revisão Sistemática

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

Extrapolação de custos e o não cumprimento de prazos sempre foram considerados alguns dos principais problemas enfrentados pelas empresas desenvolvedoras de software. Muitas vezes, essas organizações, a fim de evitar esses dois problemas, tendem a tomar decisões que priorizam a entrega rápida dos produtos de *software*. Entretanto, essas decisões geralmente implicam em adiar algumas atividades relacionadas ao processo de desenvolvimento e trazem como consequência, por exemplo, a falta de documentação, a não implementação de todos os requisitos, testes não executados e etc. Muitos desses problemas têm sido tratados recentemente utilizando o conceito de Dívida Técnica (DT).

De acordo com Ozkaya *et al.* (2011), DT é um conceito relativamente novo na comunidade da engenharia de *software*, tendo sido mencionado pela primeira vez em 1993 por Ward Cunningham. Segundo Zazworka *et al.* (2013), DT descreve a relação entre os retornos a curto prazo, obtidos através do adiamento de alguma atividade relacionada ao processo de desenvolvimento de *software*, e as consequências de longo prazo desse adiamento. Atualmente, o conceito de DT tem sido aplicado a diferentes etapas de um processo de desenvolvimento, podendo ser categorizado nos seguintes tipos (ALVES *et al.*, 2016): projeto, arquitetura, documentação, teste, código, defeito, requisitos, infraestrutura, pessoas, automação de testes, processo, build, serviço, usabilidade e versionamento. Ainda segundo Alves *et al.* (2016), a DT de projeto é a que tem sido mais investigada pela comunidade de pesquisa e se refere a itens que podem ser descobertos através da análise do código fonte, bem como identificação de violações das boas práticas de orientação a objetos. Este fato motivou este trabalho a ser direcionado para investigar esse tipo de DT.

1.2 CONTEXTO E MOTIVAÇÃO

O conceito de DT tem despertado interesse na comunidade de engenharia de *software* e, com isso, diversos trabalhos têm sido realizados com o intuito de reunir evidências a partir de estudos primários. Exemplos de revisões e mapeamentos sistemáticos da literatura já realizados são: Tom *et al.* (2012), Villar e Matalonga (2013), Tom *et al.* (2013), Ampatzoglou *et al.* (2015), Li *et al.* (2015) e Alves *et al.* (2016). Esses estudos são relevantes para a área uma vez que são abrangentes, permitindo uma visão holística sobre os trabalhos que têm sido realizados. No entanto, nenhuma destas pesquisas realizou a aplicação de síntese temática com o objetivo de investigar os resultados existentes da área provenientes da execução de estudos experimentais.

A síntese temática é uma abordagem que utiliza os princípios da análise temática, uma técnica frequentemente empregada para análise e identificação de padrões (temas) dentro de dados observados em uma pesquisa qualitativa primária Cruzes e Dyba (2011). Aplicada na área de DT, essa abordagem pode auxiliar na investigação se as evidências reportadas na literatura técnica possuem resultados alinhados ou contraditórios e se elas apontam tendências para trabalhos futuros de pesquisa.

1.3 OBJETIVO E QUESTÕES DE PESQUISA

O objetivo deste trabalho é **aplicar síntese temática para avaliar as evidências provenientes de avaliações experimentais identificadas em artigos da área de DT com foco em dívida de projeto**. Esse objetivo foi traduzido na seguinte questão de pesquisa principal: **RQ - Qual o estado atual das pesquisas sobre DT de projeto considerando as evidências geradas através de estudos primários?**

Para estruturar as atividades e responder essa questão de pesquisa, as seguintes questões complementares foram definidas:

a) **(QC1) Qual o tipo de experimento realizado no artigo?** O tipo de experimento adotado permitirá avaliar o nível de controle adotado pelo estudo, assim como a abrangência dos resultados;

b) **(QC2) Qual o perfil dos participantes do estudo?** As informações identificadas através desta questão permitirão mapear o perfil dos participantes utilizados nos estudos.

Estas informações permitem investigar a inserção das tecnologias que dizem respeito à DT em ambientes acadêmicos e industriais;

c) **(QC3) O pacote experimental utilizado no estudo foi disponibilizado? Caso o pacote tenha sido disponibilizado, quais os meios utilizados?** Esta informação servirá para conhecimento acerca dos pacotes experimentais utilizados. Além disso, também será relevante no sentido de indicar se os estudos realizados são passíveis de serem replicados por outros pesquisadores;

d) **(QC4) Quais tipos de DT foram considerados na avaliação experimental?** Essa questão busca identificar quais tipos de DT estão sendo estudados na literatura, permitindo assim, que seja realizada uma análise sobre os limites de uso do conceito no desenvolvimento de *software*;

e) **(QC5) Qual a classificação dos projetos analisados?** A identificação se um projeto é *open source* ou proprietário permitirá auxiliar futuras pesquisas quanto ao nível de acesso e de conhecimento acerca dos autores com relação aos projetos;

f) **(QC6) Os projetos analisados utilizaram alguma ferramenta? Qual?** Esta questão busca reunir as ferramentas que têm sido desenvolvidas e/ou utilizadas em estudos da área de DT. Essa organização permitirá mapear ferramentas existentes e a possibilidade de seu uso;

g) **(QC7) Entre os estudos analisados, algum deles é continuação ou replicação de outro(s)?** Essa questão busca investigar se uma preocupação crescente na comunidade de engenharia de *software* tem sido considerada na área de DT: a replicação de estudos;

h) **(QC8) Quais são as evidências encontradas através da avaliação experimental?** As evidências encontradas a partir dos estudos experimentais analisados auxiliarão novos estudos com relação a pesquisas sobre DT, bem como uma melhor compreensão e entendimento.

Com a finalidade de atingir o objetivo do trabalho e responder a questão de pesquisa principal, foram definidos dois objetivos específicos:

a) Identificar como resultados de experimentos podem ser sintetizados em revisões sistemáticas da literatura;

b) Planejar e executar uma revisão sistemática para sintetizar evidências experimentais da área de DT de projeto.

1.4 METODOLOGIA E HISTÓRICO DA PESQUISA

Toda produção científica necessita estar amparada em técnicas que a apoiam e dão sustentação teórica à metodologia aplicada. De acordo com os objetivos a serem atingidos nesta dissertação, a pesquisa fará o uso de uma revisão controlada da literatura.

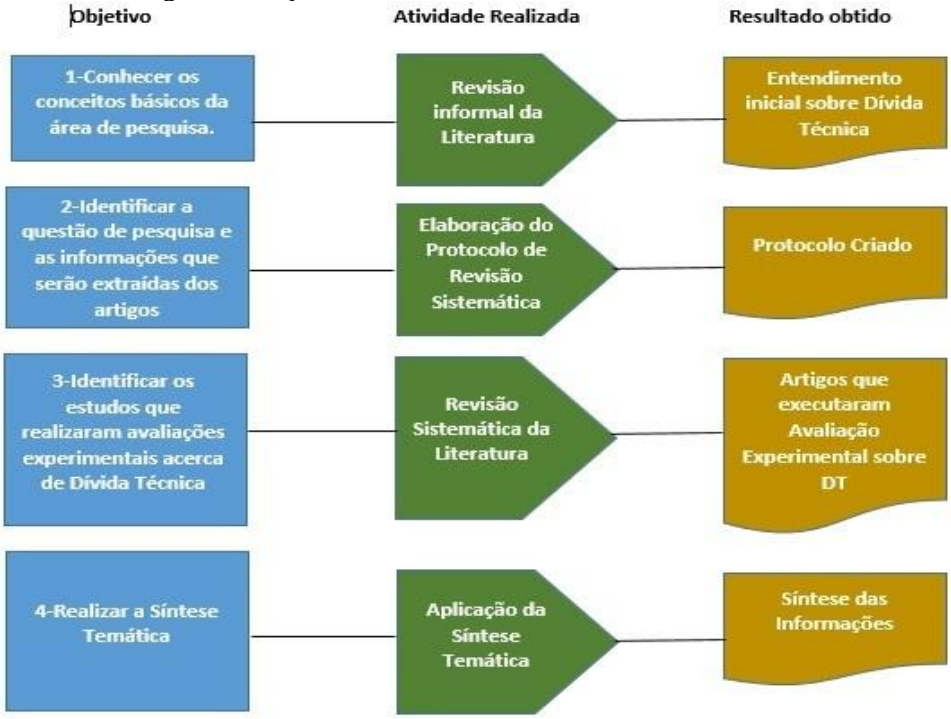
Segundo Gerhardt e Silveira (2009), a pesquisa bibliográfica é realizada a partir do levantamento de referências teóricas já analisadas, e publicadas por meios escritos e eletrônicos, como livros, artigos científicos, páginas de web sites. Qualquer trabalho científico inicia-se com uma pesquisa bibliográfica, que permite ao pesquisador conhecer o que já se estudou sobre o assunto. Existem, porém, pesquisas científicas que se baseiam unicamente na pesquisa bibliográfica, procurando referências teóricas publicadas com o objetivo de recolher informações ou conhecimentos prévios sobre o problema a respeito do qual se procura a resposta.

Neste trabalho, foi realizada uma Revisão Sistemática da literatura (RS). Esta pode ser definida como uma abordagem que possui o intuito de identificar, analisar e interpretar pesquisas relevantes, disponíveis e relacionadas com uma questão de pesquisa, um tópico ou um fenômeno de interesse (KHOMH *et al*., 2009). Uma RS pode utilizar pesquisas qualitativas ou quantitativas, sendo esta escolhida a partir do objetivo do estudo do trabalho, como também dos questionamentos (perguntas principais e complementares) presentes na pesquisa. Para este trabalho será utilizada uma pesquisa qualitativa (CRUZES *et al*., 2011) (PATERSON *et.al.*,2011).

Ainda de acordo com esses autores, em uma RS qualitativa são encontrados alguns métodos que podem ser utilizados para a realização da síntese, como: metassíntese, metaetnografia e síntese temática. A escolha do método é baseada nos objetivos atribuídos em uma RS. Neste trabalho, optou-se por realizar a síntese temática, uma vez que este tipo de método permite a síntese dos dados com um foco interpretativo. Além disso, em virtude do elevado número de produções científicas sobre DT, as RS apresentam-se como uma boa oportunidade para a captação, reconhecimento e a síntese das evidências científicas, pois os resultados obtidos são mais abrangentes e não tendenciosos.

A Figura 1 apresenta a estratégia de pesquisa utilizada para a realização desta dissertação. Através dela, percebe-se que foram definidos quatro objetivos e para cada um deles há uma atividade realizada, bem como o resultado obtido.

Figura 1 - Estratégia de Pesquisa



As atividades planejadas são brevemente descritas a seguir:

a) **Revisão informal da literatura:** esta atividade tem como objetivo elucidar os principais conceitos sobre DT para o desenvolvimento desta pesquisa;

b) **Elaboração do Protocolo de Revisão Sistemática:** nessa atividade, um protocolo será elaborado contendo a questão de pesquisa e as complementares, as informações que deverão ser extraídas dos artigos analisados, os critérios de inclusão e exclusão dos artigos e a elaboração da *string* de busca, bem como os motores de busca utilizados;

c) **Revisão Sistemática da literatura:** esta atividade é responsável pela execução do protocolo planejado na etapa anterior, apoiando a identificação dos artigos que obedecerão aos critérios de inclusão preestabelecidos no protocolo da RS;

d) **Aplicação da Síntese Temática:** esta atividade é responsável pela aplicação da síntese temática com base nas informações extraídas dos artigos selecionados. Seu resultado é o mapa temático sobre DT de projeto.

1.5 ESTRUTURA DA DISSERTAÇÃO

Este capítulo apresentou algumas das ideias que motivaram o desenvolvimento desta dissertação, seu objetivo, a questão de pesquisa e a metodologia utilizada para a realização deste trabalho. A organização do texto desta dissertação segue a linha de tempo em que as atividades de pesquisa foram realizadas 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: neste capítulo serão apresentados o conceito de DT, seu processo de identificação, seu gerenciamento, bem como as consequências da dívida em um projeto de software;

Capítulo 3. Revisões Controladas de Literatura em Engenharia de Software: apresenta conceitos sobre revisão e mapeamento sistemático da literatura, e síntese temática;

Capítulo 4. Revisão Controlada da literatura: apresenta as evidências encontradas a partir da aplicação da síntese temática sobre estudos experimentais executados na área de DT;

Capítulo 5. Considerações Finais: discute as considerações finais do trabalho, apresentando uma descrição das principais contribuições alcançadas, limitações e sugestões de trabalhos futuros.

Além disto, este trabalho contém dois apêndices:

APÊNDICE A – Artigos utilizados na revisão controlada de literatura: apresenta a lista de artigos retornados pela revisão controlada de literatura executada neste trabalho e que foi descrita no Capítulo 4.

APÊNDICE B – Planilha contendo a extração dos dados: apresenta a planilha utilizada para a extração dos dados.

Capítulo 2 – Dívida Técnica

Neste capítulo serão apresentados o conceito de DT, seu processo de identificação, seu gerenciamento, bem como as consequências da dívida em um projeto de software.

2.1 INTRODUÇÃO

Em 1992, Cunningham apresentou o conceito de DT. Ele afirmava que a introdução de uma dívida aceleraria o desenvolvimento de *software* no curto prazo. Entretanto, o tempo investido no desenvolvimento de um código mal construído geraria juros sobre esta dívida. O conceito de DT foi definido para se referir a artefatos mal elaborados, assim como a consequência deles no projeto (BROWN et *al.* 2010) (OZKAYA, 2011).

Este capítulo apresenta uma visão geral sobre DT abordando alguns de seus principais conceitos, tipos, indicadores e, por fim, discorre sobre seu gerenciamento.

2.2 DÍVIDA TÉCNICA

De acordo com Zazworka *et al.* (2013), DT descreve a relação entre os retornos a curto prazo, obtidos através do adiamento de alguma atividade relacionada ao processo de desenvolvimento de *software*, e as consequências a longo prazo deste adiamento. A DT é um termo utilizado para descrever situações nas quais a equipe de desenvolvimento adota medidas que impactam na qualidade do *software* com o objetivo de atender a uma necessidade momentânea, tal como a liberação oportuna do *software*.

Essa liberação oportuna, a princípio, pode parecer rentável com relação ao tempo e aumento de produtividade da equipe de desenvolvimento, pois resultará em uma entrega de *software* mais rápida. Entretanto, geralmente está associada à existência de artefatos incompletos, imaturos ou inadequados no ciclo de desenvolvimento do *software*.

O conceito de DT está relacionado a três propriedades principais: valor da dívida, juros e probabilidade dos juros. De acordo com Brown *et al*. (2010), o valor da dívida está relacionado à quantidade de esforço necessária para realizar o pagamento desta dívida (a conclusão de uma tarefa). Já os juros estão relacionados ao aumento do esforço e à redução da produtividade no

andamento das atividades do projeto, pois os desenvolvedores deixarão de executar novas tarefas para ajustar os artefatos que estão incompletos, imaturos ou inadequados. Além disso, existem artefatos que possuem relacionamentos com outros e, possivelmente, estes terão que ser ajustados também. Por fim, a probabilidade dos juros está relacionada às chances de um determinado item de dívida impactar negativamente no projeto, ou seja, a dívida contraída efetivamente trazer uma penalidade para a equipe.

Uma DT pode ser inserida de forma intencional ou não, além disso, a inserção pode ter sido realizada de forma planejada ou não. Baseado nisso, Fowler classificou uma DT como imprudente/prudente e deliberado/inadvertido (FOWLER, 2009). Ela é classificada como Prudente/Deliberado quando a equipe tem consciência de sua inserção e que esta deverá será paga futuramente. Já na Prudente/Inadvertido, a equipe não tem conhecimento acerca da inserção. No entanto, ao perceber a sua existência, executam o pagamento dessa dívida. Na classificação Imprudente/Deliberado, a equipe tem conhecimento sobre a inserção da dívida no projeto, entretanto, não efetua a avaliação do aumento dos juros e dos impactos futuros no projeto. Por fim, tem-se a classificação Imprudente/Inadvertido, na qual a equipe não tem conhecimento acerca da inserção da dívida no projeto e, após descobrir, não utiliza os resultados negativos para análises futuras.

2.3 TIPOS DE DÍVIDA TÉCNICA E SUA IDENTIFICAÇÃO

Alves *et al.* (2016) apresentaram os tipos de dívida que podem ser encontrados em um projeto de *software*: projeto, arquitetura, documentação, teste, código, defeito, requisitos, pessoas, infraestrutura, automação de testes, processo, build, serviço, usabilidade e versionamento. De acordo com Guo e Seaman (2011), uma **dívida de projeto** se refere a itens que podem ser descobertos através da análise do código fonte, bem como identificação de violações das boas práticas de orientação a objetos. Já a **dívida de arquitetura** está relacionada a problemas encontrados tanto na arquitetura de um *software*, como também nos requisitos arquiteturais. A **dívida de teste** se refere a problemas encontrados em atividades relacionadas a teste e podem afetar a qualidade dessas atividades. Já a **dívida de documentação** se refere a problemas encontrados na documentação em projetos de software, tais como: documentação inexistente, inadequada ou incompleta.

De acordo com Bohnet e Dullner (2011), a **dívida de código** diz respeito a problemas encontrados no código fonte e, geralmente, afeta a legibilidade deste tornando mais difícil a sua

manutenção. Já a **dívida de defeito** se refere a defeitos conhecidos, geralmente identificados pelas atividades de teste ou pelo usuário, que por algum motivo tiveram sua correção adiada (SNIPES *et al.*, 2012). A **dívida de infraestrutura** está relacionada a questões de infraestrutura que impedem ou atrasam as atividades de desenvolvimento. Já a **dívida de pessoas** se refere a questões relacionadas a pessoas que podem atrasar a realização de algumas atividades de desenvolvimento como, por exemplo, a contratação tardia de um analista de requisitos para integrar a equipe de um projeto (SEAMAN; SPÍNOLA, 2013).

A **dívida de automação de testes**, considerada como subtipo da dívida de teste, está relacionada ao trabalho envolvido na automação de testes de funcionalidades com o intuito de obter ciclos de desenvolvimentos mais rápidos. Já a **dívida de processo** se refere aos processos ineficientes existentes em projeto de *software* (CODABUX; WILLIAMS, 2013). Morgenthaler *et al.* (2012) afirmam que uma **dívida de *build*** (construção) está associada às questões que tornam a tarefa de compilação mais difícil e, como consequência disso, um processo de compilação demorado. A **dívida de serviço** está relacionada à incompatibilidade entre as características do serviço e os requisitos da aplicação (ALZAGHOUL ; BAHSOON, 2013).

Já problemas relacionados à usabilidade são denominados de **dívida de usabilidade**. E, por fim, tem-se a **dívida de versionamento,** que se refere a problemas de versionamento do código fonte e a **dívida de requisitos**, que está relacionada às decisões sobre quais requisitos devem ser implementados ou como serão (ZAZWORKA *et al*. 2013) (GREENING, 2013).

O processo de identificação de uma DT pode ser realizado de forma manual ou automatizado. Para Zazworka *et al.* (2013), a abordagem manual, provavelmente, consumirá mais tempo durante o processo de identificação quando comparada à automatizada. No entanto, pode ser mais precisa, pois é mais propensa a identificar DT significativa, enquanto análises automatizadas podem revelar muitas anomalias que se revelam pouco importantes. Outra vantagem, considerando o processo manual, é que os desenvolvedores ou as partes interessadas podem fornecer informações contextuais importantes relacionadas a cada instância de DT como, por exemplo, estimativas de esforço, impacto, justificativa de decisão e etc.

Os diferentes tipos de DT podem ser descobertos através do uso de indicadores. Alves *et al.* (2016) afirmam que “os indicadores de DT permitem a descoberta de itens da dívida ao analisar os diferentes artefatos criados durante o desenvolvimento de um projeto de software”. A Tabela 1, extraída de Alves *et al.* (2016), apresenta alguns indicadores da presença de dívida conhecidos, relacionando-os aos seus respectivos tipos.

Tabela 1 - Indicadores de Dívida Técnica

Tipos de Dívida	Indicadores
Arquitetura	Violação de Modularidade
	Problemas na Arquitetura do Software
	Betweenness na Arquitetura do Software
	Augmented Constraint Network (CAN)
	Pairwise-Dependecy Relation (PWDR)
	Index of Package Changing Impact (IPCI)
	Index of Package Goal Focus (IPGF)
Arquitetura/ Projeto e Construção	Dependências Estruturais
Construção (*Build*)	Problemas de Construção (*build issues*)
Código	Código sem padrões
	Algoritmo Lento
	Multithread Correctness
Projeto e Código	*Automatic Static Analysis (ASA) Issues*
	Code Metrics (not specified)
	Code Smells
Projeto	*Grime*
	Problemas de Projeto de Software
	Low External/ Internal Quality
	Afferent/ Efferent Couplings (AC/EC)
	Depth of Inheritance Tree (DIT)
	Referential Integrity Constraints (RICs)
Defeito e Teste	Defeitos Conhecidos, mas não corrigidos
Documentação	Comentários insuficientes no código
	Falta de documentação
	Comentários (*hack, fixme, is problematic, ...*)
	Problemas na documentação
Requisitos	Lista de pendências em requisitos
Serviço	Serviços web que não atendem as restrições de qualidade do projeto
Automação de Teste	Falta de Teste Automatizado

Tipos de Dívida	Indicadores
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
Versionamento	Uso desnecessário de *Forks*

Fonte: Alves *et al.* (2016).

2.4 GERENCIAMENTO DE DÍVIDA TÉCNICA

Durante o ciclo de desenvolvimento de *software*, equipes de desenvolvimento adotam decisões devido à existência de algumas necessidades momentâneas, tais como cumprimento de prazo. Estas decisões normalmente impactam na qualidade do *software* e estão relacionadas à inserção de DT no projeto. Entretanto, em um determinado momento, a DT deverá ser paga uma vez que sua presença poderá acarretar em problemas técnicos e financeiros ocasionando, por exemplo, aumento nos custos com relação à manutenção e evolução do *software* (NORD *et al*., 2012). O cálculo resultante dessa dívida está associado ao valor a ser pago acrescido dos juros que incidem sobre ela.

É necessário que haja o gerenciamento da DT. Espera-se que com ele seja possível obter informações para auxiliar os gestores quanto ao momento mais adequado para pagamento de uma dívida e o valor que esta custará. A gestão de um item de dívida está relacionada à redução do impacto negativo deste no projeto (BROWN *et al.*, 2010).

De acordo com Alves *et al.* (2016), a utilização de estratégias que possibilitem o gerenciamento considerando as mudanças que ocorrem na produtividade durante o processo de desenvolvimento de *software* é um caminho para monitorar a dívida no projeto. Empresas de *software* que não administram os itens de dívida presentes em seus projetos, com o passar do tempo, percebem que eles ocasionam mais esforço com relação ao pagamento de juros e, consequentemente, em um *software* que será mais difícil de manter e que possui baixa qualidade.

De acordo com Fairley (1994), métricas relacionadas à qualidade de *software* também podem ser utilizadas para realizar a medição de uma dívida existente em um projeto de *software*,

tais como: acoplamento, coesão, complexidade e profundidade de decomposição. Além disso, o tempo e esforço investidos pelos engenheiros de *software* para realizar mudanças também podem ser considerados como um indicador da presença da dívida. Por fim, a medição de uma dívida também pode ser realizada através do esforço que será investido para realizar seu pagamento.

De acordo com Brown *et al.* (2010), para realizar o pagamento da dívida, uma outra estratégia que pode ser utilizada é priorizar a eliminação dos itens centrais da dívida. Ou seja, itens que possuem impacto significativo no futuro do projeto.

Em uma outra iniciativa, Ribeiro *et al.* (2017) apresentam uma ferramenta, denominada de TD Manager, para a realização do gerenciamento de uma dívida com base em critérios de decisão. A ferramenta apoia a avaliação de itens da dívida presentes no projeto e, desta forma, auxilia no processo de tomada de decisão com relação ao pagamento deles. A tabela 2 apresenta os critérios, organizado pelos autores, que apoiam esta tomada de decisão.

