DOI: 10.5748/19CONTECSI/MSC/ISM/7124

QUAIS ESTRATÉGIAS SÃO UTILIZADAS PARA PAGAMENTO DE DÍVIDA TÉCNICA DE REQUISITOS EM DESENVOLVIMENTO DE SOFTWARE?

Napoleão Verardi Galegale ; https://orcid.org/0000-0003-2228-9151
Centro Paula Souza

Deborah de Assis Pereira dos Santos ; https://orcid.org/0000-0001-7431-5521
Centro Paula Souza

PROJETO DE: MSC (Mestrado)

WHAT STRATEGIES ARE USED FOR THE PAYMENT OF REQUIREMENTS DEBT IN SOFTWARE DEVELOPMENT?

QUAIS ESTRATÉGIAS SÃO UTILIZADAS PARA PAGAMENTO DE DÍVIDA TÉCNICA DE REQUISITOS EM DESENVOLVIMENTO DE SOFTWARE?

ABSTRACT
The concept of technical debt in software development emerged by Cunningham (1992). Since then, this topic has been discussed and explored by several authors. There are types of technical debt corresponding to the phases of the software development cycle, such as requirements debt. This type of debt was initially mentioned by Ernst (2012), approximately 20 years after the introduction of the technical debt metaphor. This article aims to identify which are the strategies for payment requirements debt since the advent of the concept. The method used was literature review and content analysis with the application based on the PRISMA-P protocol, through descriptive research. The results, so far, reveal that there is still little discussion on the subject of technical debt of requirements. The main approach of the research refers to the types of technical debt of code and architecture.

Keywords: *Technical Debt, Payment, PayOff, Requirements, Software.*

RESUMO
O conceito sobre dívida técnica em desenvolvimento de software foi mencionado preliminarmente por Cunningham (1992). Desde então, esse tema vem sendo discutido e explorado por diversos autores. Há tipos de dívida técnica correspondentes com as fases do ciclo de desenvolvimento do software, tal como a dívida técnica de requisitos. Esse tipo de dívida foi mencionado inicialmente por Ernst (2012), aproximadamente 20 anos após a apresentação da metáfora de dívida técnica. Este artigo tem como objetivo identificar quais são as estratégias para pagamento de dívida técnica de requisitos desde o advento do conceito. O método utilizado foi revisão da literatura e análise de conteúdo com a aplicação baseada no protocolo PRISMA-P, mediante pesquisa do tipo descritiva. Os resultados, até o momento, revelam que ainda há pouca discussão sobre o tema de dívida técnica de requisitos. A principal abordagem das pesquisas, é referente aos tipos de dívida técnica de código e arquitetura.

Palavras-chave: Dívida Técnica, Requisitos, Pagamento, Pagar, Software.

INTRODUÇÃO

No ano de 1992, o conceito de dívida técnica (DT) foi introduzido por Cunningham, fazendo uma alusão ao conceito da dívida financeira. A DT pode incorrer por meio de uma decisão em obter benefícios de curto prazo (aumento da produtividade e/ou menor custo) em troca da qualidade do software. Isto pode ocasionar entregas de artefatos imaturos e incompletos.

Da mesma forma como uma dívida financeira, a DT pode gerar juros. Quanto mais tempo se posterga o pagamento da dívida, chances são dos juros se tornarem maiores, acarretando prejuízos de manutenção do sistema. De acordo com a classificação de LI et al. (2015) há 10 tipos de DT que são de: arquitetura, design, código, teste, construção (*build*), documentação, infraestrutura, versionamento, defeito e requisitos.

A literatura traz ênfase de estudos sobre o tipo DT de código, contudo esse mesmo cenário é o oposto quando se trata de DT de requisitos. O levantamento de requisitos é uma etapa importante no ciclo de desenvolvimento de software, onde as necessidades do usuário são capturadas. Com isso, a equipe de desenvolvimento pode transformar os requisitos em funcionalidades do sistema. Requisitos de software inadequados e mal definidos é um dos fatores que causa falha nos projetos (McManus12 et al., 2007).

Após 20 anos do surgimento do conceito de DT, Ernst (2012) fez menção sobre o conceito de dívida técnica de requisitos, definindo como a distância entre a especificação ótima de requisitos e a implementação real do sistema. Os requisitos fazem parte do sucesso do projeto, uma vez que são a representação em sistema dos anseios do usuário.

Segundo McManus12 et al. (2007), requisitos de software inapropriados ou mal definidos podem responder por 35% das falhas dos projetos. Compreendendo os requisitos como fator de sucesso ou fracasso em um projeto, o tema de dívida técnica de requisitos possui relevância nos projetos, por isso a importância em estudar estratégia de pagamento deste tipo de dívida. Menos dívida técnica de requisitos pode contribuir com o sucesso dos projetos.