Os critérios de decisão, estabelecidos pelos autores, facilitam o processo de gerenciamento dos itens de dívida. Pois, os desenvolvedores identificarão o que deverá ser priorizado (pago), verificará a qual categoria pertence e, por fim, aplicará os critérios que estão associados a esta para a realização da gestão dos itens de DT. Ainda de acordo com a Tabela 2, percebe-se que os itens de DT podem ser analisados com base na sua gravidade, ou seja, no impacto que causará. Tem-se também que itens de dívida percebidos pelo usuário ou associados a partes muito utilizadas pelo sistema, bem como itens que causam maior impacto para o projeto deverão ser pagos. Por fim, em projetos críticos, a dívida deverá ser eliminada o quanto antes.

Tabela 2 - Critérios de Decisão

Categoria	**Critério**	**Descrição**	**Estudos**
Natureza	Gravidade da Dívida	Dívidas com nível de gravidade alta devem ser pagas. Uma análise da gravidade da DT é realizada através da investigação dos recursos afetados por ela. Em seguida, determina-se a gravidade da dívida como baixo, médio ou alto.	(SNIPES *et al.*, 2012) (CODABUX; WILLIAMS, 2013).
	Proximidade com atividade de desenvolvimento ou manutenção atual no projeto	Itens de dívida que estão associados com o desenvolvimento ou alteração de uma funcionalidade devem ser pagos. Se a dívida estiver ligada a um componente que será alterado ou está em desenvolvimento, deve-se aproveitar para efetuar o pagamento da dívida.	(GUO; SPÍNOLA; SEAMAN, 2014).

Categoria	Critério	Descrição	Estudos
Cliente	Visibilidade	Se o item de dívida puder ser percebido pelo usuário, então ele deve ser pago. Neste critério, considera-se apenas a questão ou não da percepção da existência da dívida pelo cliente. O impacto que ela pode trazer para suas atividades é considerado em outro critério.	(LIM, 2012)
	Análise de quando a parte refatorada será utilizada	Itens de dívida que estão em partes muito utilizadas do sistema devem ser pagos, caso contrário, o pagamento pode ser adiado.	(SEAMAN *et al.*, 2012) (MAMUN; BERGER; HANSSON, 2014)
	Impacto no cliente	Itens de dívida que impactam diretamente no cliente devem ser priorizados.	(SNIPES *et al.*, 2012) (CODABUX ; WILLIAMS, 2013)
Esforço	Impacto da dívida no projeto	Itens de dívida que oferecem maior impacto para o projeto devem ser eliminados	SNIPES *et al.*, 2012) (MARINESCU, 2012) (ALMAM, 2012) (LIM, 2012) (LETOUZEY ; ILKIEWICZ, 2012) (CODABUX ; WILLIAMS, 2013) (HOLVITIE ; LEPPANEN, 2013) (GUO; SPÍNOLA; SEAMAN, 2014)
	Custo/benefício	Verifica os benefícios e o custo da dívida. Se os juros acumulados (custo) forem menores do que o custo de pagar a dívida (benefício), então não se deve pagá-la agora.	(ALMAM, 2012). (BUSCHMANN, 2011) (SNIPES *et al.*, 2012). (SEAMAN *et al.*, 2011). (SEAMAN *et al.*, 2012). (OZKAYA *et al.*, 2010). (ZAZWORKA; SEAMAN; SHULL, 2011). (CODABUX; WILLIAMS, 2013).
	Esforço para implementar a proposta de correção	O esforço necessário para pagar a dívida é medido e analisado em conjunto com o tempo e recursos disponíveis. Itens de dívida que exigem menor esforço para serem pagos devem ser priorizados.	(SNIPES et al., 2012). (HOLVITIE; LEPPANEN, 2013). (CODABUX; WILLIAMS, 2013).
Projeto	Natureza do Projeto	Em projetos críticos, a dívida deve ser eliminada mais rapidamente. Verificar a natureza do projeto para decidir sobre o pagamento da dívida.	(DAVIS, 2013). (ALMAM, 2012).

Fonte: Ribeiro *et al.* (2017).

Ainda de acordo com Ribeiro *et al.* (2017), o uso da ferramenta possibilita uma maior rapidez e produtividade com relação ao processo de avaliação dos itens da dívida, bem como a eficácia da avaliação destes itens, tornando assim mais fácil a decisão sobre quando eliminá-los do projeto.

2.5 CONSIDERAÇÕES FINAIS

DT descreve situações nas quais a equipe de desenvolvimento adota medidas que impactam na qualidade interna do *software* com o objetivo de atender a uma necessidade momentânea. A inserção de uma dívida normalmente está associada à criação de dificuldades para a evolução do *software*. Sendo assim, é necessário que sejam utilizadas estratégias de gerenciamento que possibilitem identificar, monitorar e avaliar estes itens de dívida.

Em virtude do crescimento das pesquisas sobre DT, diversos trabalhos têm sido realizados, a partir de estudos primários, com o intuito de agrupar evidências. Estes trabalhos, denominados de estudos secundários, permitem fornecer uma visão mais abrangente sobre uma determinada área de pesquisa ou tema, pois são utilizados para estabelecer comparações, generalizações sistemáticas ou mapeamento das investigações individuais.

No próximo capítulo serão abordados conceitos relacionados a revisão sistemática, mapeamento sistemático da literatura e síntese temática, utilizados no desenvolvimento deste trabalho.

Capítulo 3 – Revisões Controladas da Literatura em Engenharia de Software

Neste capítulo serão apresentados conceitos relacionados a revisão sistemática, mapeamento sistemático da literatura e síntese temática.

3.1 INTRODUÇÃO

De acordo com Wohlin *et al.* (2012), uma pesquisa pode ser classificada como qualitativa ou quantitativa. Uma pesquisa quantitativa considera que tudo pode ser quantificável seja em números e informações que possam ser classificadas para serem analisadas. Ele utiliza recursos e técnicas estatísticas, como percentagem, desvio-padrão, média e etc. Além disso, normalmente envolve a formulação de hipóteses e classificação da relação existente entre as variáveis.

Já uma pesquisa qualitativa, o foco desta dissertação, considera que há uma relação dinâmica entre o mundo real e o sujeito e dificilmente pode ser traduzida em números. Além disso, possui um caráter exploratório, descritivo e indutivo e para realizar a iteração entre o observador e o mundo, comumente, pode utilizar representações, como a análise de estudos primários ou mesmo secundários, estudos de caso, entrevistas, entre outros.

A engenharia de *software* baseada em evidências (ESBE) possui dois tipos principais de investigação (YAMASHITA, 2013): estudos primários e secundários. Os estudos primários são utilizados para avaliar uma hipótese formulada pelo pesquisador, sendo esta testada sobre condições estabelecidas em um controle metodológico observacional ou experimental (CRUZES, 2007). Já os estudos secundários são utilizados para estabelecer comparações, generalizações sistemáticas ou mapeamento das investigações individuais a partir de um conjunto de estudos primários selecionados de forma controlada (YAMASHITA, 2013) (CRUZES, 2007).

Cruzes (2007) apresenta alguns métodos para síntese de evidências qualitativas e quantitativas que podem ser utilizados na engenharia de *software*, são eles: sumário narrativo, *grounded theory*, codificação ou *coding*, análise qualitativa e comparativa, revisões sistemáticas (RS), mapeamento sistemático (MS), metassíntese ou Metaestudo, meta-análise, contagem de votos, síntese temática, SecESE, etnografia e metaetnografia.

O método sumário narrativo é responsável pela descrição narrativa e ordenação de evidências primárias com comentários e interpretação. Além disso, apresenta como vantagens: procedimentos flexíveis, manipulam grandes bases e diversos tipos de evidência. No entanto, possui a seguinte desvantagem: falta de transparência na seleção e também nos estágios posteriores do processo. Já o *grounded theory* é responsável por identificar padrões e inter-relacionamentos em dados primários e apresenta como vantagens: busca por explanações e/ou teorias generalizadas. Entretanto, possui a falta de transparência como desvantagem.

O método codificação ou *coding* é utilizado nas etapas de análise e interpretação dos resultados. Este método apresenta como vantagem um procedimento analítico no qual os dados qualitativos são divididos, conceitualizados e integrados para criar uma teoria. Por outro lado, ele é caracterizado pela falta de uma definição do grau de formalidade na estrutura dos dados de codificação. Já a análise qualitativa e comparativa é um método de análise booleana das condições necessárias e suficientes para que os resultados particulares sejam observados, baseadas na presença/ausência das variáveis e dos resultados em cada estudo primário. Ele apresenta como vantagens ser sistemático e transparente e poder incorporar evidências qualitativas e quantitativas. No entanto, possui como desvantagens: aspectos não interpretativos dos dados qualitativos, como também, o foco na casualidade.

RS são caracterizadas por uma busca detalhada com base nos estudos primários, a partir de uma questão de pesquisa específica. Além disso, realiza a seleção dos estudos através de critérios de elegibilidade reproduzíveis, avaliação crítica da qualidade dos estudos, e síntese dos resultados de acordo com um método predeterminado e explícito. Este método apresenta como vantagem a captura precisa da evidência e, como desvantagem, na maioria das vezes é um método dispendioso e custoso.

O MS, considerado uma variação da RS, é responsável por realizar uma revisão ampla dos estudos primários sobre um determinado tema e, a partir desta, constrói um mapa sistemático. O objetivo principal deste método não é sintetizar novas evidências, mas identificar lacunas e áreas onde novos estudos primários possam ser conduzidos. Sendo assim, este método apresenta como vantagem uma visão geral da área pesquisada. Como desvantagem, também é considerado dispendioso. Entretanto, menos custoso quando comparado com as RSs. Já o Metassíntese ou Metaestudo é uma síntese interpretativa de achados qualitativos, provenientes de estudos fenomenológicos, etnográficos, da teoria fundamentada nos dados e outros. Este

método apresenta como vantagem uma nova interpretação dos resultados encontrados e, como desvantagem, não há consenso quanto a terminologia para descrever o processo.

A meta-análise geralmente é a etapa final de uma RS, sendo caracterizada por uma combinação estatística dos dados provenientes dos estudos com o intuito de criar uma estimativa sumária dos efeitos (quaisquer medidas de associação entre a exposição e o resultado). A principal dificuldade no uso desse método é o fato de que, para ser aplicado, os dados extraídos devem satisfazer vários requisitos, tal como um alto nível de homogeneidade entre os estudos. Já contagem de votos é um método onde os resultados obtidos para cada teste das hipóteses são caracterizados como significativamente positivos, negativos ou sem efeito significativo. Após isso, serão contabilizadas a quantidade de votos e o resultado encontrado não poderá exceder a razão entre votos positivos e negativos quando comparado com um limite predeterminado. Apresenta como vantagem: mais simples do que meta-análise e pode ser aplicado tanto para estudos qualitativos quanto quantitativos. E, como desvantagens: não leva em consideração a qualidade dos estudos, bem como não depende dos valores reais do tamanho do efeito e de medidas comparáveis.

O método síntese temática utiliza os princípios da análise temática para a identificação, análise e identificação de padrões, a partir de dados observados e provenientes de uma pesquisa qualitativa primária, apresentando como saída um modelo de alto nível. O método apresenta como vantagem a tradução de conceitos entre os estudos, através de um processo claro e transparente, assim como uma visão gráfica do mapeamento facilitando a compreensão e a confiabilidade dos resultados obtidos e, como desvantagem, nem sempre os estudos primários abordam as questões da revisão sistemática que está sendo realizada. Já SecESE é uma abordagem que utiliza a contagem de votos e formalização de atividades em uma análise secundária em ESE. Ele apresenta como vantagem a formalização da análise pela execução de um processo repetível, utilizando-se dados qualitativos e quantitativos para a análise.

O método Etnografia se baseia na observação e investigação, permitindo assim, o aumento da percepção com relação aos costumes e práticas adotados por determinados grupos que estão sendo analisados. Ele apresenta como vantagem: coleta de dados *in loco*, permitindo ao pesquisador escolher a técnica de análise que será utilizada. Por outro lado, como desvantagem: requer permissão do grupo que está sendo analisado para que a observação possa ser feita, bem como exige imparcialidade do pesquisador para não haver interferência nos resultados. Já metaetnografia é utilizada para uma integração interpretativa de resultados

qualitativos provenientes de uma RS, MS ou um conjunto de estudos etnográficos. Ela apresenta como vantagem: auxilia na criação de teorias mais abrangentes e generalizáveis e, desta forma, proporciona um nível mais elevado de análise, preservando as propriedades de interpretação dos dados primários. E, como desvantagem, existe a possibilidade da perda de contexto explicativo.

De acordo com Cruzes (2007), não existe um método considerado perfeito, todos apresentam problemas e vantagens na sua aplicação. Sendo assim, a escolha do método para realizar uma síntese deverá ser baseada no objetivo do trabalho. Sendo assim, os métodos para síntese de evidências que serão analisados em maiores detalhes por serem utilizados neste trabalho serão a revisão sistemática (seção 3.3), o mapeamento sistemático da literatura (seção 3.4) e, por fim, a síntese temática (seção 3.5).

3.2 TRABALHOS RELACIONADOS

Seis estudos secundários já foram realizados com o intuito de reunir evidências a partir de trabalhos primários, são eles: Tom *et al.* (2012), Villar e Matalonga (2013), Tom *et al.* (2013), Ampatzoglou *et al.* (2015), Li *et al.* (2015) e Alves *et al.* (2016). Em seu estudo, Tom *et al.* (2012) apresentaram o primeiro estudo secundário sobre DT que possuía como questões de pesquisa: Quais os elementos que compõe a DT? e Por que a DT surge? Estes questionamentos possibilitaram uma visão holística da DT acerca das vantagens em permitir o acúmulo da dívida, bem como o seu comportamento dentro de uma aplicação.

Já Villar e Matalonga (2013) realizaram um estudo apresentando como questões de pesquisa: Quais são as definições atuais de Dívida Técnica e Dívida de Desenvolvimento? Quais atividades de pesquisa têm sido realizadas na área? Como a área tem evoluído ao longo do tempo? Quem são os principais pesquisadores na área? Os autores apresentaram como resultados 11 definições sobre DT, o número de publicações ao longo dos anos, os autores mais atuantes com relação ao número de publicações nesta área e, por fim, apresentaram quatro categorias que foram utilizadas para realizar o agrupamento entre os artigos analisados: trabalhos genéricos sobre DT, dívida de código, outros tipos de dívida e stakeholders.

O trabalho realizado por Tom *et al.* (2013) apresentou um estudo de caso exploratório e teve como objetivo a consolidação acerca da compreensão da natureza da DT, bem como suas implicações. Os resultados apresentados através deste estudo foram uma taxonomia que

descreve e abrange diferentes tipos de DT, bem como um quadro teórico que compreende um conjunto de dimensões da DT, atributos precedentes e resultados.

Ampatzoglou *et al.* (2015) realizaram uma revisão sistemática sobre DT levando em consideração o aspecto financeiro e apresentaram como resultados: as abordagens financeiras que têm sido frequentemente utilizadas no gerenciamento da dívida que são *options*, gerenciamento de portfólio, análise de custo/benefício e análise baseada em valor. Além disso, identificaram que os termos financeiros valor e juros são os mais utilizados na área.

Já Li *et al.* (2015) apresentou um estudo que possuía o objetivo de realizar uma análise sobre DT e o seu gerenciamento. Os resultados apresentados foram a identificação de: 10 tipos de dívida, 8 atividades de gerenciamento e 29 ferramentas que realizam o gerenciamento de DT.

Por fim, em 2016, Alves *et al*. apresentaram um estudo que tinha como objetivo a caracterização dos tipos de DT, a identificação dos identificadores para detecção de dívidas, a identificação das estratégias de gestão, bem como a compreensão do nível de maturidade e quais técnicas de visualização podem ser utilizadas para auxiliar na identificação e na gestão da DT. Os resultados apresentados foram uma taxonomia dos tipos de DT, uma lista de indicadores que auxilia no processo de descoberta de um item de dívida e as estratégias de gestão existentes.

Apesar desses estudos terem permitido uma visão abrangente sobre as pesquisas que têm sido realizadas sobre DT, nenhum deles realizou uma análise específica sobre os resultados dos estudos experimentais realizados na área. Este é o objetivo principal do protocolo da revisão sistemática executado neste trabalho e que será apresentado na próxima seção.

3.3 REVISÃO SISTEMÁTICA

Uma RS envolve o planejamento de tarefas e a sistematização de todo o processo de busca por evidências originadas da literatura técnica. Normalmente, é um trabalho detalhista e que exige um alto nível de esforço. Outras características que as definem são (PRATES, 2015):

a) Tem-se a definição de um protocolo de avaliação, que conterá a questão de pesquisa a ser investigada e os métodos que serão utilizados para realizar a revisão;

b) São baseadas em uma estratégia de busca como parte do protocolo;

c) A estratégia de busca é documentada para que interessados possam replicar a revisão realizada;

d) Existem critérios de inclusão e exclusão que serão utilizados com o intuito de selecionar ou descartar estudos primários;

e) Podem ser definidos critérios de qualidade para avaliar os estudos que farão parte da revisão.

De acordo com Kitchenham (2004), a RS é uma abordagem que busca identificar, avaliar e interpretar as pesquisas relevantes e disponíveis que estão relacionadas com uma determinada questão de pesquisa, um tópico ou, até mesmo, um fenômeno de interesse, promovendo assim, um estudo detalhado e abrangente sobre um determinado assunto. Ressalta-se ainda que o planejamento e a execução de uma RS deverá ser um processo que assegure a validade dos resultados. Sendo assim, a qualidade dos estudos primários deverá ser levada em consideração, pois interfere na qualidade do estudo secundário.

Ainda de acordo com Kitchenham (2004), a necessidade para se realizar uma RS da literatura pode ser descrita através da sumarização de evidências existentes sobre um determinado fenômeno, pela identificação de lacunas existentes na pesquisa atual, pelo fornecimento de um arcabouço para posicionar novas pesquisas e pelo apoio para a geração de novas hipóteses.

Uma RS possui três fases: planejamento, execução e documentação da revisão. Na fase de planejamento, deve-se definir o objetivo, bem como os passos que serão seguidos para a realização do estudo. Assim, faz-se necessário identificar a necessidade da revisão, definir os objetivos da pesquisa, o protocolo de revisão e validá-lo (KITCHENHAM, 2004) (DYBA *et al.*, 2007). Ainda durante o planejamento, deve-se formular as questões de pesquisas e estas deverão ser elaboradas de acordo com os objetivos da RS, tendo como função direcionar a execução das demais atividades. Além disso, as questões de pesquisa servirão como guia para a revisão, pois são responsáveis por definirem os estudos que serão incluídos e quais estratégias serão utilizadas para identificá-los, bem como os dados que devem ser coletados (CRUZES *et al.*, 2011).

Na fase de execução, deve-se realizar o planejamento descrito no protocolo, identificando e selecionando os estudos primários que serão analisados. Para isso, devem-se identificar as pesquisas relevantes, selecionar os estudos primários de acordo com os critérios

de inclusão e exclusão estabelecidos, avaliar a qualidade dos estudos de acordo com os critérios estabelecidos, extrair informações dos estudos primários selecionados e analisar os dados dos estudos selecionados. Durante esse processo, deve-se elaborar o relatório da revisão que contém a análise das informações dos estudos primários. Por fim, tem-se a documentação da revisão, na qual deve-se documentar a revisão, gerando assim, relatórios que conterão a síntese das informações dos estudos primários.

3.4 MAPEAMENTO SISTEMÁTICO

De acordo com Kitchenham (2004), o MS pode ser definido como revisões mais amplas dos estudos primários com o intuito de identificar quais evidências estão disponíveis, bem como as lacunas existentes com relação aos estudos primários e, com isso, servir de guia para revisões sistemáticas futuras, assim como identificar quais são as áreas em que estudos primários precisam ser aprofundados ou realizados. Portanto, o MS fornece uma visão geral de uma determinada área de pesquisa. Ele permite identificar o número e os tipos de pesquisas realizadas, os resultados disponíveis, bem como a frequência de publicações ao longo dos anos e, com isso, obtêm-se as tendências sobre uma determinada área de pesquisa (PETERSEN *et al.*, 2008).

O processo utilizado para realizar um MS é semelhante ao definido para uma RS. Inicialmente, deve-se elaborar um protocolo, que deverá conter as questões de pesquisas que serão investigadas. Após a validação deste, o MS será realizado e, apresentará como resultado, mapas contendo as evidências encontradas (KITCHENHAM *et al.*, 2007). De acordo com Dyba *et al*. (2005), existem algumas características em comum entre as revisões e os mapeamentos sistemáticos:

a) Elaboração de um protocolo formal de pesquisa;

b) Busca abrangente por estudos primários;

c) Avaliação de qualidade dos estudos considerados;

d) Identificação dos dados necessários para responder às questões de pesquisa;

e) Extração dos dados;

f) Resumo e síntese dos resultados dos estudos;

g) Interpretação dos resultados, auxiliando a analisar a sua aplicabilidade;

h) Escrita do relatório.

Por outro lado, ainda de acordo com Dyba *et al*. (2005), existem algumas diferenças entre eles que podem ser destacadas:

a) Normalmente, o MS possui múltiplas questões de pesquisa e estas possuem caráter exploratório, uma vez que são mais abrangentes;

b) Em uma RS, as questões de pesquisa são mais focadas do que as definidas para um MS. Isso faz com que, normalmente, tenha-se um número maior de estudos a serem considerados durante a realização de um MS;

c) O processo de extração de dados para um MS é mais amplo do que em uma RS, uma vez que ele possui um caráter mais exploratório.

3.5 SÍNTESE TEMÁTICA

A síntese temática é utilizada para apoiar a análise e identificação de padrões a partir de dados provenientes de uma pesquisa qualitativa primária, utilizando-se dos princípios da análise temática (THOMAS; HARDEN, 2008). Assim, uma síntese temática é uma abordagem utilizada para referenciar um método de análise de pesquisa primária, utilizado em revisões sistemáticas, gerando assim, um estudo secundário sobre uma determinada área de pesquisa. Através dela, novos resultados são agregados aos estudos, auxiliando na identificação de teorias e possíveis generalizações (CRUZES; DYBA, 2011).

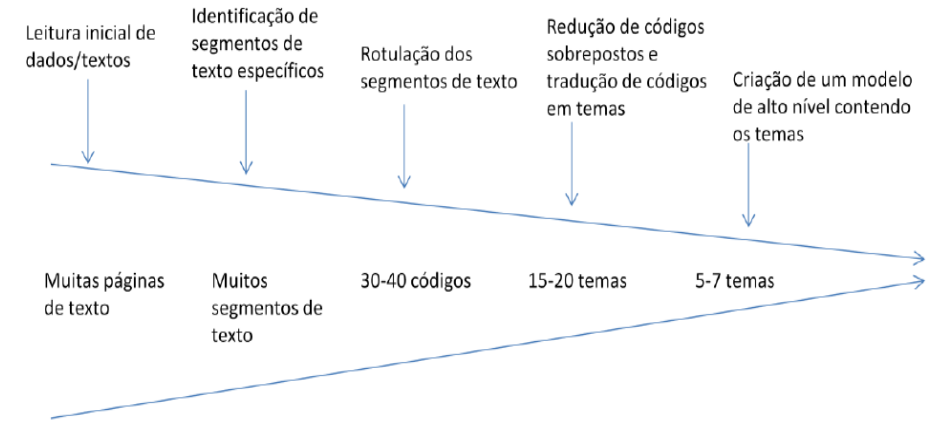
Segundo Cruzes e Duba (2011), a síntese é utilizada para sumarizar, integrar, combinar e comparar os resultados obtidos, através de diferentes estudos primários, acerca das questões de pesquisas investigadas. Além disso, ela pode ser utilizada para apoiar a construção de um mapa temático, uma taxonomia ou, até mesmo, uma teoria relacionada a uma questão de pesquisa que está sendo investigada. O mapa temático pode ser definido como um modelo, que deve conter temas de mais alto nível, subtemas e seus respectivos códigos, permitindo assim uma visualização dos resultados de forma gráfica (THOMAS; HARDEN, 2008) (CRUZES; DYBA, 2011).

De acordo com Noblit e Hare (1988), uma síntese pode ser classificada como integrativa ou interpretativa. Em uma síntese integrativa, tem-se o objetivo de analisar a combinação de

um conjunto de dados, provenientes dos estudos primários semelhantes, com base nas intervenções e variáveis quantitativas. Já a síntese interpretativa tem como objetivo avaliar estudos primários, podendo estes ser heterogêneos, fornecendo ao final explicações interpretativas.

Cruzes e Dyba (2011) definiram em seu trabalho cinco passos que devem ser realizados para uma síntese temática na área de ES, permitindo assim, que as sínteses sejam realizadas de forma mais sistemática. A figura 2 ilustra os cinco passos propostos por Cruzes e Dyba: leitura inicial de dados/texto, identificação de segmentos de texto específicos, rotulação dos segmentos de texto, redução de códigos sobrepostos e tradução de códigos em temas e a criação de um modelo de alto nível contendo os temas.

Figura 2 - Cinco passos para Síntese Temática



Fonte Cruzes e Dyba (2011).

De acordo com a Figura 2, o primeiro passo é a leitura inicial de dados/textos. Em seguida, tem-se a identificação de segmentos de texto específicos responsável pelos segmentos de texto extraídos. Já a rotulação dos segmentos de texto resultam em códigos livres. Enquanto que a redução de código sobrepostos e tradução dos códigos em temas são responsáveis por gerar temas. Por fim, tem-se a criação de um modelo de alto nível contendo os temas já tratados que representam a amostra analisada.

Ainda de acordo com os autores, uma tabela é apresentada contendo os passos sistemáticos citados individualmente e que deverão ser realizados, como também, um *checklist* contendo os principais questionamentos com relação ao passo em questão (Tabela 3). O *checklist* tem como função apoiar a execução do método, aumentando assim a sua confiabilidade e permitindo que outros pesquisadores possam repetir a execução do método para o mesmo trabalho.

Tabela 3 - Etapas do Método de Síntese Temática e Checklist

Etapa	**Descrição**	***Checklist***
Extração de Dados	Extrair dados de estudos primários, incluindo informação bibliográfica, objetivos, contexto e resultados.	1. Todos os artigos foram lidos para facilitar a imersão no contexto e nos dados? 2. Foram identificados segmentos de texto que ajudam a identificar os objetivos do artigo? 3. Detalhes da publicação, descrições de contexto e resultados foram extraídos dos artigos selecionados? 4. Outro pesquisador verificou a extração?
Codificação de Dados	Identificar e codificar conceitos interessantes, categorias, descobertas e resultados de uma forma sistemática em todo o conjunto de dados.	5. Os segmentos importantes do texto como conceitos, categorias, descobertas e resultados foram rotulados e codificados? 6. A codificação foi feita em um nível apropriado para as questões de pesquisa? 7. Lista de códigos iniciais com definições e frequências foram criadas e controladas por outro pesquisador? 8. Verificações de consistência ou confiabilidade entre pesquisadores foram realizados para estabelecer a credibilidade da codificação? 9. Existem ligações claras e evidentes entre o texto e os códigos?
Tradução de Códigos em Temas	Traduzir códigos em temas, subtemas e temas mais abrangentes.	10. Os temas foram criados a partir de uma visão geral, inclusiva e compreensiva dos códigos de todos os artigos? 11. Códigos sobrepostos foram reduzidos e os códigos restantes foram traduzidos em temas? 12. Os temas foram verificados uns contra os outros e foi feito batimento dos mesmos com os dados dos artigos originais? 13. Os temas são coerentes, consistentes e distintos?
Criação do Modelo de Alto Nível	Explorar relações entre temas e criar um modelo com os temas mais abrangentes.	14. Existem descobertas interpretadas que fazem mais sentido do que se fosse somente parafraseada ou descrita? 15. Os temas e as relações entre os temas foram comparados e verificados com as conclusões dos estudos primários? 16. Existe uma descrição clara das relações entre os temas mais abrangentes? 17. Foi criado um modelo para apresentar as relações existentes entre os temas mais abrangentes?
Avaliação da Confiabilidade da Síntese	Avaliar a confiabilidade das interpretações que antecederam a síntese temática.	18. A suposição sobre uma abordagem especifica para a síntese temática foram claramente explicadas? 19. Existe um bom ajuste entre o que é reivindicado e o que a evidência mostra? 20. A linguagem e os conceitos utilizados na síntese são consistentes? 21. As questões de pesquisa foram respondidas com base na evidência da síntese temática?

Fonte Cruzes e Dyba (2011).

3.5.1 Extração de dados

A etapa extração de dados é responsável por obter dados provenientes dos estudos primários de forma explícita, coerente e de acordo com a estratégia de extração definida. Para a extração, deve-se realizar uma leitura completa dos artigos pré-selecionados, devido à necessidade de se aprofundar no conteúdo. Além disso, durante esta fase, talvez seja necessário que os pesquisadores revisem o protocolo definido para a revisão sistemática (CRUZES; DYBA, 2011).

Cruzes e Dyba (2011) apresentam um modelo para a extração dos dados que está representado na Figura 3. Através dela, percebe-se que existem três dimensões no processo de extração de dados: publicação, contexto e resultados. Para cada dimensão tem-se um conjunto de dados relevantes que deverão ser extraídos e, além disso, cada publicação está associada a um contexto específico e, este, poderá estar associado a um ou mais resultados apresentados pelos artigos.

Figura 3 - Modelo para Extração de Dados

Fonte: Cruzes e Dyba (2011).

3.5.2 Codificação de dados

De acordo com o modelo proposto por Cruzes e Dyba (2011), o próximo passo para a realização de uma síntese temática é a codificação de dados. Os códigos devem ser vistos como rótulos descritivos e aplicados aos segmentos de texto extraídos para cada estudo, obtendo assim um significado específico. Para isso, inicialmente, todos os segmentos de texto que simplificam a mesma ideia deverão ser extraídos, permitindo que sejam organizados e agrupados de forma semelhante em categorias.

No entanto, o processo de codificação não é simplesmente a tarefa de atribuir rótulo, pois envolve análise e organização dos dados contidos em cada estudo primário da RS, bem como a exigência de conhecimento aprofundado do contexto do estudo para auxiliar no processo de identificação do objeto relevante para um determinado assunto.

Existem três abordagens que podem ser utilizadas para a codificação: dedutiva, indutiva e integrada. A dedutiva gera uma lista inicial contendo códigos previamente formatados, provenientes de teorias, questões de pesquisa, hipóteses, áreas problemáticas e variáveis chave. Essa lista, contendo os códigos, auxiliará os pesquisadores na integração de conceitos difundidos na literatura.

Já na indutiva, faz-se necessário uma análise dos dados para que os códigos apareçam para o pesquisador e sejam atribuídos. Ou seja, não existe uma lista prévia dos possíveis códigos que serão trabalhados para um determinado conceito. Por esse motivo, os códigos são refinados constantemente, mediante a inserção e análise de novos dados. Além disso, o pesquisador deverá verificar se a atribuição a um determinado código foi realizada de forma correta e, para isso, este deverá comparar os segmentos de texto extraídos e, após isso, avaliar se estes refletem o mesmo conceito.

Por fim, a abordagem integrada prevê a criação de um esquema geral de códigos sem conteúdo específico e estes deverão ser trabalhados de forma indutiva.

Cruzes e Dyba (2011) alertam para alguns problemas que podem surgir durante a fase de codificação:

a) Codificação ser realizada em um nível muito geral;

b) Identificar o que você quer ver e não o que o texto está dizendo;

c) Codificação fora do contexto.

Os autores afirmam que para evitar esses problemas, deve-se rotular e codificar os segmentos de texto, tais como: conceitos, descobertas, categorias e resultados, durante a execução do método.

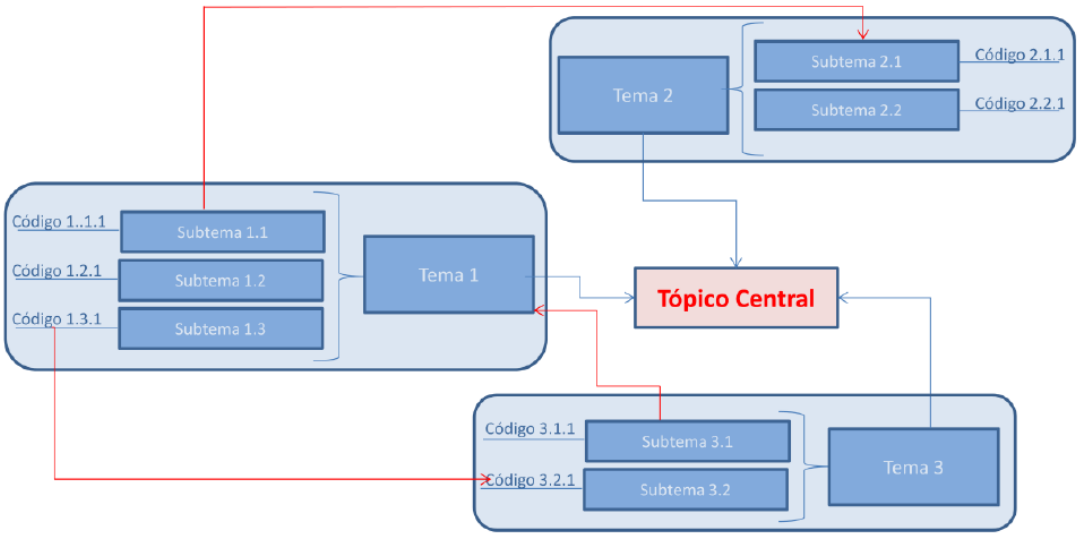
3.5.3 Tradução de Códigos em Temas

Cruzes e Dyba (2011) referem-se a uma combinação dos códigos, identificados na etapa anterior, com o intuito de gerar temas mais abrangentes com relação ao assunto pesquisado. Para isso, esses códigos deverão ser analisados e, com isso, novos códigos, temas, agrupamentos e reclassificações são criados com o intuito de representar os resultados encontrados. Os autores ressaltam que esse passo só será finalizado quando o pesquisador testar todas as possibilidades de temas.

Cruzes e Dyba (2011) recomendam o uso de representações visuais para auxiliar no entendimento e classificação dos diferentes códigos em temas. Alguns exemplos dessas representações são mapas mentais e mapas temáticos. Eles auxiliam os pesquisadores no entendimento dos resultados obtidos com a síntese, assim como a compreensão deles mediante o assunto.

A Figura 4 ilustra um exemplo de mapa temático, proposto pelos autores. É possível observar a existência de um tópico central, sendo este responsável por nortear as pesquisas em busca das descobertas. Além disso, as relações existentes entre os temas são agrupadas e denominadas de subtemas, por fim, os subtemas podem ser relacionados com outros subtemas e, ao final da hierarquia, tem-se os códigos.

Figura 4 - Exemplo de Mapa Temático



Fonte: Cruzes e Dyba (2011).

3.5.4 Criação do Modelo de Alto Nível

O próximo passo é a criação do modelo de alto nível. Para Cruzes e Dyba (2011), devem-se avaliar os temas descobertos e os trabalhos nas etapas anteriores com o intuito de identificar a representatividade dos temas e suas relações. Como resultado, tem-se um modelo de alto nível que reflete a exploração e a interpretação realizadas a respeito dos temas identificados, fenômenos observados e das relações existentes entre eles. Este modelo é uma visão gráfica dos temas e suas relações, uma visão mais descritiva, uma taxinomia ou uma teoria. Ainda de acordo com os autores, as questões de pesquisa ajudam a identificar qual o tipo do modelo de alto nível deverá ser utilizado como saída para representar a síntese temática.

Ainda de acordo com os autores, há uma heterogeneidade presente entre os estudos realizados na ES com relação às descobertas realizadas, diversidade de métodos aplicados, ferramentas ou populações. Torna-se necessário que o contexto no qual os resultados dos artigos foram encontrados deva ser levado em consideração, uma vez que se pretende unificar os conceitos recorrentes ou declarações a respeito de determinados assuntos.

Portanto, para os estudos primários que fornecem informações a respeito das relações entre as descobertas e o contexto dos estudos, deve-se comparar e confrontar como as relações foram identificadas e analisadas nestes estudos. No entanto, para os estudos que não fornecem estas informações, devem-se utilizar dados previamente extraídos destes estudos.

Cruzes e Dyba (2011) sugerem alguns passos que deverão ser seguidos para se obter um modelo de alto nível:

a) Deve-se revisar o modelo de alto nível da última etapa. Para isso, cada ramo deverá ser descrito com seu conteúdo e os resultados, o contexto das informações e a criação de temas de ordem superior;

b) Identificar as ligações entre os temas de mais alto nível e as provas que fundamentam e contexto desta evidência;

c) Explorar as conexões com a teoria e pesquisas prévias, contextualizar novamente, definir e ainda refinar os temas de mais alto nível;

d) Criar um modelo, taxonomia, mapa temático, ou teoria relacionada aos temas de mais alto nível e às suas evidências.

3.5.5 Avaliação da Confiabilidade da Síntese

Neste último passo, deve-se avaliar se os resultados da pesquisa são confiáveis. A avaliação da confiabilidade da síntese depende da qualidade dos estudos primários utilizados na pesquisa, bem como na quantidade de evidências identificadas.

De acordo com Cruzes e Dyba (2011), existem alguns fatores que são responsáveis por influenciar a confiabilidade da síntese: a qualidade metodológica dos estudos primários e os métodos utilizados na síntese. Além disso, a confiabilidade das interpretações realizadas pelo pesquisador, em uma síntese temática, está relacionada à explanação dos argumentos, permitindo assim, que interpretações alternativas possam ser realizadas pelos leitores.

3.6 CONSIDERAÇÕES FINAIS

A engenharia de software baseada em evidências possui dois métodos principais de investigação: estudos primários e secundários. Os estudos primários são utilizados para avaliar uma hipótese formulada pelo pesquisador, enquanto que os secundários são utilizados para estabelecer comparações, generalizações sistemáticas ou mapeamento das investigações individuais a partir de um conjunto de estudos primários selecionados de forma controlada.

Existem diversos métodos que podem ser utilizados para a síntese baseada em evidências. No entanto, a escolha do método a ser utilizado é baseada no objetivo do trabalho a qual se pretende atingir.

No próximo capítulo será apresentado o protocolo da revisão sistemática elaborado e adotado pelos autores deste trabalho, bem como os resultados obtidos através de sua execução.

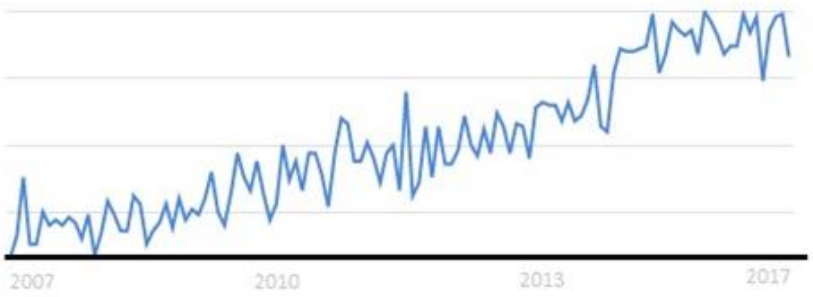
Capítulo 4 – Um Mapa Temático sobre Dívida de Projeto

Neste capítulo será apresentado o protocolo da revisão sistemática que foi utilizado no trabalho, sua execução, bem como, os resultados encontrados para cada questão de pesquisa abordada neste protocolo. Será apresentado o mapa temático resultante da análise dos estudos experimentais que têm sido realizados sobre dívida de projeto.

4.1 INTRODUÇÃO

Em virtude de ser um tema relativamente novo, DT tem despertado o interesse da comunidade e, com isso, passou a ser um termo constantemente pesquisado na web. Esta informação pode ser confirmada através da Figura 5, extraída do *trends.google.com.*

Figura 5 - Resultado da pesquisa sobre o termo "Technical Debt" no trends.google.com



Diversos trabalhos têm sido realizados na área de DT com o intuito de reunir evidências a partir de estudos primários. Dentre eles, pode-se citar: Tom *et al.* (2012), Villar e Matalonga (2013), Tom *et al.* (2013), Ampatzoglou *et al.* (2015), Li *et al.* (2015) e Alves *et al.* (2016). Embora esses estudos tenham fornecido uma visão abrangente, nenhum realizou uma análise específica sobre os resultados dos estudos experimentais encontrados na área. A aplicação de mapa temático com base nos trabalhos que apresentaram estudos experimentais permite estabelecer comparações, generalizações sistemáticas ou mapeamento das investigações individuais.

Assim, este trabalho realizou uma Revisão Sistemática (RS) da literatura com o objetivo de responder à seguinte questão de pesquisa: "***Qual o estado atual das pesquisas sobre dívida***

técnica considerando as evidências geradas através de estudos primários?". Para responder essa questão, as seguintes questões complementares foram formuladas:

a) QC1. Qual o tipo de experimento realizado no artigo?

b) QC2. Qual o perfil dos participantes do estudo?

c) QC3. O pacote experimental utilizado no estudo foi disponibilizado? Caso o pacote tenha sido disponibilizado, quais os meios utilizados?

d) QC4. Qual tipo de DT foi considerado na avaliação experimental?

e) QC5. Qual a classificação dos projetos analisados?

f) QC6. Os estudos analisados utilizaram alguma ferramenta? Qual?

g) QC7. Entre os estudos analisados, algum deles é continuação ou replicação de outro (s)?

h) QC8. Quais são as evidências encontradas através da avaliação experimental?

O protocolo utilizado nesta revisão da literatura foi desenvolvido pelo autor em conjunto com seu orientador. Após finalizado, três pesquisadores da área o revisaram com o intuito de verificar se os questionamentos abordados estavam de acordo com o objetivo do protocolo e com as informações que deveriam ser retornadas.

Além desta introdução, este capítulo possui mais seis seções. A seção 4.2 apresenta os trabalhos relacionados, já a 4.3 apresentará o planejamento da RS executada. Em seguida, a seção 4.4 exibirá os resultados obtidos, a seção 4.5 apresenta uma discussão sobre eles e, a seção 4.6, as limitações do estudo. Por fim, a seção 4.7 conterá as considerações finais deste capítulo.

4.2 PLANEJAMENTO DA REVISÃO DA LITERATURA

Esta seção apresenta o protocolo elaborado considerando: a definição das questões de pesquisa, a *string* de busca, as bibliotecas utilizadas e o processo de seleção e de exclusão dos artigos retornados.

4.2.1 Questão de Pesquisa

O protocolo de revisão sistemática utilizado neste trabalho apresenta como questão de pesquisa principal: ***Qual o estado atual das pesquisas sobre Dívida Técnica de Projeto considerando as evidências geradas através de estudos primários?*** Com o intuito de responder esta questão principal, foram elaboradas nove questões complementares, sendo estas responsáveis por estabelecer parâmetros que auxiliem na identificação do estado atual das pesquisas sobre DT:

4.2.1.1 QC1. Qual o tipo de experimento realizado nos artigos?

Em uma avaliação experimental, pode-se ter quatro tipos de experimentos: *in vitro, in vivo, in virtuo e in silico.* Experimentos considerados *in vivo* envolvem pessoas e seus ambientes de trabalho. Sendo assim, pesquisas que são realizadas em organizações desenvolvedoras de software durante o processo de desenvolvimento deste e em condições reais são considerados experimentos *in vivo*. Os estudos *in vitro* são executados em ambientes controlados. Já experimentos *in virtuo* são caracterizados pela interação entre os participantes e um modelo computadorizado que representa a realidade. Nesse tipo de experimento, o comportamento do ambiente com o qual os participantes interagem é descrito como um modelo e é representado por um software. Por fim, têm-se estudos *in silico* que são caracterizados pela representação de modelos computacionais tanto para os participantes quanto para o mundo real. A identificação do tipo de experimento utilizado nos trabalhos analisados permitirá avaliar o nível de controle adotado pelo estudo, bem como a abrangência dos resultados (MIAN *et al*., 2016).

4.2.1.2 QC2. Qual o perfil dos participantes do estudo?

As informações identificadas através desta questão permitirão mapear o perfil dos participantes utilizados nos estudos, sendo estes: estudantes, profissionais e, estudantes e profissionais. Além disso, essas informações servirão para investigar se as tecnologias que abordam a DT estão inseridas em ambientes acadêmicos e/ou industriais.

4.2.1.3 QC3. O pacote experimental utilizado no estudo foi disponibilizado? Em caso positivo, em quais meios?

Esta questão tem como objetivo identificar se o pacote do experimento foi disponibilizado. O acesso a essa informação permite que pesquisadores analisem as informações contidas nos pacotes e possam, por exemplo, replicar os estudos realizados.

4.2.1.4 QC4. Qual tipo de DT foi considerado na avaliação experimental?

A identificação dos tipos de DT que têm sido avaliados através de estudos experimentais permite que seja realizada uma análise sobre os limites do conceito de DT no desenvolvimento de *software*.

4.2.1.5 QC5. Qual a classificação dos projetos analisados?

Os projetos de software podem ser classificados como *open source* ou proprietário. Esta informação será coletada para auxiliar na caracterização dos estudos que têm sido realizados na área.

4.2.1.6 QC6. Os estudos analisados utilizaram alguma ferramenta? Qual?

Essa questão objetiva identificar as ferramentas que têm sido utilizadas na execução dos estudos na área de DT. O mapeamento dessa informação permitirá auxiliar futuras pesquisas quanto à existência ou uso de ferramentas na área de DT.

4.2.1.7 QC7. Entre os estudos analisados, algum deles é continuação ou replicação de outro (s)?

As informações extraídas para responder este questionamento permitirão identificar se replicações de estudos têm sido realizadas na área.

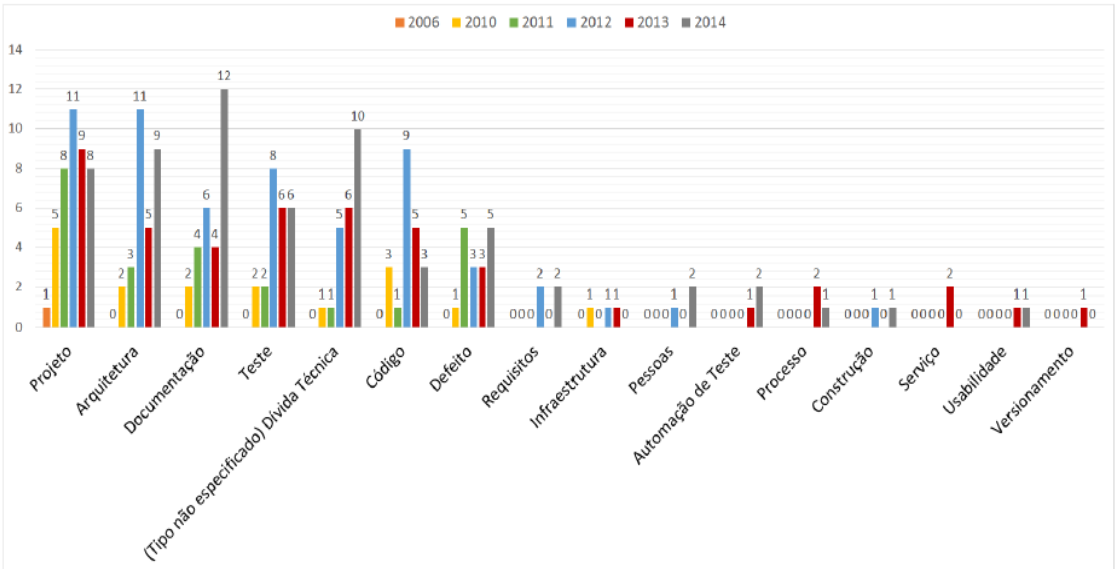
4.2.1.8 QC8. Quais são as evidências encontradas através da avaliação experimental?

A identificação da avaliação experimental bem como a análise das evidências encontradas possibilitará uma melhor compreensão sobre o que os estudos primários têm

apontado na área de DT. Essas informações servirão como um guia para os pesquisadores, além de disponibilizar para profissionais informações que refletem o conhecimento atual sobre DT.

Apenas os estudos primários que realizaram alguma avaliação experimental na área de DT foram selecionados. Além disso, para análise dessa questão, apenas a dívida de projeto foi considerada. De acordo com Alves *et al* (2016), esse é o tipo de dívida que tem sido mais investigado pela comunidade científica (Figura 6).

Figura 6 - Estudos realizados acerca dos tipos de dívida técnica



Fonte: Alves *et al.* (2016).