KRUCHTEN et al. (2015), sugerem que a dívida técnica é acumulada durante todas as fases de desenvolvimento do software, como por exemplo, análise de requisitos, arquitetura e implementação e por isso, a DT deve ser monitorada e gerenciada durante todo o ciclo de vida do software.

Diante deste contexto, este artigo tem como principal objetivo responder a seguinte questão de pesquisa: Quais as estratégias utilizadas para o pagamento de dívida técnica de requisitos em desenvolvimento de software? Para atingir este objetivo principal, foram determinados os seguintes objetivos específicos: realizar a revisão da literatura, complementada com a bibliometria e analisar o conteúdo dos artigos resultantes da bibliometria. E a partir desta questão de pesquisa, apresentam-se duas proposições:

- **P1:** Não há estratégias para pagamento de dívida técnica de requisitos em desenvolvimento de software.
- **P2:** Há estratégias para o pagamento de dívida técnica de requisitos em desenvolvimento de software.

Esse estudo faz parte de uma pesquisa maior que visa a oportunidade de desenvolvimento de uma estratégia de pagamento de dívida técnica de requisitos para gestão deste tipo de dívida técnica. Este artigo é o início desse ensejo.

FUNDAMENTAÇÃO TEÓRICA

No ano de 1992, o tema dívida técnica (DT) foi apresentado por Cunningham, fazendo uma alusão ao conceito de dívida financeira. Desde então, diversos autores vêm explorando sobre essa definição. McConnell (2008) trouxe a classificação de DT como intencional e não intencional. A DT intencional é dívida técnica que se assume conscientemente como ferramenta estratégica, onde os benefícios são maiores do que a consequência da DT contraída no momento. Já a DT não intencional é quando a dívida técnica é gerada de forma inconsciente e acidental.

Fowler (2009) complementou o conceito de McConnell, incluindo mais duas classificações: a DT prudente e imprudente. Desta forma, tem-se formado o quadrante da DT com os tipos: intencional e imprudente, intencional e prudente, não intencional e imprudente, não intencional e prudente. A DT geralmente ocorre em projetos de software quando há a necessidade de escolher entre manter os padrões de qualidade do sistema e colocar o software em funcionamento no menor tempo possível, utilizando o mínimo de recursos. Esses itens de DT podem ter que ser pagos com juros posteriormente no projeto (ALVES et al., 2016).

A DT, em geral, demonstra que a equipe de desenvolvimento escolheu um atalho para obter benefícios de curto prazo. Isto pode prejudicar o software a longo prazo, visto que há consideráveis possibilidades de manutenção constante para mantê-lo operante. Uma das características da dívida técnica é sua natureza interdisciplinar, visto que combina elementos da teoria financeira e da engenharia de software (Ampatzoglou et al., 2015).

Dívida Técnica de Requisitos

ALVES et al. (2016), sugerem que pode haver dívida técnica em todo o ciclo de vida do projeto. Assim, garantir somente a qualidade do código fonte do projeto não é a única maneira de melhorar a qualidade do projeto. No entanto, apesar de já ter catalogado vários tipos de DT, grande parte do trabalho da literatura se concentra em problemas existentes em código fonte. Se não bem gerenciada, a dívida técnica faz com que os projetos enfrentem problemas técnicos e financeiros relevantes (KRUCHTEN et al., 2012).

Somente após 20 anos do surgimento do conceito de DT, que foi definido a concepção de dívida técnica de requisitos, por Ernst (2012). Ele definiu a dívida de requisitos como a distância entre a especificação ótima de requisitos e a implementação real do sistema. A dívida técnica também pode se manifestar na fase de requisitos dos sistemas (ERNST, 2012). Os requisitos são a validação final do sucesso do projeto, uma vez que são a representação em sistema dos desejos do usuário. A engenharia de requisitos é fundamental e central para todo projeto de desenvolvimento de software bem-sucedido. Há diversas razões pelas quais os projetos de software podem falhar, contudo requisitos mal elicitados, documentados, validados e gerenciados contribuem consideravelmente para o fracasso de projetos de software (HUSSAIN et al., 2016).

A falha do projeto de software geralmente é cara e arriscada. Representam um desafio no ambiente de mercado competitivo, podendo afetar a imagem da empresa, a geração de receita e diminuir a satisfação percebida de clientes. A engenharia de requisitos é o alicerce sobre o qual se baseia o sucesso dos projetos de software (HUSSAIN et al., 2016).