4.2.2 *String* de Busca e Bibliotecas Digitais

Esta revisão de literatura objetiva investigar apenas evidências geradas a partir de estudos experimentais sobre DT. Assim, para definir a *string* de busca, os seguintes critérios foram utilizados:

População:

Technical Debt, Software

Intervenção:

Empirical: Experimentation, Experiment

Study: Evaluation

Feasibility Study, Case Study, Survey, Controlled Experiment, Exploratory Study

A partir dessas palavras chave, a seguinte *string* de busca foi definida:

P: (“*Technical Debt*”) and (“*Software*”)

I: (“*Empirical Study ”or “ Empirical Evaluation” or “Experimentation Study” or “Experimentation Evaluation” or ”Experiment” or “Feasibility Study” or “Case Study” or “Survey” or “Controlled Experiment” or “Exploratory Study”*)

Resultando na seguinte *string* utilizada nas bibliotecas digitais:

(((“Technical Debt”) and (“Software”)) AND (“Empirical Study ”or “ Empirical Evaluation” or “Experimentation Study” or “Experimentation Evaluation” or ”Experiment” or “Feasibility Study” or “Case Study” or “Survey” or “Controlled Experiment” or “Exploratory Study”))

A Tabela 4 representa as bibliotecas digitais que foram utilizadas para a execução da *string* de busca neste trabalho. Nela, também pode-se observar a quantidade de artigos que foram retornados em cada biblioteca digital. Percebe-se que a ACM retornou 57 trabalhos, enquanto que a IEEE retornou 28. Já a *Springer* retornou 113 estudos, enquanto que a *Science Direct* 37, a *Scopus* 71 e a *Engineering Village* 60. Ao total, 366 artigos foram inicialmente identificados.

Tabela 4 - Bibliotecas digitais utilizadas

#	Biblioteca Digital	Link	Resultados
1	ACM Digital Library	http://dl.acm.org/	57
2	IEEE Xplorer	http://ieeexplore.ieee.org/search/advsearch.jsp?expression-builder	28
3	Science Direct	http://www.sciencedirect.com/science/search	37
4	Engineering Village	https://www.engineeringvillage.com/search/quick.url	60
5	Springer Link	http://www.springer.com/gp/	113
6	Scopus	http://www.scopus.com/home.url	71

Fonte: Elaboração própria do autor desta dissertação (2017).

4.2.3 Processo de Seleção

Após a execução da *string* de busca, os artigos retornados passaram por um processo de seleção baseado nos critérios de inclusão e exclusão. Os seguintes critérios de inclusão foram considerados:

a) O estudo deverá apresentar uma avaliação experimental relacionada ao assunto DT;

b) A data de publicação do artigo deverá ser até 2016, pois a pesquisa com relação aos estudos primários que realizaram alguma avaliação experimental ocorrerá até o mês de dezembro deste ano.

Em contrapartida, os critérios de exclusão foram:

a) Artigos que não apresentem uma avaliação experimental relacionada ao assunto DT;

b) Artigos que só estão disponíveis em formato de resumos ou conferências ou apresentações do Power Point;

c) Capítulos de livros serão descartados.

O processo de seleção consistiu na leitura do título e do resumo de cada artigo. Ao final, um segundo pesquisador revisou os artigos filtrados com o objetivo de avaliar o resultado do processo. Após a finalização da seleção, a extração de dados para cada estudo selecionado foi iniciada.

A Tabela 5 contém informações a respeito da quantidade de artigos que foram analisados. A coluna descartados representa a quantidade de artigos que não obedeceram aos critérios de inclusão. Já a coluna repetidos representa o número de artigos que já haviam sido retornados por uma biblioteca digital. Por fim, a coluna indisponíveis representa a quantidade de artigos que não estavam acessíveis por meio do download ou era necessário realizar o pagamento de uma taxa. Ao final, 51 artigos foram selecionados para análise.

Tabela 5 - Artigos a serem analisados

Biblioteca Digital	Artigos Retornados	Descartados	Repetidos	Indisponíveis	Total
ACM	57	35	0	0	22
IEEE	28	3	4	1	20
Scopus	71	7	32	28	4
Engineering Village	60	6	35	19	0
Science Direct	37	30	5	0	2
Springer	113	17	2	91	3
				Total:	51

Fonte: Elaboração própria do autor desta dissertação (2017).

4.3 ANÁLISE DE DADOS

Para a elaboração do mapa temático, as seguintes etapas foram realizadas:

a) **Primeira Etapa:** Realizou-se a execução da *string* de busca nas seguintes bibliotecas digitais: ACM, IEEE, Scopus, Engineering Village, Science Direct e Springer com o intuito de obter os estudos primários que realizaram alguma avaliação experimental acerca de DT;

b) **Segunda Etapa:** Após o término da primeira etapa, cada artigo retornado foi submetido ao processo de seleção, descrito na seção anterior, com o intuito de verificarmos se eles obedeciam ou não aos critérios pré-estabelecidos. Após, a execução desta fase, outro pesquisador conferiu os resultados obtidos e, em caso de divergência, outro pesquisador era utilizado para que pudéssemos chegar a um consenso;

c) **Terceira Etapa:** Após a identificação dos estudos que obedeceram ao critério de seleção, realizou-se o processo de identificação dos tipos de dívidas estudados em cada artigo. Após esse processo de identificação, outro pesquisador conferiu os resultados obtidos e, em caso de divergência, outro pesquisador era utilizado para que pudéssemos chegar a um consenso;

d) **Quarta Etapa:** Para cada artigo, realizou-se a extração das seguintes informações: questões de pesquisa e evidências;

e) **Quinta Etapa:** Após a extração das informações, inicia-se a fase de códigos. Nesta fase, para cada questão de pesquisa encontrada foi atribuída uma numeração

sequencial (rótulo). Tendo em vista que para este estudo foi adotada uma abordagem interativa e incremental, a qualquer momento a numeração poderia ser alterada. Após a finalização desta atribuição, têm-se os subtemas;

f) **Sexta Etapa:** Após a extração das informações, inicia-se a fase de códigos. Nesta fase, para cada questão de pesquisa encontrada foi atribuída uma numeração sequencial (rótulo). Tendo em vista que neste estudo adotou-se uma abordagem interativa e incremental, a todo instante a numeração poderia ser alterada. Após a finalização desta atribuição, têm-se os subtemas;

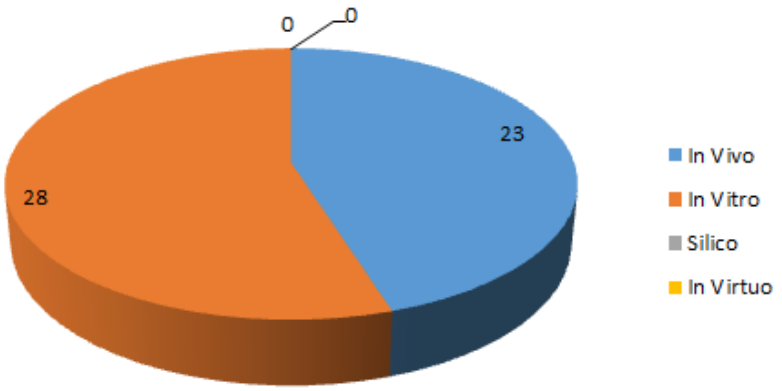
g) **Sétima Etapa:** Os subtemas encontrados são agrupados a um tema, onde este representa algum aspecto relacionado ao desenvolvimento de software e a DT. Para a escolha do tema, levou-se em consideração o aspecto ao qual as questões de pesquisa estavam relacionadas.

4.4 RESULTADO

4.4.1 Qual o tipo de experimento realizado nos artigos?

A Figura 7 apresenta os tipos de experimentos que foram encontrados durante o processo de análise dos artigos selecionados. De acordo com os dados apresentados, os pesquisadores utilizaram, em sua maioria, ambientes controlados para o desenvolvimento de suas pesquisas, embora muitos estudos também tenham sido realizados *in vivo*. Percebe-se que, dentre os 51 artigos selecionados, 28 eram do tipo *In Vitro*, enquanto 23 eram *In Vivo*. Quanto aos tipos *in Silico* e *in Virtuo*, eles não foram utilizados como meio de avaliação experimental entre os artigos analisados.

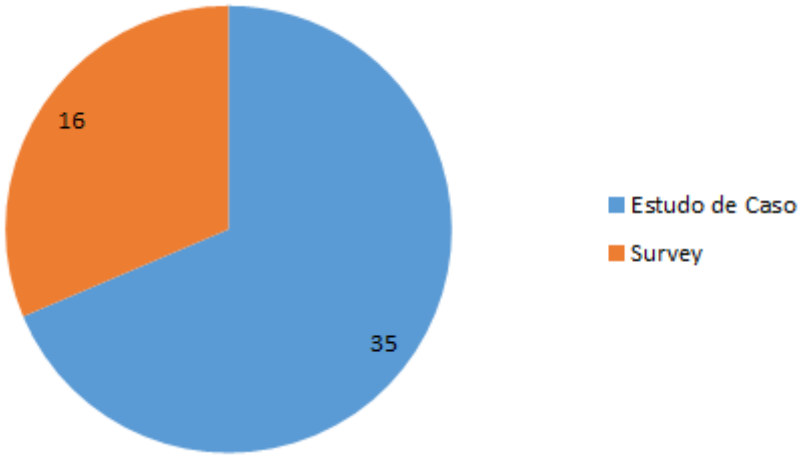
Figura 7 - Tipos de Experimentos



Fonte: Elaboração própria do autor desta dissertação (2017).

A Figura 8 representa os tipos de estudos realizados nos trabalhos analisados. Percebe-se que 16 estudos utilizaram survey, enquanto que 35 realizaram estudos de caso.

Figura 8 - Tipos de Estudos



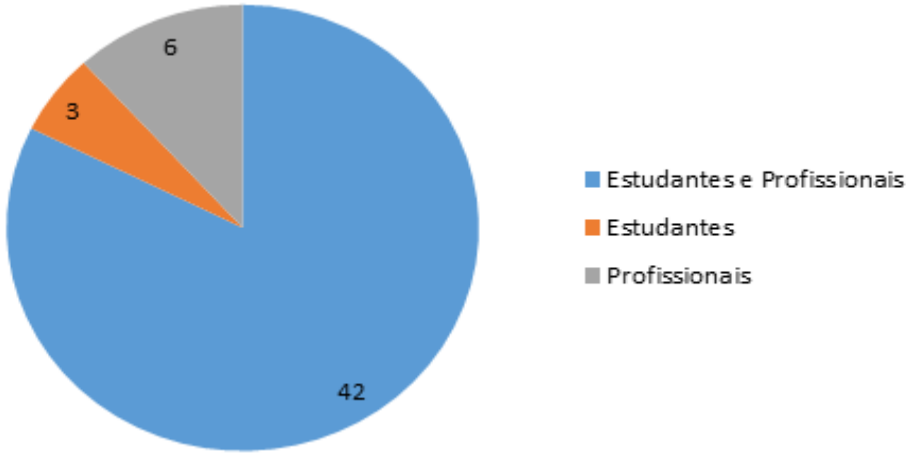
Fonte: Elaboração própria do autor desta dissertação (2017).

4.4.2 Qual o perfil dos participantes do estudo?

Três perfis de participantes foram identificados: estudantes, profissionais e, estudantes e profissionais. Conforme pode ser observado na Figura 9, 42 estudos realizaram avaliações

experimentais considerando a participação de estudantes e profissionais, enquanto que 6 consideraram somente profissionais e 3 apenas estudantes.

Figura 9 - Perfil dos Participantes



Fonte: Elaboração própria do autor desta dissertação (2017).

4.4.3 O pacote experimental utilizado no estudo foi disponibilizado? Em caso positivo, em quais meios?

Dentre os 51 artigos analisados, apenas três disponibilizaram o pacote experimental:

a) A Large-Scale Empirical Study on Self-Admitted Technical Debt;

b) A Contextualized Vocabulary Model for Identifying Technical Debt on Code Comments;

c) Examining the Impact of Self-admitted Technical Debt on Software Quality.

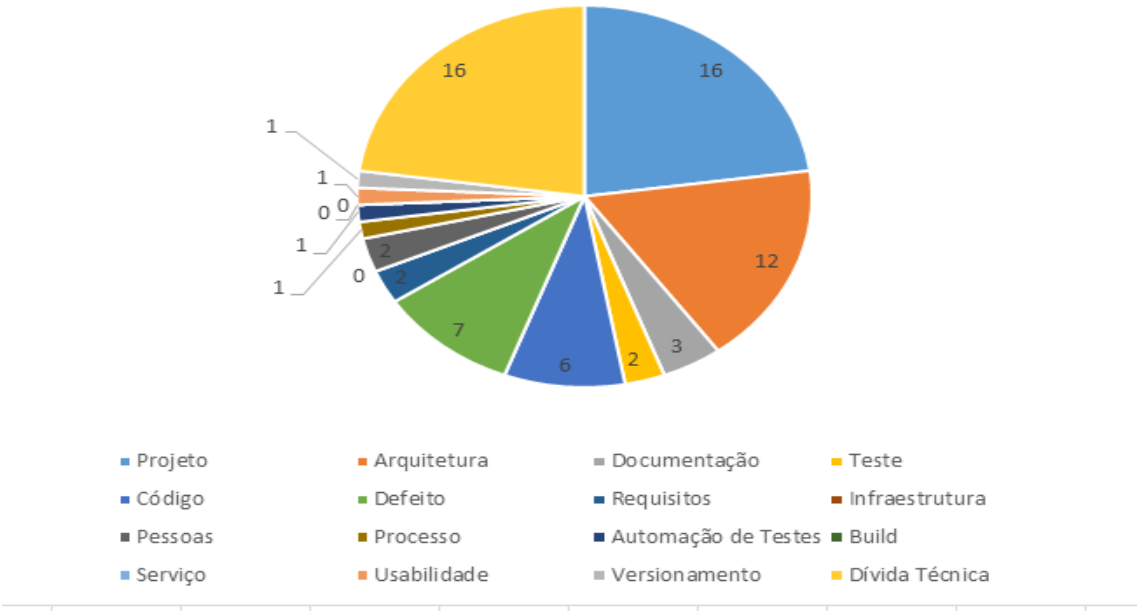
Os pacotes experimentais foram disponibilizados nos seguintes endereços, respectivamente: https://projects.apache.org/indexes/quick.html e https://github.com/eclipse, http://homes.dcc.ufba.br/~marioandre/page/cvm-td/TD_SENouns.pdf, http://users.encs.concordia.ca/˜eshihab/data/ICSME2014/data.zip.

Estes são os estudos que podem ser replicados por outros pesquisadores, levando-se em consideração o mesmo ambiente e/ou configuração ao qual foi submetido.

4.4.4 Qual tipo de DT foi considerado na avaliação experimental?

A Figura 10 ilustra os tipos de dívida encontrados nos artigos analisados. Percebe-se que 16 estudos analisaram dívida de técnica (os artigos que realizaram avaliações experimentais sobre DT de uma forma geral, bem como gestão de dívida, foram classificados como DT e não estão associados a um tipo específico), 16 analisaram dívida de projeto, 12 dívida de arquitetura, 7 dívida de defeito, 6 dívida de código, 3 dívida de documentação, 2 dívida de pessoas, 2 dívida de testes, 2 dívida de requisitos, 1 dívida de automação de testes, 1 dívida de processo, 1 dívida de usabilidade e 1 dívida de versionamento. Observou-se também que os tipos dívida infraestrutura, *build* e serviço não foram identificados.

Figura 10 - Tipos de Dívidas considerados

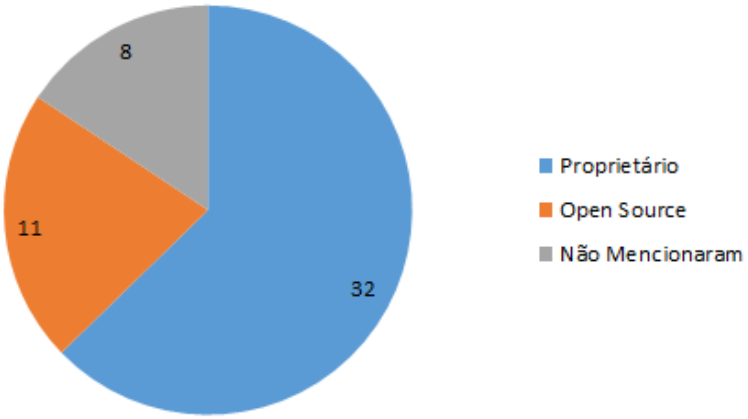


Fonte: Elaboração própria do autor desta dissertação (2017).

4.4.5 Qual a classificação dos projetos analisados?

A Figura 11 apresenta os tipos de projetos identificados. 32 dos estudos utilizaram projetos proprietários, enquanto que 11 *open source*. É importante observar ainda que 8 artigos não mencionaram qualquer indicativo que permitisse a classificação destes.

Figura 11 - Classificação dos Projetos

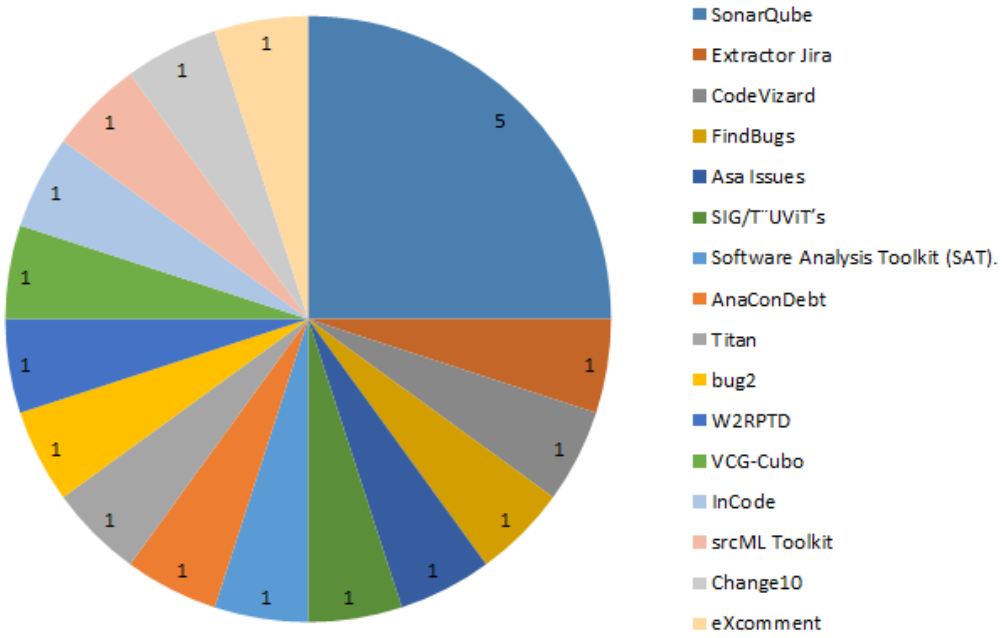


Fonte: Elaboração própria do autor desta dissertação (2017).

4.4.6 Os estudos analisados utilizaram alguma ferramenta? Qual?

Durante o processo de extração de dados, algumas ferramentas foram identificadas: *Titan, Change10, SonarQube, Bug2, CodeVizard, FindBugs, ASA issues, SIG/T¨UViT's, Software Analysis Toolkit (SAT), AnaConDebt, DebtFlag, RE-KOMBINE, Extractor Jira, srcML Toolkit, W2RPTD, VCG-Cubo,* e *eXcomment.* A Figura 12 ilustra a quantidade de vezes que cada uma delas foi considerada nos trabalhos analisados. Percebe-se que a *SonarQube* foi a mais utilizada nos estudos, sendo ela classificada como uma ferramenta que auxilia no processo de identificação de DT.

Figura 12 - Ferramentas utilizadas



Fonte: Elaboração própria do autor desta dissertação (2017).

4.4.7 Dentre os estudos analisados, algum deles é continuação ou replicação de outro (s)?

Dos 51 artigos analisados, apenas um apresentou uma replicação de um estudo [*A Large-Scale Empirical Study on Self-Admitted Technical Debt*]. O estudo em questão realizou uma replicação do trabalho de Potdar e Shihab (2014) e apresentou como objetivo a investigação da evolução da DT *Self-Admitted*, bem como a sua relação com a qualidade do *software*.

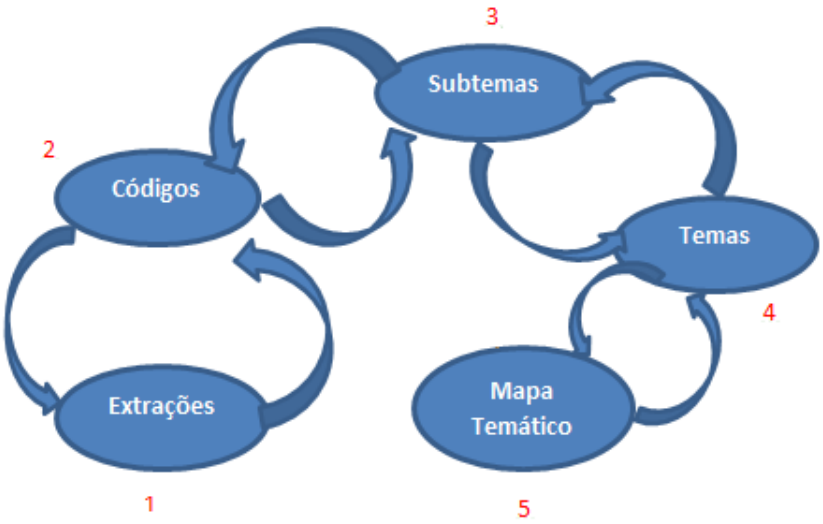
4.4.8 Quais são as evidências encontradas através da avaliação experimental?

A subseção a seguir apresentará as evidências identificadas para a dívida de projeto no processo de extração de dados, bem como o tipo de avaliação experimental utilizado.

O processo de análise dos dados adotado neste trabalho considerando a abordagem incremental e iterativa do método adotado são ilustrados na figura 13. Através desta, percebe-se que a primeira etapa consistiu na extração das informações. Nesta etapa, para o tipo de dívida de projeto, foram extraídas as informações referentes às questões de pesquisas, bem como as evidências encontradas.

Após a fase de extração, tem-se a etapa de códigos. Nesta fase, cada questão de pesquisa encontrada, através do processo de extração, foi rotulada (numeração), originando assim os subtemas. Após isso, cada subtema é agrupado de acordo com um tema, sendo que um tema representa algum aspecto relacionado ao processo de desenvolvimento de *software* e à dívida técnica, tais como: aspectos humanos, programação, desenvolvimento, manutenção e etc. Por fim, apresenta-se o mapa temático sobre o referido tipo de dívida.

Figura 13 - Níveis de Interpretação



Fonte: Elaboração própria do autor desta dissertação (2017).

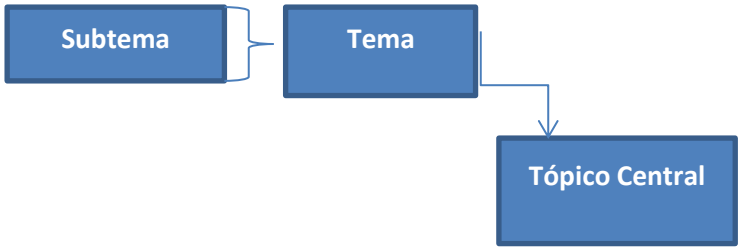
A Figura 14 representa um exemplo de mapa temático, bem como os seus componentes, onde:

a) Tópico Central: é responsável por apresentar o tipo de DT para qual estará sendo gerado o mapa temático;

b) Tema: são componentes responsáveis por agrupar as evidências encontradas (subtemas). Sendo assim, estes componentes estão relacionados aos termos relacionados a DT, bem como a aspectos relacionados ao processo de desenvolvimento de *software;*

c) Subtema: são responsáveis por representar as evidências encontradas referentes ao tema ao qual estão associadas.

Figura 14 - Exemplo de mapa temático



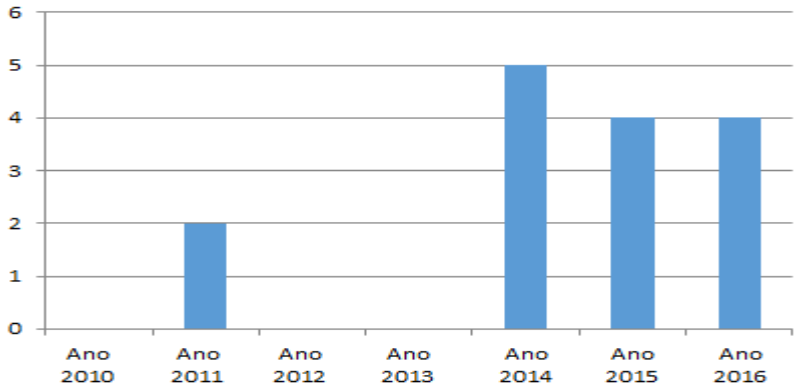
Fonte: Elaboração própria do autor desta dissertação (2017).

A próxima seção apresenta o mapa temático gerado para a dívida de projeto encontrada a partir da execução do protocolo desta revisão.

4.4.8.1 Dívida de Projeto

De acordo com Guo e Seaman (2011), uma dívida de projeto se refere a itens que podem ser descobertos através da análise do código fonte, bem como identificação de violações das boas práticas de orientação a objetos. A Figura 15 ilustra a quantidade de trabalhos que realizaram avaliações experimentais sobre dívida de projeto. Pode-se perceber que existe uma maior concentração de estudos relacionados entre os anos de 2014 e 2016.

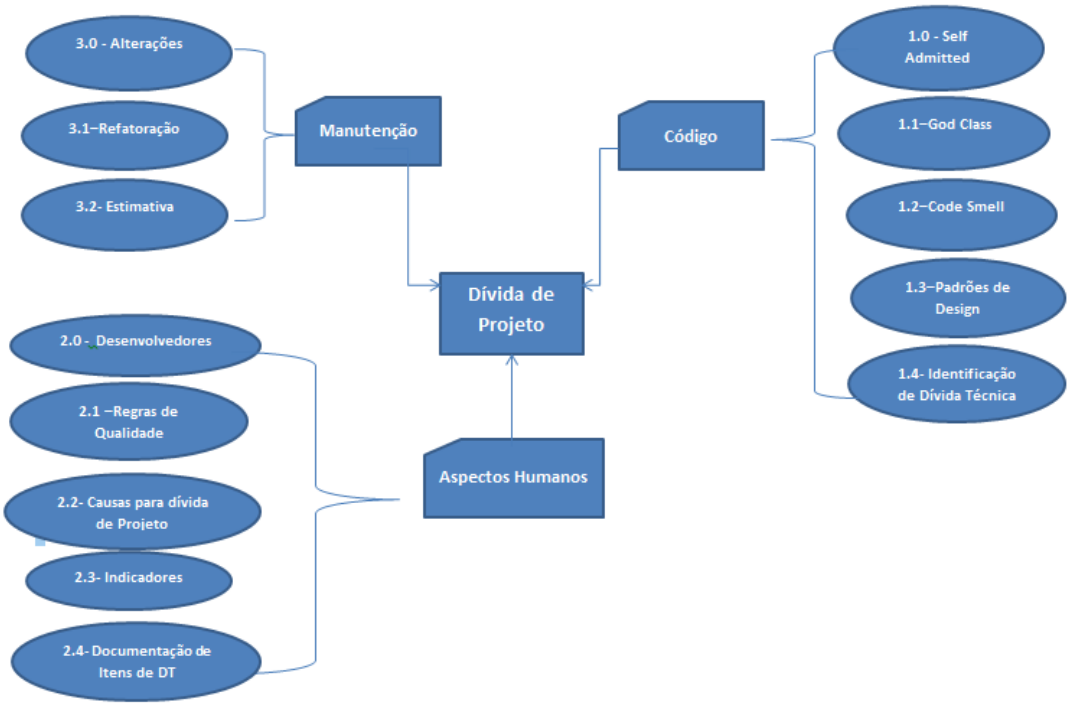
Figura 15 - Estudos que realizaram avaliação experimental sobre Dívida de Projeto



Fonte: Elaboração própria do autor desta dissertação (2017).

A Figura 16 representa o mapa temático elaborado para este tipo de dívida com base nas informações extraídas dos artigos analisados.

Figura 16 - Mapa Temático dívida projeto



Fonte: Elaboração própria do autor desta dissertação (2017).

Percebe-se que existe um tópico central, denominado de *dívida de projeto*, e associado a ele tem-se os temas e estes, por sua vez, possuem subtemas:

a) Código: este tema está relacionado ao nível de incidência de uma DT *Self-Admitted*. Além disso, como se comporta esse tipo de dívida com relação ao histórico de alterações em projetos de *software*, os componentes que são mais propensos ao surgimento de DT *Self-Admitted*, o motivo para a sua inserção nos projetos de *software* e a quantidade deste tipo de dívida removida, após a sua inserção. Além disso, durante a evolução do sistema, o número de classes que possuem *smell* sem padrões de *design* pode ser alterado em virtude da adoção destes padrões. Por fim, também identificou-se evidência sobre o uso de ferramentas para identificação de itens DT em projetos de *software*, bem como se há sobreposição entre os itens identificados pelas ferramentas quando comparados aos identificados manualmente;

b) Aspectos Humanos: este tema está relacionado ao nível de conhecimento acerca da DT entre os *stakeholders*, bem como a dificuldade encontrada por estes para os processos de identificação e documentação de um item de DT, e a importância das regras de qualidade em um projeto de *software*. Além disso, as atividades dos desenvolvedores podem ser utilizadas como indicadores para a detecção de DT e, além disso, quais os fatores que contribuem de forma significativa para o acúmulo da dívida;

c) Manutenção: este tema está relacionado ao comportamento da DT *Self-Admitted* com relação ao histórico de alterações ocorridas em projetos de *software*, bem como os efeitos provocados por estas alterações. Além disso, o uso de métodos para estimar a DT com relação à qualidade e a adoção de abordagens que possibilitem melhorias na qualidade de um projeto também são considerados neste tema.

As próximas subseções apresentarão as evidências encontradas para os subtemas associados a cada tema encontrado.

4.4.8.2 Código

Ao longo de um ciclo de vida de desenvolvimento de *software*, desenvolvedores criam soluções incompletas ou temporárias que irão exigir retrabalho e produção de erros,

ocasionando assim, a DT. Tais soluções podem impactar negativamente na manutenção do *software* (WEHAIBI *et al.*, 2016).

Potdar e Shihab (2014) apud Sultan Wehaibi *et al.*, (2016) relataram em seu trabalho que através da análise dos comentários presentes nos códigos fontes de uma aplicação, torna-se possível a identificação de dívida técnica, denominada de *Self-Admitted* (SATD). Sendo esta, uma DT que os próprios desenvolvedores informam sobre sua existência por meio de comentários presentes no código fonte.

Em 2016, Gabriele Bavota e Barbara Russo (2016) realizaram um estudo em 159 sistemas de código aberto com o intuito de descobrir o nível de incidência de uma DT *Self-Admitted* nestes projetos. Os resultados alcançados indicam a difusão do SATD nos sistemas analisados em termos de número de instâncias. Durante a análise dos comentários, percebeu-se, em média, que a porcentagem de comentários que relatam o SATD foi baixa para ambos os sistemas.

Dentre os 366 comentários analisados manualmente e que relataram SATD, 93 foram classificados como falsos positivos. Sendo assim, tem-se um total de 25% da DT identificada automaticamente através dos padrões estabelecidos por Potdar e Shihab (2014). Além disso, o tipo mais usado de SATD são as dívidas de código (30% da instância), defeito (20%) e de *design* (13%). Sendo assim, pode-se afirmar que projetos que possuem dívidas SATD, geralmente, estão relacionados aos três tipos mencionados.

Aniket Potdar e Emad Shihab (2014) realizaram um estudo visando identificar a quantidade de DT *Self-Admitted* existente nos projetos de *software*, bem como a quantidade desta dívida que é removida, após a sua inserção. Dentre os projetos analisados, existe uma margem entre 2,4% e 31% com relação ao tipo de dívida *Self-Admitted* encontrada em cada projeto. Além disso, desenvolvedores com mais experiência tendem a introduzir a maior parte da DT *Self-Admitted* e existem fatores que contribuem para a inserção deste tipo de dívida, tais como: pressão com relação ao tempo de entrega do projeto, como também, a complexidade do código. Por último, a quantidade de DT *Self-Admitted* removida em um projeto varia entre 26,3% e 63,5%.

Wehaibi *et al.* (2016) apresentaram um estudo com o intuito de identificar se os arquivos com SATD têm mais defeitos quando comparados com arquivos sem SATD, bem como se as alterações SATD introduzem defeitos futuros e, por fim, se as mudanças relacionadas ao SATD tendem a ser mais difíceis de serem tratadas. De acordo com o estudo, não existe uma relação

clara entre defeitos e SATD, uma vez que entre os projetos analisados, os arquivos SATD apresentam mais mudanças com relação à correção de erros, enquanto que os arquivos sem SATD têm mais defeitos. Além disso, as alterações SATD estão associadas a menos defeitos futuros. Sendo assim, o impacto causado por uma DT não está relacionado a defeitos, mas sim ao grau de dificuldade com relação à futura manutenção dos sistemas. Pois, em arquivos SATD, as alterações apresentam um nível de dificuldade médio ou grande, desta forma, indicando que as mudanças no SATD são mais difíceis do que em arquivos não-SATD. Por fim, uma vez que o SATD é introduzido, há uma porcentagem maior de correção de defeitos e, por isso, existe uma tendência daquele trecho do projeto apresentar um menor número defeitos futuros.

Zazworka *et al.* (2011) realizaram um estudo sobre a relação existente entre o tamanho da classe e as alterações sofridas por ela. De acordo com os resultados obtidos, geralmente, as *good class* são mais propensas a mudanças e, consequentemente, a possuírem mais defeitos, devido ao seu alto grau de acoplamento. Sendo assim, torna-se importante que este tipo de classe seja monitorado e gerenciado nos projetos de *software*. Além disso, estas classes estão relacionadas à DT e, por isso, vão exigir mais esforços com relação à manutenção devido a sua complexidade. Por fim, não se pode afirmar que há uma correlação entre a quantidade de defeitos e o tamanho de uma classe.

Zazworka *et al.* (2011) afirmaram que para realizar o pagamento de uma DT relacionada a *good class,* deve-se realizar algumas atividades típicas da refatoração: dividir a *good class* em múltiplas classes, liberar o código não utilizado, simplificando assim, a *good class* e, por fim, mover alguns métodos. Atualmente, a detecção de *code smell* tem sido utilizada para ajudar a identificar componentes que precisam ser refatorados. No entanto, as propriedades das dívidas, o valor desta e a taxa de juros a ser paga ainda não foram bem estabelecidos.

De acordo com Tufano *et al.* (2015), geralmente, códigos apresentam *code smell* desde o momento da sua criação. Desta forma, essa afirmação contradiz o senso comum de que eles são provenientes apenas da evolução do *software*. Por outro lado, artefatos de *software* apresentam *code smell* também como consequência das atividades de manutenção e evolução. Sendo assim, atividades relacionadas à implementação de novos recursos ou a melhoria dos já existentes tendem a apresentar *code smell*. Além disso, a existência de *code smell* em classes que possuem padrões de projeto é considerada pequena.

De acordo com Bartosz Walter e Tarek Alkhaeir (2016), padrões de projeto descrevem soluções para problemas de *design* recorrentes que podem ser adaptados e aplicados

repetidamente. Além disso, o uso de padrões de projeto em produtos de *software* fornece melhorias em várias características, como: compreensão, legibilidade, confiabilidade, etc. Enquanto que *code smell* são sintomas relacionados a problemas de *design* e, com isso, dificultam a manutenção futura de um *software*.

De acordo com o experimento realizado por Bartosz Walter e Tarek Alkhaeir (2016), as classes que possuem padrões de projeto inibem a presença de *code smell* em projetos de *software*. Além disso, o número relativo de *smells* em classes que possuem padrões e o número de *smells* em outras classes é aproximadamente estável ou ligeiramente decrescente com relação ao tempo. Por fim, a frequência de *smells* em classes com padrões é menor ou igual durante a evolução do sistema quando comparados com outras classes.

Zazworka *et al.* (2014) afirmam que o acúmulo de DT em *softwares* ocorre devido à necessidade de uma implementação rápida e, muitas vezes, "suja" para o cumprimento dos prazos estabelecidos para entrega do produto. Além disso, alguns tipos de DT acumulam ao longo do tempo sob a forma de código-fonte. Os autores propuseram um estudo para a detecção de DT através do uso de ferramentas. O estudo teve como intuito descobrir se estas ferramentas relatam a presença da dívida nas mesmas classes. Para isso, os autores utilizaram quatro técnicas para a identificação de DT, são elas: *code smell, automatic static analysis issues, grime buildup e modularity violations*. De acordo com o estudo, as quatro abordagens utilizadas apresentam pouca sobreposição. Ou seja, apresentam poucos resultados comuns ao relatarem os problemas encontrados nas classes. Além disso, violações de acoplamento disperso e da modularidade foram encontradas em classes com maior propensão ao defeito. Foi identificado também que há uma forte relação entre as violações da modularidade e classes que possuem tendência à mudança e *code smell*.

Zazworka *et al.* (2013) realizaram um estudo visando a identificação de itens de DT de forma automatizada através das ferramentas CodeVizard, FindBugs e ASA Issues. Após este processo de identificação, houve uma comparação entre os resultados obtidos pelas ferramentas com os relatados pelos desenvolvedores. Ainda de acordo com os autores, no projeto analisado, as ferramentas apoiaram a identificação de dívidas de defeitos e de *design*. Entretanto, existiram outros tipos de DT relatados pelos desenvolvedores que não foram identificados pelas ferramentas. Dessa forma, o uso de ferramentas para o processo de identificação dos tipos de dívida ainda é um desafio para a comunidade científica.

Um conjunto de indicadores tem sido utilizado por abordagens automatizadas para identificar itens DT, entretanto, alguns tipos de dívida não podem ser diretamente identificados a partir das métricas coletadas no código-fonte. Diante deste contexto, o CVM-TD é um modelo utilizado para apoiar a identificação de DT através da análise nos comentários presentes nos códigos fontes (Farias *et al.*, 2015). De acordo com os autores, o modelo apoia o processo de identificação dos itens de DT em um projeto, pois as dimensões consideradas pelo modelo são utilizadas pelos desenvolvedores ao escreverem comentários no código fonte.

As abordagens para a identificação de DT podem ser categorizadas como manual ou automatizada. No entanto, Zazworka *et al.* (2013) afirmam que processos de identificação realizados de forma manual pode ser mais precisa. Pois, provavelmente, os *stakeholders* identificarão a DT que seja mais significativa, enquanto que as análises automatizadas podem identificar muitas anomalias, no entanto, algumas sem importância. Além disso, os *stakeholders*, durante o processo de identificação, podem fornecer informações contextuais adicionais relacionadas a cada instância de DT, tais como: estimativas de esforço, impacto e etc. No entanto, o processo manual é mais demorado.

Durante o processo de identificação de DT de forma manual, Zazworka *et al.* (2013) afirmaram que o conhecimento acerca da DT ainda não é consolidado e, além disso, o nível de percepção entre os *stakeholders* é diferenciado.

A Tabela 6 descreve as evidências encontradas para cada um dos subtemas encontrados para o tema código, bem como os trabalhos associados a cada uma delas.

Tabela 6 - Evidências para o tema Código

Subtema	Evidências	Referência
Self-Admitted	Os comentários presentes nos códigos fontes de uma aplicação podem ser utilizados para a identificação da dívida *Self-Admitted*.	Potdar e Shihab (2014)
	Geralmente, itens de dívida SATD estão relacionados às dívidas de código, defeito e de *design*.	Gabriele Bavota e Barbara Russo (2016)
	A pressão com relação ao tempo de entrega do projeto, como também, a complexidade do código contribuem para a inserção deste tipo de dívida.	Potdar e Shihab (2014)

Subtema	Evidências	Referência
	Desenvolvedores com mais experiência tendem a introduzir a maior parte da DT *Self-Admitted*.	Potdar e Shihab (2014)
	A quantidade de DT *Self-Admitted* removida, após a sua inserção, varia entre 26,3% e 63,5%.	Potdar e Shihab (2014)
	Não existe uma relação clara entre defeitos e SATD.	Wehaibi et *al*., (2016)
	O impacto causado por uma dívida técnica não está relacionado a defeitos, mas ao grau de dificuldade com relação à futura manutenção dos sistemas.	Wehaibi et *al*., (2016)
	Alterações em arquivos SATDs apresentam um nível de dificuldade médio ou grande.	Wehaibi et *al*., (2016)
	Mudanças em arquivos SATD são mais difíceis do que em arquivos não-SATD.	Wehaibi et *al*., (2016)
	A introdução da dívida SATD em um determinado trecho do projeto tende a diminuir o número de defeitos futuros, pois há correção de defeitos.	Wehaibi et *al*., (2016)
Good Class	As *Good Class* são mais propensas a mudanças.	Zazworka et *al*., 2011
	As *Good Class* possuem mais defeitos.	Zazworka et *al*., 2011
	As *Good Class* devem ser monitoradas e gerenciadas nos projetos de *software*.	Zazworka et *al*., 2011
	Não se pode afirmar que há uma correlação entre a quantidade de defeitos e o tamanho de uma classe.	Zazworka et *al*., 2011
	Good Class estão relacionadas à dívida técnica.	Zazworka et *al*., 2011
	Good Class exigem mais esforços com relação à manutenção.	Zazworka et *al*., 2011
Code Smell	Geralmente, códigos apresentam *code smell* desde o momento da sua criação.	Tufano et *al*.(2015)
	Artefatos de *software* apresentam *code smell* como consequência das atividades de manutenção e evolução.	Tufano et *al*.(2015)

Subtema	Evidências	Referência
	Atividades relacionadas à implementação de novos recursos ou a melhoria dos já existentes tendem a apresentar *code smell*.	Tufano et *al*.(2015)
	A presença dos padrões de projetos inibe o surgimento de *code smell*.	Tufano et *al*.(2015)
Padrões de Design	O uso de padrões de projeto em produtos de *software* fornece melhorias significativas.	Bartosz Walter e Tarek Alkhaeir (2016)
	Code smell são sintomas relacionados a problemas de *design*	Bartosz Walter e Tarek Alkhaeir (2016)
	Code smell dificultam as futuras manutenções em *software*.	Bartosz Walter e Tarek Alkhaeir (2016)
Identificação de Dívida Técnica	O uso de ferramentas para a detecção de dívida ainda apresentam pouca sobreposição.	Zazworka et *al*., (2014)
	Violações de Acoplamento Disperso e de Modularidade são mais presentes em classes com maior propensão ao defeito.	Zazworka et *al*., (2014)
	Há uma forte relação entre as violações da Modularidade e classes que possuem tendência à mudança.	Zazworka et *al*., (2014)
	Code smell e alguns tipos de problemas de ASA não estão inter-relacionados.	Zazworka et *al*., (2014)
	As ferramentas CodeVizard, FindBugs e ASA Issues podem ser utilizadas para o processo de identificação de dívida de defeitos e de *design*.	Zazworka et *al*., (2013)
	O modelo CVM-TD apoia o processo de identificação de DT através dos comentários presentes em um código fonte.	Farias et *al*., (2015)
	Identificação de DT através de ferramentas podem relatar anomalias sem importância.	Zazworka et *al*., (2013)

Subtema	Evidências	Referência
	O nível de percepção entre os *stakeholders* é diferenciado.	Zazworka et *al*., (2013)

Fonte: Elaboração própria do autor desta dissertação (2017).

4.4.8.2.1 Aspectos Humanos

Desenvolvedores recém contratados não são necessariamente responsáveis pela introdução de dívida, mas desenvolvedores que possuem altas cargas de trabalho e pressão são os mais propensos. Outro fator que contribui para o surgimento de uma DT é a presença de *code smells* (TUFANO *et al.*, 2015).

Davide Falessi e Alexander Voegele (2015) afirmam que as regras de qualidade diferem com relação ao interesse dos desenvolvedores, uma vez que estes apresentam uma diferenciação com relação à importância delas no processo de desenvolvimento de *software*. Além disso, classes que violam várias regras de qualidade são cinco vezes mais defeituosas que as classes que não violam qualquer regra.

Mamun *et al*. (2014) afirmam que os indivíduos pesquisados estão cientes da gravidade do *code smell* muitas vezes presente no seu próprio código fonte. Além disso, a falta de conhecimento não é a principal razão para o acúmulo da dívida de projeto. Ainda de acordo com o estudo, destacam-se outros fatores que contribuem de forma significativa para o acúmulo da dívida: a pressão com relação ao tempo para a entrega do *software*, integração entre *hardware* e *software*, refatoração incompleta, reutilização de códigos desenvolvidos por terceiros, legados ou abertos. Além disso, durante o experimente realizado, constatou-se que uma grande parcela da DT foi acumulada com a aquisição de códigos abertos disponíveis na internet. No entanto, os autores afirmam que a melhoria nos processos relacionados a testes e da integração de *hardware/software* aumentariam a qualidade geral do sistema.

Constantemente, desenvolvedores necessitam realizar alterações em produtos de *software* não desenvolvidos por eles. Portanto, faz-se necessário que os códigos presentes nestas aplicações sejam bem estruturados para que possam ser entendidos facilmente. Sendo assim, deve-se compreender o efeito da DT com relação ao esforço para a compreensão deste código. E, para isso, deve-se utilizar os *logs* dos desenvolvedores, pois estes permitem que as interações realizadas pelos desenvolvedores dentro de um código fonte possam ser

quantificadas, permitindo assim, estimar o esforço necessário para entendimento das classes de uma aplicação, como também investigar correlações potenciais entre o esforço para a compreensão e correções de defeitos em um código.

Singh *et al.* (2016) afirmam que duas métricas relacionadas às atividades dos desenvolvedores, provavelmente, são os melhores indicadores para *code smell*: a porcentagem de tempo gasto no arquivo durante todo o período de tempo registrado e o número total de sessões para este arquivo. No entanto, não há uma diferença significativa entre estas duas métricas. De acordo com os experimentos, a porcentagem de tempo gasto no arquivo durante todo o período de tempo está fracamente correlacionada com as métricas *Tight Class Cohesion* e *Weighted Method Count*, entretanto, não com a *Access to Foreign Data*. Sendo assim, há uma correlação do tempo gasto em um arquivo e as métricas estáticas, embora esta correlação seja considerada fraca, pouca ou nenhuma com as atividades dos desenvolvedores.

Com relação à análise de correlação entre as métricas de atividade dos desenvolvedores e as classes com propensão à mudança, há uma correlação moderada entre a porcentagem de tempo gasto no arquivo, bem como entre o número total de sessões e as classes com propensão à mudança. Logo, as classes com propensão às mudanças podem ser utilizadas para inibir as atividades dos desenvolvedores relacionadas à criação de dívida técnica.

De acordo com Zazworka *et al*. (2013), algumas dificuldades são encontradas para identificar e documentar os itens de DT, pois investe-se muito o tempo para realizar estas tarefas. Ainda de acordo com o estudo, os entrevistados concordaram que o montante dos juros e a probabilidade apresentavam as maiores dificuldades com relação a sua documentação. Portanto, a identificação inicial de itens DT pode ser realizada em um tempo razoável, mas os principais parâmetros financeiros da DT são difíceis de serem estimados.

A Tabela 7 descreve as evidências encontradas para cada um dos subtemas encontrados para o tema Aspectos Humanos, bem como os trabalhos associados a cada uma delas.