Inicialmente, o foco da DT estava nas atividades de codificação, mas à medida que a pesquisa se desenvolveu, o conceito foi ampliado para as demais fases de software e desenvolvimento de software, por exemplo, na engenharia de requisitos (MELO et al., 2022). Segundo Ernst (2012) , uma elicitação ou análise inadequada de requisitos causa erros que aumentam a incidência de DT em projetos de software.

De acordo com Van Vliet et al. (2008), etapas e tarefas da engenharia de requisitos quando executadas de forma inadequada podem causar problemas que afetam o desenvolvimento do software, como elicitação de baixa qualidade, requisitos incompletos ou desatualizados juntamente com outros problemas existentes, são exemplos reais de dívida técnica.

Nesse cenário, a dívida técnica de requisitos pode ocorrer de forma intencional, como quando uma decisão consciente é tomada de não ser rigoroso no processo de elicitação , ou não intencionalmente, quando os engenheiros de requisitos são inexperientes e podem não ter as habilidades necessárias para realizar tarefas técnicas e procedimentos de longo prazo (Rios et al., 2019).

A DT de requisitos pode ocorrer também quando é decidido priorizar requisitos que não são necessários e nem entregam maior valor para o cliente (ERNST, 2012). Mesmo após definidos e aprovados, os requisitos podem sofrer alterações durante o desenvolvimento do sistema. Um gerenciamento de mudanças de requisitos de software adequado não só minimiza o aumento de custo, tempo e recursos de projeto de software, mas também tornar o projeto bem-sucedido (BHATTI et al., 2010).

Os requisitos são uma parte importante para o sucesso em projetos de software. Por conseguinte, ter dívida técnica de requisitos, pode contribuir para fracasso destes projetos. Tão importante quanto identificar os itens de DT em um projeto é implementar uma estratégia de gestão eficiente e eficaz para eles. Lenarduzzi e Fucci (2019) propõe mais três tipos de dívida técnica de requisitos:

- **Tipo 0: *Incomplete User´s Needs* (Necessidades incompletas dos usuários):** Representa a dívida contraída ao negligenciar as necessidades de alguns *stakeholders* ou de um grupo específico de *stakeholders*
- **Tipo 1: *Requirements smells* (cheiro de requisito)**, esse tipo se caracteriza por requisitos mal elaborados, com linguagem subjetiva ou termos vagos. *Requirements smells* não levam necessariamente a um defeito, podem ser detectados por meio do documento de especificação. A implementação de um *requirement smells*, remete a risco de juros na dívida técnica.
- **Tipo 2: *Mismatch implementation* (divergência de implementação):** Representa a dívida incorrida quando os desenvolvedores implementam uma solução de requisito incompatível com o que foi especificado. Este tipo de dívida também pode ser incorrido quando o requisito descrito no documento de especificação funcional muda enquanto a implementação não muda em conformidade com a especificação alterada. Nesta situação, os juros podem ser entendidos como a quantidade de mudança entre a implementação atual e o documento de requisitos.

É importante observar que pode haver dívida técnica em todo o ciclo de vida do projeto. Assim, garantindo a qualidade do código-fonte do projeto não é a única maneira de melhorar a qualidade do projeto. No entanto, apesar de haver vários tipos de dívida técnica, grande parte do trabalho se concentra no estudo de problemas existentes no código-fonte. A ênfase em código-fonte pode ser explicada pelo fato de haver métricas e ferramentas de suporte automatizadas para extrair informações referente ao código, que podem ser usadas como indicadores de dívida técnica (ALVES et al., 2016).

Uma das características mais predominantes da dívida técnica é sua natureza interdisciplinar, pois combina elementos da teoria financeira e da engenharia de software. Embora essa natureza possa levar a desafios adicionais, resultantes da lacuna entre o software e as perspectivas financeiras, ela pode potencialmente ajudar na comunicação entre profissionais e pesquisadores (AMPATZOGLOU et al., 2015). A gestão da dívida técnica é um fator decisivo para aumentar o sucesso dos projetos de software (Seaman, & Guo, 2011).

Pagamento de dívida técnica

A literatura atual sobre o tema de dívida técnica carece de um estudo que resuma o estado da arte e a prática em relação às abordagens financeiras utilizadas em dívida técnica. Isso torna difícil para os profissionais procurar por tais abordagens e avaliá-las quanto à adequação ao propósito e aplicabilidade para o problema específico em questão (AMPATZOGLOU et al., 2015).

De acordo com LI et al. (2014), a gestão da dívida técnica pode ser dividida em cinco atividades: identificação, medição, priorização, pagamento e monitoramento. O pagamento é uma das estratégias de gestão de dívida técnica. Pagar seria eliminar o ativo da dívida técnica.