Tabela 7 - Evidências para o tema Aspectos Humanos

Subtema	Evidência	Referência
Desenvolvedores	Desenvolvedores que possuem altas cargas de trabalho e pressão são os mais propensos a inserirem dívida técnica.	Tufano *et al* (2015)
	Más escolhas com relação ao *design* e a implementação contribuem para o surgimento de uma dívida técnica.	Tufano et *al.*, (2015)

Subtema	Evidência	Referência
Regras de Qualidade	As regras de qualidade diferem com relação ao interesse dos desenvolvedores.	Davide Falessi e Alexander Voegele (2015)
	Classes que violam várias regras de qualidade são cinco vezes mais defeituosas.	Davide Falessi e Alexander Voegele (2015)
Causas para Dívida de Projeto	Desenvolvedores estão cientes da gravidade do termo *code smell*.	Mamun et *al*. (2014)
	A falta de conhecimento não é a principal razão para o acúmulo da dívida de projeto.	Mamun et *al*. (2014)
	Fatores que contribuem de forma significativa para o acúmulo da dívida: a pressão com relação ao tempo para a entrega do *software*, integração entre *hardware* e *software*, refatoração incompleta, reutilização de códigos desenvolvidos por terceiros, legados ou abertos.	Mamun et *al*. (2014)
Indicadores	Duas métricas relacionadas às atividades dos desenvolvedores, provavelmente, são os melhores indicadores para *code smell*: a porcentagem de tempo gasto no arquivo durante todo o período de tempo registrado e o número total de sessões para este arquivo.	Singh et *al.* (2016)
	Há uma correlação do tempo gasto em um arquivo e as métricas estáticas, embora esta correlação seja considerada fraca.	Singh et *al.* (2016)
	Classes com propensão às mudanças podem ser utilizadas para inibir as atividades dos desenvolvedores relacionadas à criação de dívida técnica.	Singh et *al.* (2016)
Documentação de Itens DT	O montante de juros e a probabilidade apresentavam as maiores dificuldades com relação a sua documentação.	Zazworka et *al*. (2013)

Fonte: Elaboração própria do autor desta dissertação (2017).

4.4.8.2.2 Manutenção

Melissa Dale e Clemente Izurieta (2014) realizaram um estudo com base nos efeitos causados pelas alterações no código, afetando assim a qualidade de um projeto. Através desta, constatou-se que não se deve preocupar somente com o crescimento "*grime*" ("sujeira") no código, como também, os tipos relacionados a este. *Grimes* temporários (TEAG, TEEG, TIG) podem ser mais caros do que os persistentes (PEAG, PEEG, PIG). Sendo assim, incialmente, faz-se necessário que o *grime* seja quantificado para depois ser incluído no controle da DT.

Para melhorar a qualidade das aplicações, uma das técnicas amplamente utilizadas é a refatoração. A refatoração é definida como o processo de melhoria do projeto de um sistema existente e, para isso, esta técnica altera a estrutura interna da aplicação sem alterar o comportamento externo. Entretanto, a maioria dos trabalhos de refatoração existentes não considera o impacto das refatorações na qualidade das futuras versões de um sistema (Fowler, 2009).

Visando solucionar este problema, Hanzhang *et al.* (2015) propuseram uma abordagem combinando o uso de engenharia de *software* baseada em pesquisa com séries temporais para recomendar boas estratégias de refatoração, a fim de gerir a dívida técnica. Através desta abordagem, verificou-se que todos os atributos relacionados à qualidade apresentaram melhoras nos sistemas analisados. Isso proporcionou uma melhoria na qualidade do projeto e dos atributos de valor em comparação com todas as abordagens existentes consideradas nas experimentações.

Para a identificação das atividades que deverão ser priorizadas ou adiadas com relação à refatoração, deve-se ser levado em consideração o impacto na qualidade, bem como o esforço necessário para a refatoração. Sendo assim, adiar uma refatoração não deverá implicar na perda de qualidade, bem como no aumento do seu custo. Além disso, existem métodos que podem ser utilizados para o processo de identificação das atividades que deverão ser priorizadas com relação à refatoração (HANZHANG *et al.* 2015).

Griffith *et al.* (2014) realizaram um estudo com o intuito de identificar se os principais métodos de estimativas para DT apresentam um nível de concordância. Além disso, a relação destas com os atributos do modelo de qualidade foi analisada.

De acordo com os experimentos, existe uma forte correlação entre três estimativas relacionadas à reutilização e a compreensão. Entretanto, os autores ressaltam que esse resultado era esperado de acordo com a pesquisa existente. Em uma outra análise, os resultados apresentados identificaram que não há uma relação entre os atributos de qualidade e os métodos para estimativas de dívida técnica, com exceção do método de estimativa (flexibilidade e eficácia). Por fim, nenhuma das principais estimativas da DT mostrou qualquer relação com o tamanho do sistema.

A Tabela 8 descreve as evidências encontradas para cada um dos subtemas encontrados para o tema manutenção, bem como os trabalhos associados a cada uma delas.

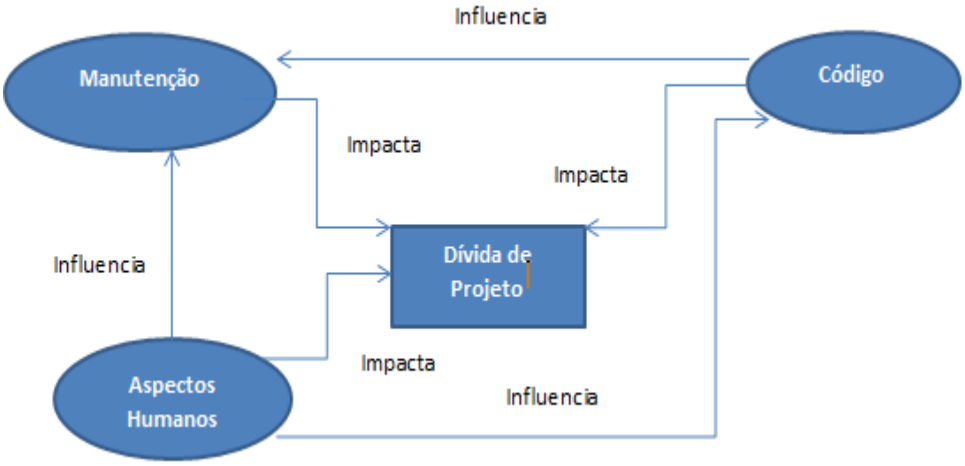
Tabela 8 - Evidências para o tema Manutenção

Subtema	Evidências	Referências
Alterações	*Grimes* temporários podem ser mais caros do que os persistentes.	Melissa Dale e Clemente Izurieta (2014)
Refatoração	Existem abordagens que melhoram os atributos relacionados a qualidade.	Hanzhang *et al.* (2015)
	Para a identificação das atividades que deverão ser priorizadas ou adiadas com relação à refatoração, deve-se ser levado em consideração o impacto na qualidade, bem como o esforço necessário para a refatoração.	Hanzhang *et al.* (2015)
	Existem métodos que podem ser utilizados para o processo de identificação das atividades que deverão ser priorizadas com relação à refatoração.	Hanzhang *et al.* (2015)
Estimativa	Existe uma forte correlação entre estimativas relacionadas à reutilização e a compreensão	Griffith *et al.* (2014)
	Há uma relação entre os atributos de qualidade e os métodos para estimativas de dívida técnica, com exceção do método de estimativa (flexibilidade e eficácia).	Griffith *et al.* (2014)
	Não existe relação entre o tamanho do sistema e as estimativas da dívida técnica.	Griffith *et al.* (2014)

Fonte: Elaboração própria do autor desta dissertação (2017).

A Figura 17 ilustra as relações encontradas ente os temas e o tópico central.

Figura 17 - Relações encontradas entre os temas e o tópico central



Fonte: Elaboração própria do autor desta dissertação (2017).

De acordo com a Figura 17, existe uma relação entre os temas encontrados onde, o tema **Código** impacta no tópico central dívida de projeto. Os comentários presentes em códigos fontes de uma aplicação estão associados à descoberta deste tipo de dívida. Geralmente, as *Good Class*, devido ao seu alto grau de acoplamento, apresentam mais defeitos. Além disso, as atividades relacionadas à implementação de novos recursos ou à melhoria dos já existentes tendem a apresentar *code smell*.

Para o tema **Aspectos Humanos,** o impacto está relacionado ao conhecimento acerca da DT entre os *stakeholders*, pois ainda não é consolidado e existe uma diferenciação no nível de percepção entre estes. Além disso, os desenvolvedores possuem dificuldade em realizar o processo de identificação e documentação do item de DT. Alguns fatores contribuem para o surgimento e o acúmulo deste tipo de dívida, dentre os quais: alta carga de trabalho e a pressão sofrida pelos desenvolvedores, bem como refatoração incompleta, reutilização de códigos desenvolvidos por terceiros, legados ou abertos e más escolhas com relação a *design* e implementação.

Por fim, o tema **Manutenção** apresenta impactos neste tipo de dívida. Durante a evolução do *software*, a presença de "*grimes*" nos projetos influencia na qualidade do produto.

Ainda de acordo com a figura 17, percebe-se que o tema **Código e Aspectos Humanos** influenciam no tema **Manutenção.** A presença de *code smell* e *good class* nos projetos está relacionada a existência de DT de projeto, além disso, manutenções corretivas e adaptativas podem contribuir com a presença de *code smell*. Por fim, o conhecimento acerca da DT ainda não é consolidado, assim como o nível de percepção entre os *stakeholders* é diferenciado, dificultando assim, o processo de identificação e documentação de DT. Já o **Aspectos Humanos** influencia no tema **Código**. A inserção de novas dívidas está relacionada, entre outros fatores, ao nível de conhecimento dos desenvolvedores e a discrepância deste conhecimento entre eles.

4.5 DISCUSSÃO

4.5.1 Implicações para Pesquisadores

A seguir são listadas algumas implicações dos resultados obtidos neste estudo para pesquisadores:

a) Apesar de vários estudos contarem com a participação de profissionais, poucos estudos *in vivo* têm sido realizados na área. Embora esse seja um cenário comum em estudos sobre engenharia de software, o autor deste trabalho acredita que mais avaliações *in vivo* deveriam ser realizadas na área, uma vez que DT é um tópico de pesquisa aplicado;

b) Existem algumas ferramentas que podem auxiliar no processo de identificação de DT, tais como: *Titan, Change10, SonarQube, Bug2, CodeVizard, FindBugs, ASA issues, SIG/T¨UViT's, Software Analysis Toolkit (SAT), AnaConDebt, DebtFlag, RE-KOMBINE, Extractor Jira, srcML Toolkit, W2RPTD, VCG-Cubo.* Entretanto, não foi identificado um estudo para caracterizar as ferramentas existentes e, por exemplo, o quanto elas diferem ou se complementam nos resultados;

c) Existem algumas dificuldades com relação a estimar, comunicar a natureza e a magnitude de uma DT aos envolvidos nos projetos, sejam estes técnicos ou não. Sendo assim, o autor deste trabalho acredita que o desenvolvimento de mecanismos que possibilitem o entendimento dos conceitos por parte destes facilitará a gestão da DT;

d) Existe uma imprecisão com relação à medição da DT *Self-Admitted*, este fato pode

ocorrer devido à falta de compreensão por parte dos desenvolvedores. Estudos poderiam ser realizados com o objetivo de investigar como atenuar esta falta de alinhamento entre a teoria e a prática.

4.5.2 Implicações para Profissionais

A seguir, têm-se algumas implicações desta revisão de literatura para profissionais:

a) Durante a análise dos artigos foram identificadas algumas ferramentas que foram utilizadas pelos autores dos trabalhos, dentre as quais: *Titan, Change10, SonarQube, Bug2, CodeVizard, FindBugs, ASA issues, SIG/T¨UViT's, Software Analysis Toolkit (SAT), AnaConDebt, DebtFlag, RE-KOMBINE, Extractor Jira, srcML Toolkit, W2RPTD, VCG-Cubo;*

b) A DT pode ser inserida intencionalmente ou não. Entretanto, uma dívida técnica quando é classificada como não intencional, normalmente, é considerada mais complexa;

c) Decisões relacionadas à DT, normalmente, não são quantificadas. Isso ocorre porque a DT não é absoluta, mas relativa a um conjunto de metas, requisitos e partes interessadas, dificultando assim, o processo de quantificação. Além disso, itens de dívida poderão sofrer mudanças no decorrer do projeto;

d) A falta de mecanismos que possibilitem um *feedback* eficiente com relação a inserção ou não e o pagamento de uma DT dificultam a compreensão dos envolvidos nos projetos de software com relação a decisão para adquirir ou não uma dívida;

e) Existem quatro categorias que representam os custos, são elas: custo de identificação DT, custo de atualização da lista de DT, custo de utilização da lista DT para o planeamento do *sprint* e custo de comunicação. Dentre esses custos, a identificação da DT representa a maior parte do custo total de gestão da DT;

f) As soluções sugeridas para evitar ou pagar uma dívida são: refatoração, uso de padrões de projeto, a melhoria da concepção da arquitetura e a priorização do pagamento de itens de dívida;

g) Os impactos causados por uma DT em sistemas legados são maiores e geram um custo maior quando comparados a sistemas não legados;

h) Códigos apresentam *code smell* desde o momento da sua criação. Desta forma, essa afirmação contradiz o senso comum de que eles são provenientes apenas da evolução do *software*;

i) A incursão de uma DT não está relacionada a novas contratações de funcionários pelas empresas, mas com a carga de trabalho atribuída a estes, bem como a pressão sofrida por estes com relação ao cumprimento dos prazos;
j) A falta de conhecimento não é a principal razão para o acúmulo de DT baseada em *code smell*. Outros fatores, como: a pressão com relação ao tempo para a entrega do *software*, integração entre hardware e *software*, refatoração incompleta, reutilização de códigos desenvolvidos por terceiros, legados ou abertos contribuem para o acúmulo da dívida.

4.6 LIMITAÇÕES DO ESTUDO

As limitações impostas ao estudo estão ligadas à disponibilidade dos artigos para *download*s provenientes das bibliotecas digitais utilizadas. Alguns artigos não estavam disponíveis e outros eram necessários realizar pagamentos para obter o acesso destes, principalmente, em se tratando da biblioteca *Springer*.

Embora a revisão sistemática seja um procedimento realizado de forma controlada e rigorosa, não podemos garantir que todos os estudos primários considerados relevantes tenham sido retornados durante sua execução. Além disso, não é possível afirmar que todos os estudos relevantes tenham sido selecionados com base nos critérios de inclusão. Por fim, a falta de um padrão estrutural em alguns artigos selecionados e a ausência de algumas informações importantes acerca do contexto ao qual o estudo foi submetido contribuem para as limitações da revisão da literatura executada.

4.7 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o planejamento e execução de uma revisão sistemática da literatura com o objetivo de caracterizar o estado atual das pesquisas relacionadas à DT de projeto que realizaram alguma avaliação experimental. O próximo capítulo irá apresentar as considerações finais deste trabalho.

Capítulo 5 – Considerações Finais

Este capítulo apresenta as considerações finais sobre o trabalho realizado, incluindo suas principais contribuições, limitações e próximos passos a serem realizados como resultado desta pesquisa.

5.1 CONSIDERAÇÕES FINAIS

Dívida Técnica descreve a relação entre os retornos a curto prazo, obtidos através do adiamento de alguma atividade relacionada ao processo de desenvolvimento de *software*, bem como as consequências a longo prazo deste adiamento. Assim, o conceito é utilizado para descrever situações nas quais a equipe de desenvolvimento adota medidas que impactam na qualidade do *software* com o objetivo de atender a uma necessidade momentânea.

A inserção de uma DT é proveniente, muitas vezes, das decisões adotadas pela equipe de desenvolvimento para atender prazos e, com isso, impacta na qualidade do *software.* No entanto, a DT deverá ser paga. Sua presença pode acarretar em problemas técnicos e financeiros, bem como, o aumento nos custos com relação à manutenção e evolução do *software*. Assim, é necessário a utilização de estratégias que possibilitem seu gerenciamento.

Por se tratar de um assunto relativamente novo, a DT tem despertado interesse na comunidade de engenharia de *software* e, com isso, diversos trabalhos têm sido realizados. Esta dissertação realizou uma síntese temática, com base nos trabalhos que realizaram alguma avaliação experimental sobre DT de projeto.

5.2 CONTRIBUIÇÕES

Além das contribuições para pesquisadores e profissionais apresentadas nas seções 4.5.1 e 4.5.2 desta dissertação, este trabalho traz mais duas contribuições para a área:

a) Mapeamento e caracterização dos estudos experimentais primários que foram realizados na área de DT;

b) Elaboração de mapa temático sobre a Dívida Técnica de Projeto, que é o tipo de dívida que tem sido mais discutido na literatura técnica.

5.3 LIMITAÇÕES

As principais limitações deste trabalho estão relacionadas ao estudo executado e que foi apresentado no capítulo 4:

a) Embora a revisão sistemática seja um procedimento realizado de forma controlada e rigorosa, não podemos garantir que todos os estudos primários considerados relevantes tenham sido retornados durante sua execução. Além disso, também não se pode garantir que todos os estudos relevantes para o desenvolvimento desta dissertação tenham sido selecionados com base nos critérios de inclusão;

b) Neste trabalho, optou-se por realizar a elaboração do mapa temático apenas para o tipo de dívida de projeto. Embora o mapa elaborado represente o escopo para o qual ele foi definido (dívida de projeto), contemplar os demais tipos de dívida seria interessante para se ter um mapa que represente as evidências geradas na área para os diferentes tipos;

c) Esta dissertação considerou estudos publicados até o ano de 2016, momento em que as buscas nas bibliotecas digitais foram realizadas. Sendo assim, possíveis novos trabalhos realizados em 2017 que possam evoluir os resultados e mapa temático elaborados não foram contemplados.

5.4 TRABALHOS FUTUROS

A seguir, tem-se uma lista com possíveis trabalhos futuros que podem ser realizados tendo esta dissertação como ponto de partida:

a) As implicações para pesquisadores apresentadas na seção 4.5.1 indicam diferentes lacunas que podem exploradas na área. Essa lista pode ser tomada como base para estimular novos trabalhos de pesquisa;

b) O mapa temático elaborado nesta dissertação considera apenas a dívida de projeto. É interessante dar continuidade ao trabalho através da organização da evidência sobre os demais tipos de dívida em mapas temáticos;

c) Manter o mapa temático atualizado é uma forma de manter organizada as evidências existentes na área. Assim, um outro possível trabalho futuro seria a replicação programada do protocolo elaborado.

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.

ALVES, N. S. R. **Organização do corpo de conhecimento sobre dívida técnica: tipos, indicadores, estratégias de gerenciamento e causas.** 2016. 163 f. Dissertação (Mestrado em Sistemas e Computação) – UNIFACS Universidade Salvador. Salvador, 2016.

ALZAGHOUL, E.; BAHSOON, R. CloudMTD: Using real options to manage technical debt in cloud-based servisse selection. In: INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT (MTD), 4., 2013**. Proceedings...** 2013. p. 55-62.

AMPATZOGLOU, A. et al. The financial aspect of managing technical debt: a systematic literature review. **Information and Software Technology**, v. 64, p.52-73, apr. 2015.

BOHNET, J.; DOLLNER, J. Monitoring code quality and development activity by software maps. In: INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT (MTD), 2011. **Proceedings...** 2011. p. 9-16.

BROWN, N. et al. Managing technical debt in software-reliant systems. In: WORKSHOP ON FUTURE OF SOFTWARE ENGINEERING RESEARCH, 2010. **Proceedings...** 2010. p. 47-52.

CODABUX, Z.; WILLIAMS, B. Managing technical debt: an industrial case study. In: INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT (MTD), 4., 2013. **Proceedings...** 2013. p. 8-15.

CRUZES, D. S.; DYBA, T. Recommended Steps for Thematic Synthesis in Software Engineering. In: INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, 2011. **Proceedings...** 2011. p. 275-284.

CRUZES, D. S.; DYBA, T. Research synthesis in software engineering: a tertiary study. **Information and Software Technology**, v.53, p. 440-455, 2011.

CRUZES, D. S. **Análise secundária de estudos experimentais em Engenharia de Software.** 2007, 275 f. Tese (Doutorado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e de Computação. São Paulo, 2007.

DYBA, T.; DINGSOYR, T.; HANSSEN, G. K. Applying systematic reviews to diverse study types: An experience report. In: INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, ESEM, 1., 2007. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2007.

DYBA, T.; KITCHENHAM, B.; JORGENSEN, M. Evidence-based Software Engineering for Practitioners. **IEEE Software**, v. 22, p. 58-65, 2005.

FOWLER, M. **Technical debt Bliki** [Blog]. Disponível em: <https://martinfowler.com/bliki/TechnicalDebt.html> Acesso em: 12 dez. 2003.

FOWLER, M. **Refactoring:** improving the design of existing code. Addison Wesley object technology series. [S.l.]: Addison-Wesley, 1999.

GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de pesquisa coordenado pela Universidade Aberta do Brasil – UAB/UFRGS e pelo Curso de Graduação Tecnológica – Planejamento e Gestão para o Desenvolvimento Rural da SEAD/UFRGS**. Porto Alegre: Editora da UFRGS, 2009.

GREENING, D. R. Release duration and enterprise agility. In: INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES (HICSS), 46., 2013, Hawaii. **Proceedings...** 2013.

GUO, Y. et al Domain-specific tailoring of code smells: an empirical study. In: ACM/IEEE INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 32., 2011. **Proceedings...** 2011.

GUO, Y.; SEAMAN, C. A portfolio approach to technical debt management. MTD11: OF THE WORKSHOP ON MANAGING TECHNICAL DEBT, 2., 2011. **Proceedings...** 2011.

KHOMH, F.; PENTA, M. D.; GUEHENEUC, Y. G. An exploratory study of the impact of code smells on software change-proneness. In: WORKING CONFERENCE ON REVERSE ENGINEERING (WCRE), 16., 2009. **Proceedings...** 2009.

KITCHENHAM, B. Procedures for Performing Systematic Reviews. Technical Report TR/SE-0401. Keele University, and Technical Report 0400011T.1, NICTA, 2004.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing systematic literature reviews in software engineering**. **Technical Report EBSE 2007-001.** [S.l.]: Keele University and Durham University, 2007.

KRUCHTEN, P.; NORD, R. L.; OZKAYA, I. Technical debt: from metaphor to theory and practice. **IEEE Software**, Published by the IEEE Computer Society, 2012.

LI, Z.; AVGERIOU, P.; LIANG, P. A systematic mapping study on technical debt and its management. **Journal of Systems and Software**, v.101, p.193–220, mar. 2015.

MIAN, P. G.; TRAVASSOS, G. H.; ROCHA, A. R. C. **A computerized infrastructure for supporting experimentation in software engineering.** 2005. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.7336&rep=rep1&type=pdf> Acesso em: 5 dez. 2016.

MORGENTHALER, J. et al. Searching for build debt: experiences managing technical debt at Google. In: MANAGING TECHNICAL DEBT (MTD), INTERNATIONAL WORKSHOP, 3., 2012. **Proceedings...** 2012.

NOBLIT, G. W.; HARE, R. D. **Meta-ethnography**: synthesizing qualitative studies. London: Sage Publications, 1988.

NORD, R. et al **In search of a metric for managing architectural technical Debt.** [S.l.]: WICSA/ECSA; IEEE CS, 2012.

OZKAYA, I. et al N. Managing technical debt in software development. In: INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT, HELD AT ICSE, 2., 2011. **Proceedings...** 2011.

PATERSON, B. L et al **Meta-study of qualitative health research**: a practical guide to meta-analisys and meta-synthesis. Thousand Oaks, California: Sage Publications, Inc., 2011. v. III.

PETERSEN, K. et al Systematic mapping studies in software engineering. In: INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 12., 2008, Italy. **Proceedings...** Italy: University of Bari, 2008.

POTDAR, A.; SHIHAB, E. An exploratory study on self-admitted technical debt. In: SOFTWARE MAINTENANCE AND EVOLUTION (ICSME), 2014. **Proceedings...** Italy: 2014. p. 91–100.

PRATES, L. C. L. **Aplicando síntese temática em Engenharia de Software**. 2016. Dissertação (Mestrado)-Universidade Federal da Bahia - UFBA. Salvador, 2016. Disponível em: <https://repositorio.ufba.br/ri/handle/ri/19380, 2015>. Acesso em: 12 jun. 2017.

RIBEIRO, L. F. et al. A strategy based on multiple decision criteria to support technical debt management. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS, 2017, Viena. **Proceedings...** 2017. v. 1.

SNIPES, W. et al. Defining the decision factors for managing defects: A Technical debt perspective. In: INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT (MTD), 2012. **Proceedings...** 2012.

SPÍNOLA, R. et al. Investigating technical debt folklore: shedding some light on technical debt opinion. In: MANAGING TECHNICAL DEBT (MTD), INTERNATIONAL WORKSHOP ON, 4., **Proceedings...** 2013.

THOMAS, J.; HARDEN, A. Methods for the thematic synthesis of qualitative research in systematic reviews. **BMC Med Res Methodol**, v. 8, 2008.

TOM, E.; AURUM, A.; VIDGEN, R. B. An exploration of technical debt. **Journal of Systems and Software** v.86, n.6, p.1498-1516, 2013.

TOM, E.; AURUM, A.; VIDGEN, R. T. A consolidated understanding of technical debt. In: EUROPEAN CONFERENCE ON INFORMATION SYSTEMS, ECIS, 20., 2012, Barcelona, Spain. **Proceedings...** 2012. p. 16.

VILLAR, A.; MATALONGA, S. Definiciones y tendencia de deuda técnica: un mapeo sistemático de la literatura. CIBSE13 - CONGRESSO IBERO-AMERICANO EM ENGENHARIA DE SOFTWARE, 2013, Montevideo, Uruguai. **Anais...** 2013. p 33-46.

WOHLIN, C. et al. **Experimentation in Software Engineering.** [S.l.]: Springer Berlin Heidelberg, 2012.

YAMASHITA, A.; MOONEN, L. To what extent can maintenance problems be predicted by code smell detection? An empirical study. In: INFORMATION AND SOFTWARE TECHNOLOGY, 2013. **Proceedings...** 2013.

ZAZWORKA, N. et al. Investigating the impact of design debt on software quality. In: INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT, 2., 2011, Waikiki, Honolulu, Hawaii, USA. **Proceedings...** 2011.

ZAZWORKA, N. et al. A Case study on effectively identifying technical debt. In: INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 17., 2013. **Proceedings...** 2013.

ZAZWORKA, N. B. et al Comparing four approaches for technical debt identification. **Software Quality Journal**, 2013.

APÊNDICE A - ARTIGOS IDENTIFICADOS NO MAPEAMENTO SISTEMÁTICO DA LITERATURA

Abad, Z. S. H., Karimpour, R., Ho, J., Didar-Al-Alam, S. M., Ruhe, G., Tse, E., Barabash, K., Hargreaves, I., Understanding the Impact of Technical Debt in Coding and Testing: An Exploratory Case Study, 2016.

Akbarinasaji, S., Bener, A. B., Erdem, A., Measuring the Principal of Defect Debt, 2016.

Bavota, G., Russo, B., A Large-Scale Empirical Study on Self-Admitted Technical Debt, 13th Working Conference on Mining Software Repositories, 2016.

Bellomo, S., Nord, R. L., Ozkaya, I., Popeck, M., Got Technical Debt? Surfacing Elusive Technical Debt in Issue Trackers, 13th Working Conference on Mining Software Repositories, 2016.

Brondum, J., Zhu, L., Visualising Architectural Dependencies, 2012.

Chen, J., Xiao, J., Wang, Q., Osterweil, L. J., Li, M., Perspectives on refactoring planning and practice an empirical study, 2016.

Codabux, Z., Williams, B., Managing Technical Debt: An Industrial Case Study, 2013.

Dale, M. R., Izurieta, C., Impacts of Design Pattern Decay on System Quality, 2014.

Eliasson, U., Martiniy, A., Kaufmanny, R., Odeh, S., Identifying and Visualizing Architectural Debt and Its Efficiency Interest in the Automotive Domain: A Case Study, 2015.

Ernst, N. A., Bellomo, S., Ozkaya, I., Nord, R. L., Gorton, I., Measure It? Manage It? Ignore It? Software Practitioners and Technical Debt, 2015.

Falessi, D., Voegele, A., Validating and Prioritizing Quality Rules for Managing Technical Debt: An Industrial Case Study, 2015.

Farias, M. A. F., Silva, A. B. D., Neto, M. G. D. M., Spínola, R. O., A Contextualized Vocabulary Model for Identifying Technical Debt on Code Comments, 2015.

Ganesh, L., Board Game as a tool to teach Software Engineering Concept – Technical Debt, 2014.

Griffith, I, Reimanis, D., Izurieta, C., Codabux, Z., Deo, A., Williams, B., The Correspondence between Software Quality Models and Technical Debt Estimation Approaches, 2014.

Guo, Y., Seaman, C., Silva, F. Q.B.D., Costs and obstacles encountered in technical debt management –A case study, 2016.

Guo, Y., Seaman, C., Gomes, R., Cavalcanti, A., Tonin, G., Silva, F. Q. B. D., Santos, A. L. M., Siebra, C., Tracking Techical Debt, 2011.

Ho, J., Ruhe, G., When-to-release decisions in consideration of technical debt, 2014.

Holvitie, J., Leppanen, V., Hyrynsalmi, S., Technical Debt and the Effect of Agile Software Development Practices on It – An Industry Practitioner Survey, 2014.

Kazman, R., Caiz, Y., Moz, R., Fengz, O., Xiaoz, L., Haziyevy, S., Fedaky, V., Shapochkay, A., A Case Study in Locating the Architectural Roots of Technical Debt, 37th IEEE International Conference on Software Engineering, 2015.

Klinger, T., Tarr, P., Wagstrom, P., Williams, C., An Enterprise Perspective on Technical Debt, 2011.

Li, Z., Liang, P., Avgeriou, P., Guelf, N., Ampatzoglou, A., An Empirical Investigation of Modularity Metrics for Indicating Architectural Technical Debt, 2014.

Li, Z., Liang, P., Avgeriou, P., Architectural Technical Debt Identification based on Architecture Decisions and Change Scenarios, 2015.

Mamun; M. A. A., Berger, C., Hansson, J., Explicating, Understanding and Managing Technical Debt from Self-Driving Miniature Car Projects, 2014.

Martini, A., Bosch, J., An Empirically Developed Method to Aid Decisions on Architectural Technical Debt Refactoring: AnaConDebt, 38th IEEE International Conference on Software Engineering Companion, 2016.

Martini, A., Bosch, J., Chaudron, M., Architecture Technical Debt: Understanding Causes and a Qualitative Model, 2014.

Martini, A., Bosch, J., The Danger of Architectural Technical Debt Contagious Debt and Vicious Circles, 2015.

Martini, A., Bosch, J.,Towards Prioritizing Architecture Technical Debt: Information Needs of Architects and Product Owners, 2015.

Martini, A., Bosch, J., Chaudron, M., Investigating Architectural Technical Debt accumulation and refactoring over time: A multiple-case study, 2015.

Mayr, A., Plosch, R., Korner, C., A Benchmarking-based Model for Technical Debt Calculation, 2014.

Mendes, T. S., Soares, H. F., Farias, M. A. F., Kalinowski, M., Mendonça, M., Spínola, R. O., Impacts of Agile Requirements Documentation Debt on Software Projects: A Retrospective Study, 2016.

Monteith, J. Y., McGregor, J. D., Exploring Software Supply Chains From a Technical Debt Perspective, 2013.

Nugroho, A., Visser, J., Kuipers, T., An Empirical Model of Technical Debt and Interest, 2011.

Potdar, A., Shihab, E., An Exploratory Study on Self-Admitted Technical Debt, 2014.

Schulte, L., Sajnani, H., Czerwonka, J., Active Files as a Measure of Software Maintainability, 2014.

Singh, V., Pollock, L. L., Snipes, W., Kraft, N. A., A Case Study of Program Comprehension Effort and Technical Debt Estimations, 2016.

Snipes, W., Robinson, B., Guo, Y., Seaman, C., Defining the Decision Factors for Managing Defects: A Technical Debt Perspective, 2012

Spínola, R. O., Zazworka, N., Vetrò, A., Seaman, C., Shull, F., Investigating Techinical Debt Folklore Shedding Some Light on Techinical Debt Opinion, 2013.

Soares, H. F., Alves, N. S. R., Mendes, T. S., Mendonça, M., Spínola, R. O., Investigating the Link between User Stories and Documentation Debt on Software Projects, 2015.

Tamburri, D. A, Kruchten, P., Lago, P., Vliet, H, V., Social debt in software engineering insights from industry, S/D.

Tamburri, D. A., Nitto, E. D., When Software Architecting Leads to Social Debt, 2015.

Tufano, M., Palomba, F., Bavota, G., Oliveto, R., Penta, D. D., Lucia, A. D., Poshyvanyk, D., When and Why Your Code Starts to Smell Bad, 2015.

Vetro, A., Using Automatic Static Analysis to Identify Technical Debt, 2012.

Walter, B., Alkhaeir, T., The relationship between design patterns and code smells An exploratory study, 2016.

Wang, H., Kessentini, M., Grosky, W., Meddeb, H., On the Use of Time Series and Search Based Software Engineering for Refactoring Recommendation, 2015.

Wehaibi, S., Shihab, E., Guerrou, L., Examining the Impact of Self-admitted Technical Debt on Software Quality, 2016.

Wiklund, k. Eldh, S., Sundmark, D., Lundqvist, K., Technical Debt in Test Automation, 2012.

Yli-Huumo, J., Maglyas, A., Smolander, K., How do software development teams manage technical debt? –An empirical study, 2016.

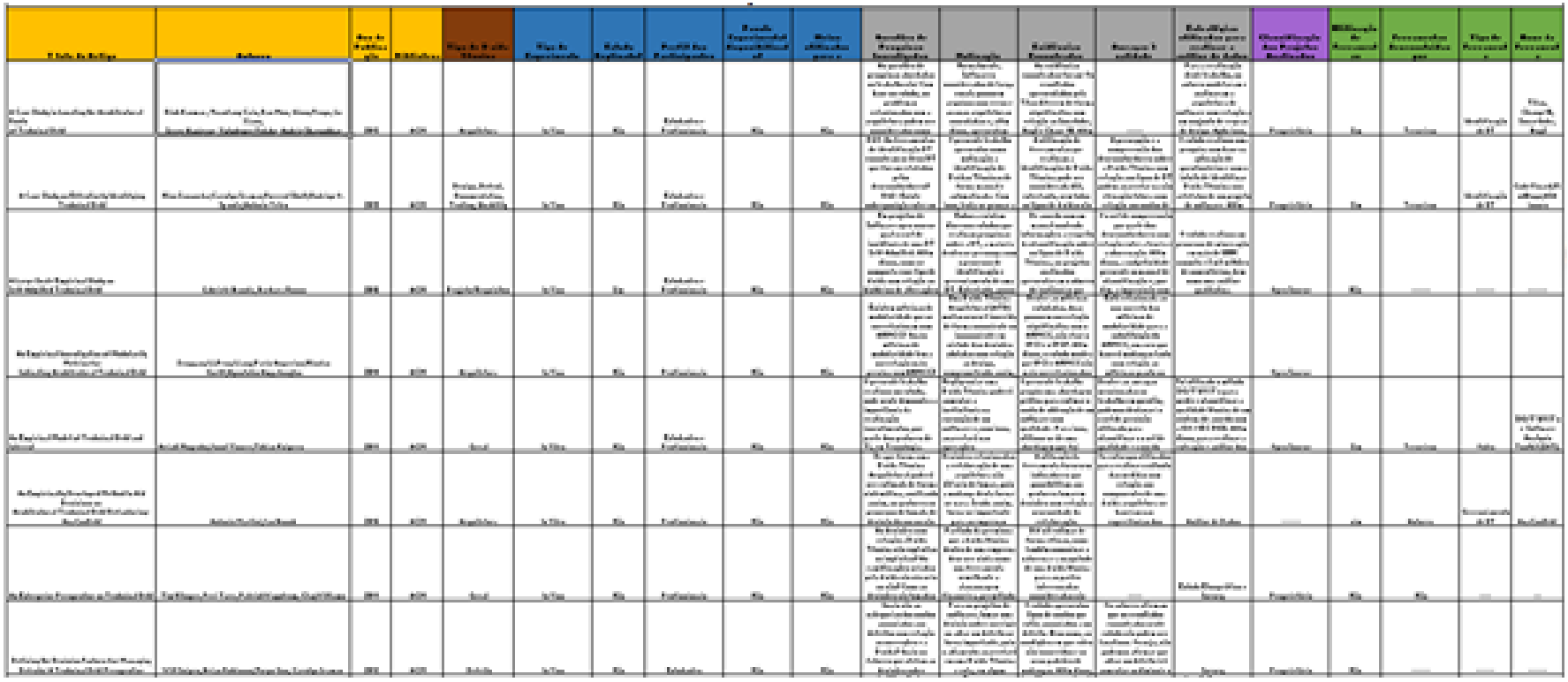
Zazworka, N., Vetro, A., Izurieta, C., Wong, S., Cai, Y., Seaman, C., Shull, F., Comparing four approaches for technical debt identification, 2014.

Zazworka, N., Seaman, C., Shull, F., Spínola, R. O., Vetro, A., A Case Study on Effectively Identifying Technical Debt, 2013.

Zazworka, N., Shaw, M. A., Shull, F., Seaman, C., Investigating the Impact of Design Debt on Software Quality, 2011.

Zazworka, N., Seaman, C., Shull, F., Prioritizing Design Debt Investment Opportunities, 2011.

APÊNDICE B - PLANILHA CONTENDO A EXTRAÇÃO DOS DADOS



Exploring Software Supply Chains From a Technical Debt Perspective	Yates Monteith; John D. McGregor	2013	ACM	Geral	In Vitro	Não	Estudantes	Não	Não	A utilização dos métodos de análise geram informações que poderão ser utilizadas com o intuito de auxiliar aos gestores na decisão para a inserção ou não de um	Empresas desenvolvedoras de software passaram a desenvolver componentes que podem ser reutilizáveis em outras aplicações. Pois, o	A análise proposta pelo trabalho identificou dependências entre os arquivos analisados e através de uma abordagem heurística estas informações poderão auxiliar os	....	A medição ocorreu através de um plugin presente no Sonar e, além disso, utilizou-se de gráficos para representar a dependência entre os arquivos analisados	....	Sim	Teoricos	Identificação de DT	SonarQube
Got Technical Debt? Surfacing Elusive Technical Debt in Issue Trackers	Stephany Bellomo, Robert L. Nord, Ipek Ozkaya, and Mary Popeck	2016	ACM	Geral	In Vivo	Não	Estudantes	Não	Não	Os desenvolvedores utilizam o termo Dívida Técnica quando discutem questões ou tarefas relacionadas aos seus rastreadores de problemas? Itens de dívida técnica podem	Relatar sobre a existência de uma Dívida Técnica e as consequências associadas a esta é de interesse comum aos pesquisadores e engenheiros de	A disseminação dos conceitos relacionados a Dívida Técnica para os desenvolvedores, tais como: Acumulado, acumulados e pagamento da dívida. Além disso, o estudo	Devido ao estudo ser caracterizado como exploratório, a inspeção manual dos dados poderá ter inserido conceitos chaves com relação ao objetivo analisado.	Survey e análise dos dados	Open Source	Não	....	....	....
Impacts of Agile Requirements Documentation Debt on Software Projects: A Retrospective Study	Thiago Souto Mendes;Henrique Frota Soares;Mário André de F. Farias;Marcos Kalinowski;Manoel Mendonça;Rodrigo Oliveira Spínola	2016	ACM	Documentação	In vitro	Não	Estudantes e Profissionais	Não	Não	Quais são os impactos de uma dívida técnica de documentação com relação ao esforço, tempo gasto e custo em projetos de software que utilizaram requisitos ágeis?	Ao contrário dos métodos tradicionais, as metodologias ágeis geralmente empregam ciclos iterativos curtos e contam com o conhecimento tácito dentro de uma equipe.	Nossa análise concentrou-se em dados de 132 tarefas de manutenção e evolução. Deste total, 65 foram relacionadas com a presença de dívida de documentação e foram	O estudo envolveu um projeto real desenvolvido por 10 desenvolvedores de software profissionais. Contudo, os resultados não podem ser	Análise dos Dados	Proprietário	Não	....	....	....
Impacts of Design Pattern Decay on System Quality	Melissa R Dale;Clemente Izurieta	2014	ACM	Projeto	In Vitro	Não	Estudantes e Profissionais	Não	Não	O presente trabalho visa um estudo sobre os efeitos provocados nas alterações sofridas em um software.	O desenvolvimento de um software deverá prever a escalabilidade deste,sem que o mesmo perca desempenho com as novas	Através da pesquisa desenvolvida pelos autores, constatou-se que não podemos nos preocupar somente com o crescimento "grime" ("sujeira") no código,	Atualmente, não existem referências que possam ser utilizadas para realizar a medição de uma Dívida Técnica (DT) em aplicações de	O presente trabalho utiliza o SonarQube para calcular a pontuação referente a Dívida Técnica para projetos desenvolvidos em java, com o intuito	**Não informou se é open ou não, apenas afirma que os testes foram feitos em implementações desenvolvidas**	Sim	Teoricos	Gerenciamento de DT	SonarQube
Investigating Technical Debt Folklore Shedding Some Light on Technical Debt Opinion	Rodrigo O. Spínola;Nico Zazworka;Antonio Vetrò;Carolyn Seaman;Forrest Shull	2013	ACM	Geral	In Vivo	Não	Estudantes e Profissionais	Não	Não	Sobre os folclores existentes sobre a Dívida Técnica, os entrevistados concordam ou não? Com relação ao folclore existente sobre a Dívida Técnica	Diversas declarações têm sido publicadas a respeito de dívida técnica. Desta forma, o trabalho propõe uma análise a respeito destas publicações com o intuito de identificar	As declarações mais concordadas com mostra que TD é um fator importante na gestão de projetos de software e não apenas um outro termo para "código ruim".	A quantidade de dados analisada foi pequena; a pesquisa foi traduzida para a língua portuguesa; a opção não sei disponível no questionário; pode ter sido tendenciosa de	Survey e Análise de dados	....	Não	....	....	....
Investigating the Impact of Design Debt on Software Quality	Nico Zazworka, Michele A. Shaw, Forrest Shull, Carolyn Seaman	2011	ACM	Projeto	In Vitro	Não	Estudantes e Profissionais	Não	Não	Existe uma relação entre o tamanho da classe e as mudanças sofridas por esta a qual justifica LOC-normalização? Existe uma relação entre o tamanho da classe com	Dívida técnica descreve situações nas quais desenvolvedores aceitam "sacrifícios" que afetam, por exemplo, a qualidade do software em detrimento da	Geralmente, as classes god são propensas a possuírem erros. Sendo assim, torna-se importante que este tipo de classe seja monitorada e gerenciada com	A análise foi realizada em dois sistemas comerciais, sendo assim, os resultados alcançados não podem ser considerados de uma forma geral. Além disso, code smell não é	Utilizou probabilidade de mudança e de defeito.	Proprietário	Não	....	....	....
Managing Technical Debt: An Industrial Case Study	Zadia Codabux, Byron Williams	2013	ACM	Geral	In Vivo	Não	Estudantes e Profissionais	Não	Não	Compreensão sobre como a Dívida Técnica é caracterizada, monitorada e priorizada, bem como a forma como cada fator influencia as decisões com relação ao pagamento desta dívida.	Em virtude dos desenvolvedores definirem a Dívida Técnica de acordo com o trabalho o qual foram designados, bem como a compreensão pessoal do termo. O trabalho identifica as melhores	A necessidade de analisar melhor os riscos para a decisão de gerar certos tipos de dívida tanto a curto como a longo prazo. Além disso, as empresas devem possuir equipes que	A falta de consenso entre os engenheiros que participaram da pesquisa com relação a terminologia referente a Dívida Técnica	A utilização de Survey para coleta de dados e posteriormente serem analisados e apresentados através de gráficos	Proprietário	Não	....	....	....
Measure It? Manage It? Ignore It? Software Practitioners and Technical Debt	Neil A. Ernst, Stephany Bellomo, Ipek Ozkaya, Robert L. Nord, and Ian Gorton	2015	ACM	Geral/Arquitetura	In Vivo	Não	Estudantes e Profissionais	Não	Não	Quais as práticas de gestão e ferramentas são utilizadas pelas empresas desenvolvedoras de software com relação a Dívida Técnica? Dentre as dívidas	A metáfora de dívida técnica é amplamente utilizada pelos desenvolvedores para relatar problemas relacionados a qualidade de software. No entanto, esta	Dívida técnica é útil e comumente entendida em um nível abstrato para transmitir sobre a acumulação de custos de software. Além disso, profissionais de software não estão de	O estudo foi baseado em empresas de grande porte que possuem sistemas complexos. Entretanto, a aplicação do estudo em equipes menores, auto-suficientes em	Survey e análise de dados	Proprietário	Não	....	....	....
On the Use of Time Series and Search Based Software Engineering for Refactoring Recommendation	Hanzhang Wang, Marouane Kessentini, William Grosky and Haytham Meddeb	2015	ACM	Projeto/Geral	In vitro	Não	Estudantes e Profissionais	Não	Não	A abordagem proposta melhora a qualidade? A abordagem proposta melhora os resultados das sugestões para a utilização da refatoração?	Para melhorar a qualidade dos sistemas de software, uma das técnicas amplamente utilizadas é a refatoração, sendo esta definida como o processo de melhoria	Após a aplicação das operações de refatoração propostas pela abordagem (NSGA-II), verificou-se que, em média, todos os atributos de qualidade foram melhorados em	....	Análise comparativa- Avaliação Empírica	Proprietário	Não	....	....	....

Prioritizing Design Debt Investment Opportunities	Nico Zazworka;Carolyn Seaman;Forrest Shull	2011	ACM	Projeto	In Vitro	Não	Estudantes e Profissionais	Não	Não	Como podemos priorizar e decidir quais god class devemos refatorar e, além disso, qual a estimativa de custo e o impacto desta refatoração?	Atualmente, a detecção de código smell foi utilizada para ajudar a identificar componentes que precisam ser refatorados. No entanto, as	O método apresentado pelos autores ajuda a identificar quais atividades deverão ser priorizadas para a realização da refatoração e quais destas devem ser	A falta de uma avaliação empírica para que possam construir evidências consistentes que determinem quais os god class exigem maiores esforços de	O método é baseado na análise de métricas e software de mineração de repositório	Proprietário	Não	....	....	....
Understanding the Impact of Technical Debt in Coding and Testing: An Exploratory Case Study	Zahra Shakeri Hossein Abad, Reza Karimpour, Jason Ho, S.M. Didar-Al-Alam, Guenther Ruhe, Edward Tse, Kevin Barabash, Ian Hargreaves	2016	ACM	Código/Testes	In Vivo	Não	Estudantes e Profissionais	Não	Não	Quais os impactos de uma dívida técnica em um projeto? Qual o nível de percepção da dívida técnica pelas equipes de desenvolvimento e de testes? Quais as sugestões dadas pelas	A falta de atenção ao realizar um controle da gestão de uma dívida teste e de desenvolvimento podem afetam custos, como também resultam em problemas de qualidade	As equipes de desenvolvimento e de testes, em geral, tem conhecimento do conceito de TD. Além disso, as soluções técnicas sugeridas são, por exemplo, o uso	A ausência de métricas para controlar e qualificar a pesquisa, bem como modelos e metodologias de comparação e validação.	Entrevista e análise qualitativa	Proprietário	Não	....	....	....
Using Automatic Static Analysis to Identify Technical Debt	Antonio Vetro	2012	ACM	Código	In Vitro	Não	Profissionais	Não	Não	A utilização de análise estática automática para realizar a identificação Dívida Técnica ao nível do código no que diz respeito às diferentes dimensões de	Dívida Técnica é um termo utilizado para traduzir decisões acerca do que pode ser adiado durante o processo de desenvolvimento de software em	Os resultados alcançados até agora não específicos para três características de qualidade: Funcionalidade-usabilidade, eficiência desempenho e	....	Análise Empírica	....	Não	....	....	....
Visualizing Architectural Dependencies	John Brondum;Liming Zhu	2012	ACM	Arquitetura	In Vitro	Não	Estudantes e Profissionais	Não	Não	Qual o nível de conhecimento arquitetônico com base nas dimensões relacionadas ao design, tais como: tipo de dependência, a fonte da dependência e o grau	Atualmente, as abordagens existentes para identificação de dívidas técnicas relacionadas à arquitetura de um software, baseiam-se na dependência entre	Modelos estimativos mais precisos; ajuda a identificar as fontes das dívidas em ambientes heterogêneos. Além disso, a abordagem utilizada permite uma	....	....	....	Não	....	....	....
When and Why Your Code Starts to Smell Bad	Michele Tufano , Fabio Palomba, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Andrea De Lucia, Denys Poshyvanyk	2015	ACM	Projeto	In Vivo	Não	Estudantes e Profissionais	Não	Não	Quando e por que se deve inserir code smell?	Atualmente, questões relacionadas sobre a gestão da dívida técnica passam a receber uma atenção significativa dos pesquisadores. Pois, diversos fatores contribuem para uma	Geralmente, os códigos apresentam code smell desde o momento da sua criação. Desta forma, essa afirmação contradiz o senso comum de que eles não provenientes da	As principais ameaças estão relacionadas as imprecisões / erros nas medições que realizamos.	Survey e Análise de Dados	....	Não	....	....	....
A Benchmarking-based Model for Technical Debt Calculation	Alois Mayr; Reinhold Plösch;Christian Körner	2014	IEEE	Geral	In Vitro	Não	Estudantes e Profissionais	Não	Não	Uma nova abordagem para o cálculo automático da dívida técnica	Dívida técnica tornou-se um termo popular na comunidade de engenharia de software, termo este utilizado para representar os riscos de desenvolvimento incorridos	Os custos de reparação resultantes dependem do nível de qualidade real dos projetos.,	Qualidade de Projetos avaliados	Análise dos Dados	Open Source	Não	....	....	....
An Exploratory Study on Self-Admitted Technical Debt	Aniket Potdar;Emad Shihab	2014	IEEE	Código/Projeto/Defeito	In Vitro	Não	Estudantes e Profissionais	Não	Não	Qual a quantidade de dívida técnica Self-Admitted existente nos projetos de software estudados? Por que a dívida técnica de Self-Admitted é introduzida em projetos de	Ao longo de um ciclo de vida de desenvolvimento de software, desenvolvedores comumente criam soluções incompletas ou temporárias que	Dentre os projetos analisados, existe uma margem entre 2,4 % e 31% em relação ao tipo de dívida técnica Self-Admitted encontrada em cada projeto. Além disso,	Para a execução deste trabalho, os autores realizaram um processo manual referente a identificação dos comentários que possuam a dívida	Análise Qualitativa	Open Source	Sim	Terceiros	Outro	srcML Toolkit
Architectural Technical Debt Identification based on Architecture Decisions and Change Scenarios	Zengyang Li;Peng Liang;Paris Avgeriou	2015	IEEE	Arquitetura	In Vitro	Não	Estudantes e Profissionais	Não	Não	Qual a eficiência da abordagem utilizada? A abordagem utilizada possui um nível aceitável quanto a sua usabilidade para a identificação de uma ATD em projetos de	A dívida técnica arquitetônica (ATD) é incorrida por decisões de design de forma intencional ou não, refletindo, comprometem atributos relacionados a	Os resultados apresentados pelos autores mostram que a abordagem proposta é útil e fácil de ser utilizada, além disso, suporta o planejamento de liberação e medição	Os participantes da pesquisa podem não ter o mesmo entendimento sobre as perguntas da entrevista que os investigadores. Além disso, os resultados deste estudo não podem	Survey	Proprietário	Não	....	....	....

Architecture Technical Debt: Understanding Causes and a Qualitative Model	Antonio Martini; Jan Bosch; Michel Chaudron	2014	IEEE	Arquitetura	In Vivo	Não	Estudantes e Profissionais	Não	Não	Quais os fatores que causam a acumulação de ATD? Quais são as tendências atuais na prática na acumulação e recuperação de ATD ao longo do tempo?	O equilíbrio entre a priorização de curto prazo com a capacidade de resposta a longo prazo tem sido um problema comum entre grandes empresas desenvolvedoras de software. Pois,	Interessa quando os casos de uso que deverão ser implementados, a evolução da arquitetura gera uma ATD, pressão com relação ao tempo para a entrega, bem como sanções,	....	Survey	Proprietária	Não	....	....	....
A Case Study of Program Comprehension Effort and Technical Debt Estimations	Vallary Singh; Lori L. Pollock;Will Snipes; Nicholas A. Kraft	2016	IEEE	Projeto	In Vivo	Não	Estudantes e Profissionais	Não	Não	Qual a medida que as técnicas de análise estática utilizadas para a detecção de dívida técnica nos arquivos com altos indicadores das atividades dos	Dívida técnica tem sido amplamente debatida entre a comunidade relacionada à área de Engenharia de Software, entretanto, é considerada um conceito novo. Desta	Como pode ser visto a partir da Tabela III, a porcentagem de tempo gasto nos arquivos sobre toda registrada a por cada de tempo é fracamente correlacionada com os	O estudo utiliza apenas o log de um desenvolvedor e uma única aplicação como estudo de caso. Além disso, utilizam três métricas estáticas que não responsáveis por	Análise de Dados	Proprietária	Não	....	....	....
Board Game as a tool to teach Software Engineering Concept - Technical Debt	Lakshmi Gaurab	2014	IEEE	Processo	In Vivo	Não	Profissionais	Não	Não	Os investigados que utilizaram o jogo Hard Choices compreenderam o conceito de Dívida Técnica? A aprendizagem através do jogo Hard Choices é	O método existente de ensino de disciplinas de engenharia de software é geralmente limitado pelo escopo, tempo e exposição a contextos do mundo real. Desta	Atribuir importância ao SDP, entenderam a consequência de pular um passo no SDP. Além disso, o estudo revelou que os alunos perceberam que a utilização do jogo Hard	O tamanho da amostra, ou seja, a quantidade de pessoas que participaram do experimento foi pequena. Além disso, os experimentos poderiam ter sido	Survey e Análise de Dados	Proprietária	Não	....	....	....
A Contextualized Vocabulary Model for Identifying Technical Debt on Code Comments	Mário André de Freitas Farias;André Batista da Silva;Manoel Gomes de Mendonça Neto;Rodrigo Oliveira Spínola	2015	IEEE	Código, Projeto, Defeito, Requisitos	In Vivo	Não	Estudantes e Profissionais	Não	Não	A Dívida Técnica presente em projetos de software poderá ser identificada através do modelo CVM-TD?	A identificação de dívida técnica (TD) é um passo importante para que esta possa ser gerida de forma eficaz. Neste contexto, um conjunto de indicadores tem sido	A fim de reduzir a diferença entre os comentários relevantes de acordo com o vocabulário e aqueles que podem indicar um item de TD.	O vocabulário criado pelo CVM-TD não pode ser considerado de forma plena como uma representação para identificação de comentários que contenham um	Análise de Dados	OpenSource	Não	....	....	....
Examining the Impact of Self-admitted Technical Debt on Software Quality	Sultan Wehaibi; Emad Shihab;Latifa Guerrou	2016	IEEE	Código, Projeto, Defeito	In Vitro	Não	Estudantes e Profissionais	Não	Não	Arquivos contendo SATD têm mais defeitos do que arquivos sem SATD? Será que os arquivos SATD têm mais defeitos após a introdução do SATD?	Dívida Técnica se refere a soluções incompletas ou temporárias que não permitem acelerar o desenvolvimento de software, a curto prazo. Entretanto,	O estudo concluiu que uma dívida técnica Self-admitted não impacta somente sobre defeitos, este tipo de dívida também torna o sistema mais difícil de ser mudado no futuro.	Para identificar SATD, usamos os comentários presentes nos códigos fontes. Em alguns casos, os desenvolvedores não podem adicionar comentários quando	Análise Quantitativa e Qualitativa	Proprietária	Não	....	....	....
Exploiting, Understanding and Managing Technical Debt from Self-Driving Miniature Car Projects	Md Abdullah Al Mamun;Christian Berger;Jorgen Hansson	2014	IEEE	Projeto	In Vitro	Não	Estudantes e Profissionais	Não	Não	Como surgiu e qual a evolução ao longo do tempo de uma dívida técnica para um carro em miniatura selfdriving que competiu duas vezes em uma competição	Dívida Técnica se refere a várias deficiências na concepção ou implementação de um sistema resultante de trocas durante o desenvolvimento de	A análise dos dados indica que os indícios pesquisados estão níveis da gravidade de termos code smell muitas vezes presentes no seu próprio código fonte. A falta de conhecimento	O tamanho da amostra que foi pesquisada, bem como a configuração específica. Desta forma, não se deve generalizar os resultados.	Survey e Análise de Dados	Proprietária	Sim	....	Outro	SonarQube
Identifying and Visualizing Architectural Debt and Its Efficiency Interest in the Automotive Domain: A Case Study	Ulf Eliasson ;Antonio Martini; Robert Kaufmann; Sam Odeh	2015	IEEE	Arquitetura	In Vitro	Não	Estudantes e Profissionais	Não	Não	Qual o impacto relacionado a um item ATD? Qual item relacionado a um ATD é típico em uma indústria automotiva? Como a visualização de uma ATD poderá auxiliar a	A Dívida Técnica Arquitetural recebeu recentemente a atenção da comunidade científica, como uma metáfora adequada para descrever soluções	Consideramos analisar identificar um item específico de ADT proveniente de violações da arquitetura, ATD Misplaced IC. Além disso, para Misplaced	O estudo apresentado foi realizado em única empresa, desta forma, limita-se a arquitetura do sistema que a empresa utiliza, não podendo ser generalizada. Além	Survey	Proprietária	Sim	Autores	Outro	VCG-Coba
Investigating the Link between User Stories and Documentation Debt on Software Projects	Henrique F. Soares; Nicolli S. R. Alves; Thiago S. Mendes; Manoel Mendonça;Rodrigo O. Spínola	2015	IEEE	Documentação	In Vivo	Não	Estudantes e Profissionais	Não	Não	Quais são as principais dificuldades quando se trabalha com requisitos ágeis? Quais são as dificuldades na identificação e gestão de requisitos com	A dívida técnica é uma metáfora que descreve o efeito de artefatos imaturos no desenvolvimento de software. Um de seus tipos é a dívida de documentação, que	Dentre as dificuldades relatadas na pesquisa ao se trabalhar com histórias de usuários, identificou-se oito problemas, dentre os quais os mais mencionados foram a	O conjunto de dificuldades identificadas neste estudo pode não ser suficientemente abrangente. Como também, não foi possível garantir que	Survey	....	Não	....	....	....

Technical Debt and the Effect of Agile Software Development Practices on It – An Industry Practitioner Survey	Johannes Holvitie; Ville Leppanen; Sami Hyrynsalmi	2014	IEEE	Geral	In Vivo	Não	Estudantes e Profissionais	Não	Não	Qual o nível de conhecimento acerca de dívida técnica? Como a dívida técnica se manifesta em seus projetos? E quais dos componentes de desenvolvimento de software Ágil são	Uma das principais razões para a popularidade de métodos de software Ágeis é a sua capacidade de funcionar em processos rápidos e ambientes com pouca exigência.	Os resultados mostram que há variações no conhecimento acerca da dívida técnica, embora estejam familiarizados com o conceito. Em relação às práticas Ágeis de desenvolvimento de	Os dados foram coletados em empresas de desenvolvimento com sede na Finlândia, desta forma, a cultura do mercado pode ter influenciado a pesquisa. Além disso, é possível que apenas	Survey	Proprietário	Não	....	....	....
Technical Debt in Test Automation	Kristian Wiklund; Sigrid Eldh; Daniel Sundmark; Kristina Lundqvist	2012	IEEE	Automação de Testes	In Vitro	Não	Estudantes e Profissionais	Não	Não	Quais práticas em uma organização de projeto de software são os principais contribuintes para a criação de dívida técnica na execução automática de teste?	A execução de testes automatizados é uma das estratégias mais populares e disponíveis para minimizar o custo de testes de software e também está se tornando um dos	Uma avaliação qualitativa dos achados indica que as principais áreas de melhoria neste caso são os campos de concepção de interação e princípios gerais de	....	Avaliação Qualitativa	Proprietário	Não	....	....	....
The Correspondence between Software Quality Models and Technical Debt Estimation Approaches	Isaac Griffith; Derek Reimanis; Clemente Izurieta; Zadia Codabux; Ajay Deo; Byron Williams	2014	IEEE	Geral/Projeto	In Vitro	Não	Estudantes e Profissionais	Não	Não	Analisar como o método SonarQube TD-Plugin, o Método CAST e o método Marinescu estimam a DT com relação à qualidade?	A dívida técnica tornou-se recentemente uma grande preocupação na indústria de software. Embora tenha sido demonstrado que a dívida técnica tem um efeito adverso sobre a	Com exceção de um método de estimativa (flexibilidade e efetividade), não houve relação observável entre os atributos de qualidade e as estimativas de dívida	Não podemos ter a certeza de que as nossas conclusões serão válidas para outros domínios e aplicações, tornando assim, a generalização uma ameaça. Além	Análise de correlação e de regressão linear múltipla	Open Source	Não	....	....	....
The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles	Antonio Martini; Jan Bosch	2015	IEEE	Arquitetura	In vitro	Não	Estudantes e Profissionais	Não	Não	Quais são os itens de dívida técnica arquiteturais que geram termos de esforços para serem pagos futuramente? Quais são os efeitos desencadeados por tais	Um problema conhecido em grandes empresas de software é equilibrar a priorização de curto prazo com a viabilidade a longo prazo. Especificamente, as	Os itens de DT que geram grandes esforços com relação ao seu pagamento estão relacionados às Violações de dependência e desconhecimento; Não	Com a utilização de um contexto real, o qual apresenta uma inter-relação complexa de variáveis que talvez não tenhamos levado em consideração, influenciando assim,	Análise Qualitativa	Proprietário	Não	....	....	....
Towards Prioritizing Architecture Technical Debt: Information Needs of Architects and Product Owners	Antonio Martini; Jan Bosch	2015	IEEE	Arquitetura	In Vivo	Não	Estudantes e Profissionais	Não	Não	Qual é a informação necessária que deverá ser utilizada pelos desenvolvedores e arquitetos de software com relação à priorização de uma ATD com relação ao	Dívida Técnica Arquitetural é uma metáfora para representar soluções relacionadas à arquitetura que podem causar economia com relação ao esforço e	Os aspectos de priorização mais relevantes são o lead time, custos de manutenção e riscos. Além disso, os principais efeitos causados por uma ATD	O número de entrevistados considerado pequeno e, com isso, a generalização.	Survey e Análise Qualitativa	Proprietário	Não	....	....	....
Tracking Technical Debt	Yuepu Guo; Carolyn Seaman; Rebeka Gomes; Antonio Cavalcanti; Graziela Tonin; Fabio Q. B. da Silva; André L. M. Santos; Clauirton Siebra	2011	IEEE	Geral	In Vitro	Não	Estudantes e Profissionais	Não	Não	Como e em qual medida a dívida técnica afeta um projeto de software? A simulação da decisão sobre a inserção da dívida técnica apresentou benefícios com relação a	A metáfora da dívida técnica é cada vez mais usada para descrever o efeito de atrasar certas tarefas de manutenção projetos de software. Os profissionais	O item de dívida técnica analisado teve a sua inserção devido à limitação com relação ao tempo, além disso, as alterações impostas ao sistema não estavam claras. A análise, com	A simplicidade excessiva da abordagem, além disso, o modelo em questão não tem incorporado os fatores de negócios.	Estimativa e análise dos dados	Proprietário	Não	....	....	....
Validating and Prioritizing Quality Rules for Managing Technical Debt: An Industrial Case Study	Davide Falessi; Alexander Voegele	2015	IEEE	Projeto	In Vivo	Não	Estudantes e Profissionais	Não	Não	Os desenvolvedores consideram as regras de qualidade importantes? As regras de qualidade são igualmente importantes?	Um grande problema no uso de analisadores estáticos para gerenciar, monitorar, controlar e raciocinar sobre a dívida técnica é que os projetos industriais têm uma	Classes que violam várias regras de qualidade são cinco vezes mais defeituosas que as classes que não violam qualquer regra. Além disso, os desenvolvedores	Os resultados apresentados referem-se a uma única empresa.	Análise Empírica e Qualitativa	Proprietário	Não	....	....	....
When Software Architecting Leads to Social Debt	Damian A. Tamburri; Elisabetta Di Nitto	2015	IEEE	Pessoas	In Vitro	Não	Profissionais	Não	Não	A dívida social está associada às arquiteturas de software, além disso, como podem ser medidas?	Dívidas sociais e técnicas representam o estado de organizações de desenvolvimento de software como resultado de decisões acumuladas. No caso da dívida social, as	O presente trabalho utiliza o DAHLIA (Análise de Incomunicabilidade de Nível de Arquitetura Orientada à Dívida) para realizar uma medição com relação a	Um único projeto foi utilizado	Análise de Dados	Proprietário	Não	....	....	....

When-to-release decisions in consideration of technical debt	Jason Ho; Guenther Ruhe	2014	IEEE	Geral	In Vitro	Não	Estudantes e Profissionais	Não	Não	A ferramenta W2RPTD estima de forma efetiva a dívida técnica auxiliando os gestores quanto a liberação do produto de software?	Liberação de software é essencial quando se trata de um produto competitivo no mercado. No entanto, existem fatores dentre os quais podemos destacar: pedidos de	A utilização da ferramenta W2RPTD possibilita aos gerentes de software informações que demonstram alternativas sobre as implicações da	A modelagem e a mensuração da dívida técnica, pois qualificar a TD é considerada complexa, especialmente no caso de cenários de testes integrados de grandes	....	Proprietário	Sim	Autores	Outra	W2RPTD
Active Files as a Measure of Software Maintainability	Lukas Schulte; Hitesh Sajnani; Jacek Czerwonka	2014	Scopus	Versão/Defeito	In Vitro	Não	Estudantes e Profissionais	Não	Não	Existem conjuntos ativos e, em caso afirmativo, qual o tamanho deles em relação ao tamanho geral da base de código? Que tipos de arquivos de origem	Denomina-se arquivos de origem aqueles que são alterados com muita frequência, coletados descobertos através do histórico de versões e na classificação de	Os arquivos ativos constituem apenas entre 2% e 8% do tamanho total do sistema, contribuem com 20% e 40% de alterações nos arquivos do sistema e	Estudo empírico em engenharia de software é difícil porque além de variáveis independentes, o processo depende de muitas variáveis	Análise Qualitativa e Survey	Proprietário	Não	....	....	....
How do software development teams manage technical debt? - An empirical study	Jesse Yli-Huumo; Andrey Maglyas; Kari Smolander	2016	Scopus	Geral	In Vivo	Não	Estudantes e Profissionais	Não	Não	Como as equipes de desenvolvimento de software gerenciam dívida técnica? Quais atividades TDM são usadas nas equipes de desenvolvimento estudadas? Quais	Dívida técnica (TD) é uma metáfora para a tomada de atalhos ou soluções alternativas em decisões técnicas para obter benefícios a curto prazo em relação a liberação de	No cenário da DT, todas as equipes de desenvolvimento usaram refatoração, reescrita ou re-design como o principal processo para resolver os	O número de equipes de desenvolvimento de software estudadas e o fato de que todas elas eram da mesma organização.	Survey e Análise de Dados	Proprietário	Sim	Terceiros	Outra	SonarQube e JIRA
Investigating Architectural Technical Debt accumulation and refactoring over time: A multiple-case study	Antonio Martini; Jan Bosch; Michel Chaudron	2015	Scopus	Arquitetura	In Vitro	Não	Estudantes e Profissionais	Não	Não	Quais fatores causam a acumulação de ATD? Como a ATD é acumulada e refatorada ao longo do tempo? Quais possíveis estratégias de refatoração podem ser	Um problema conhecido em grandes empresas de software é equilibrar a priorização de curto prazo com a velocidade de entrega de recursos de longo prazo.	Os fatores que constantemente causam a dívida técnica arquitetural são: Incertezas do caso de uso em estágios iniciais, Evolução do negócio, Pressão do	Possivelmente existem outros fatores que deverão ser levados em consideração e que não foram listados no trabalho e que causariam uma ATD, além disso, a	Indução analítica	Proprietário	Não	....	....	....
Measuring the Principal of Defect Debt	Shirin Akbarinasaji; Ayse Basar Bener; Atakan Erdem	2016	Scopus	Defeito	In Vitro	Não	Estudantes e Profissionais	Não	Não	O modelo de previsão estima os bugs que não propagam a dívida?	Devido ao orçamento apertado e cronograma, as equipes de desenvolvimento de software podem não resolver todos os bugs existentes nos	A regressão KNN supera a regressão linear simples com o poder preditivo (R2) variando entre 74% e 85%.	O modelo de regressão KNN proposto pode ser usado somente para sistemas evolutivos de grande porte, onde o desempenho da equipe de desenvolvimento é estável e previsível.	....	Open Source	Não	....	....	....
Costs and obstacles encountered in technical debt management – A case study	Yuepu Guo; Carolyn Seaman; Fabio Q.B. da Silva	2016	Science Direct	Geral	In Vitro	Não	Estudantes e Profissionais	Não	Não	Quais são os custos de uma gestão explícita de DT utilizando a abordagem proposta? Quais são os obstáculos à gestão explícita de DT? O que contribui para o desvio	Dívida técnica é uma metáfora que caracteriza o efeito de artefatos de software imaturos. Os custos e benefícios da DT, juntamente com a incerteza de seu	Existem quatro categorias que representam os custos, são elas: custo de identificação DT, custo de atualização da lista DT, custo de utilização da lista DT	Os participantes da pesquisa não estavam muito motivados, pois ofereceram resistência quanto a adoção dos conceitos relacionados a DT. Sendo assim, os	Análise Qualitativa	Proprietário	Não	....	....	....
The relationship between design patterns and code smells: An exploratory study	Bartosz Walter; Tarek Alkhaeir	2016	Science Direct	Projeto	In Vitro	Não	Estudantes e Profissionais	Não	Não	Quais são as classes que participam dos padrões de design e possuem menos smell do que outras classes? Durante a evolução do sistema, o número de classes que possuem	Padrões de design representam soluções genéricas recomendadas para vários problemas de design, enquanto odores de código são sintomas de problemas	Classes que participam em padrões de design parecem exibir cod smell com menos frequência do que outras classes. A proporção entre o número relativo de	Está relacionada à escolha das ferramentas usadas para coletar dados que podem afetar os resultados. Além disso, para computar o número de classes os	Análise de Dados	Open Source	Sim	Autores	Identificação de DT	InCode
Comparing four approaches for technical debt identification	Nico Zazworka; Antonio Vetro; Clemente Izurieta; Sunny Wong; Yuanfang Cai; Carolyn Seaman; Forrest Shull	2014	Springer	Arquitetura, Projeto, Código	In Vitro	Não	Estudantes e Profissionais	Não	Não	Dentre as técnicas utilizadas para detecção de DT, estas relatam os mesmos problemas nas classes analisadas? Quais os indicadores DT correlacionados com os indicadores de	Os sistemas de software acumulam dívida técnica (TD) quando metas de curto prazo no desenvolvimento de software são negociadas para metas de longo prazo.	As saídas das quatro abordagens têm muito pouca sobreposição e, portanto, apontam para diferentes problemas no código-fonte. As violações do Acoplamento Disperso e da Modularidade	A escolha com relação a métrica de tamanho, bem como a não normalização das medidas por tamanho.	Funções estatísticas e de significância	Open Source	Não	....	....	....

Perspectives on refactoring planning and practice an empirical study	Jie Chen; Junchao Xiao; Qing Wang; Leon J. Osterweil; Mingshu Li	2016	Springer	Geral	In Vivo	Não	Estudantes e Profissionais	Não	Não	Como uma equipe de desenvolvimento planeja para cada tipo de tarefa? Qual a perspectiva das pessoas que desempenham papéis diferentes no	O desenvolvimento iterativo promove cada vez mais incorporar modificações de projeto e refatoração cada uma para manter a qualidade do código mesmo em ambientes	O estudo encontrou um forte acordo entre todos os papéis sobre a importância da refatoração, e acordo sobre os efeitos negativos sobre a agilidade de adiamento	Opiniões subjetivas durante as pesquisas	Survey	Proprietário	Não	....	....	....
Social debt in software engineering insights from industry	Damian A Tamburri; Philippe Kruchten; Patricia Lago; Hans van Vliet	S/D	Springer	Pessoas	In Vivo	Não	Estudantes e Profissionais	Não	Não	Quais são os fatores em torno da dívida social durante o ciclo de vida do software? Existem Padrões nos referidos fatores? Eles podem ser mitigados?	A dívida social é análoga à dívida técnica de muitas maneiras: representa o estado das organizações de desenvolvimento de software como resultado de decisões "acumuladas". No caso	A Dívida Social não pode ser atribuída a nenhum artefato de software ou processo de desenvolvimento e operação, embora seja uma propriedade emergente da própria comunidade de	Os resultados podem ser específicos para o contexto onde foi aplicada a avaliação.	Pesquisa Qualitativa	Proprietário	Não	....	....	....