Em um cenário geral de gestão de dívida técnica, o pagamento da dívida envolve a implementação de uma tarefa de manutenção no software. Nesse sentido, selecionar uma abordagem para estimativa de dívida técnica nada mais é do que escolher uma abordagem de estimativa de custos para projetos de manutenção de software (GUO et al., 2016).

Ações de pagamento da dívida técnica fazem parte da gestão. Os principais motivos para o não pagamento de DT são falta de interesse organizacional, baixa prioridade na dívida, foco em metas de curto prazo, custo e falta de tempo (FREIRE et al., 2020). O interessante da dívida técnica é que ela se aposenta automaticamente com o fim da vida útil do software, diferentemente de uma dívida financeira que não acaba até que o pagamento ocorra (BUSCHMANN, 2011). Quando uma equipe de projeto decide pagar uma dívida técnica, é responsabilidade dos arquitetos ajudar a escolher a melhor forma de realizar o pagamento, pois principalmente pode haver também riscos de piorar o cenário, haja vista que a maioria dos sistemas que possuem dívida técnica está em operação produtiva e, portanto, tem valor que não deve ser destruído (BUSCHMANN, 2011).

A dívida técnica também pode ser medida pela caracterização do custo de quitar a dívida. Por exemplo, após determinar os esforços necessários para o pagamento, por

exemplo: redesenho de arquitetura, refatoração de código, dentre outros, a mão de obra, o tempo e o custo de oportunidade podem ser estimados para corrigir os problemas identificados. Esse custo reflete o valor da dívida técnica atualmente no sistema (SEAMAN & GUO 2011).

Os gerentes de software precisam equilibrar os custos e benefícios da dívida técnica e tomar decisões informadas sobre quando e qual dívida técnica deve ser paga. O desenvolvimento ágil parece ser mais propenso ao acúmulo de dívida técnica em comparação com as abordagens tradicionais de desenvolvimento de software, devido ao seu foco orientado à entrega (GUO et al., 2016).

Quando não gerenciada adequadamente, a dívida técnica é considerada como tendo efeitos negativos para o sucesso a longo prazo dos projetos de software. (LENARDUZZI & FUCCI, 2019).

Por outro lado, identificamos que refatoração de código, refatoração de design e atualização de documentação do sistema são as práticas mais utilizadas para pagamento de TD. Embora haja alguma discussão sobre pagamento de DT na literatura técnica, há pouca evidência sobre se os itens de DT foram eliminados de projetos de software, quais práticas de pagamento de DT são atualmente aplicadas por profissionais de software (FREIRE et al., 2020).

Tão importante quanto identificar os itens de DT em um projeto é implementar uma estratégia de gestão eficiente e eficaz para eles. Segundo FREIRE et al. (2020), priorização de DT, pode ser considerada uma prática de pagamento de dívida técnica e está relacionada a práticas que suportam a ordenação de itens de TD de acordo com critérios de classificação. LENARDUZZI e FUCCI (2019), sugerem que uma vez identificada as necessidades dos usuários negligenciadas, ela deve ser formalizada e incluída no documento de especificação de requisitos de software, para que haja o pagamento dessa dívida.

A pesquisa sobre DT é altamente concentrada em alguns tipos de dívida (design, arquitetura, código e defeito), enquanto outros tipos de dívida técnica são pouco investigados, até o momento. Isso mostra uma lacuna clara que pode ser explorado nos próximos anos (ALVES et al., 2016). O tipo de dívida técnica de requisitos foi citado pela primeira vez em 2012 por Ernst. Por ser um tipo de dívida que foi definido recente, por conseguinte há a possibilidade de que ainda não foi totalmente verificado pela comunidade científica sobre estratégias de pagamento de dívida técnica de requisitos.

MÉTODO

Para o desenvolvimento deste estudo, com característica qualitativa, foi realizada revisão da literatura sobre o tema de pagamento de dívida técnica de requisitos. Na etapa de identificação, foi realizada a 1ª bibliometria nas bases *Web of Science* e *Scopus*, sem delimitar o período, ou seja, considerando todos os anos, A sintaxe de busca utilizada foi *("repay" OR "back" OR "payment" OR "pay" OR "return" OR "off" OR "free" OR "out" OR "charge") AND "requirements debt" AND "software"* no título, *abstract* e *keywords.* Como resultado da 1ª bibliometria foi obtido 2 artigos, os quais foram triados na ferramenta excel para retirada de artigos duplicados. Resultou em 1 artigo.

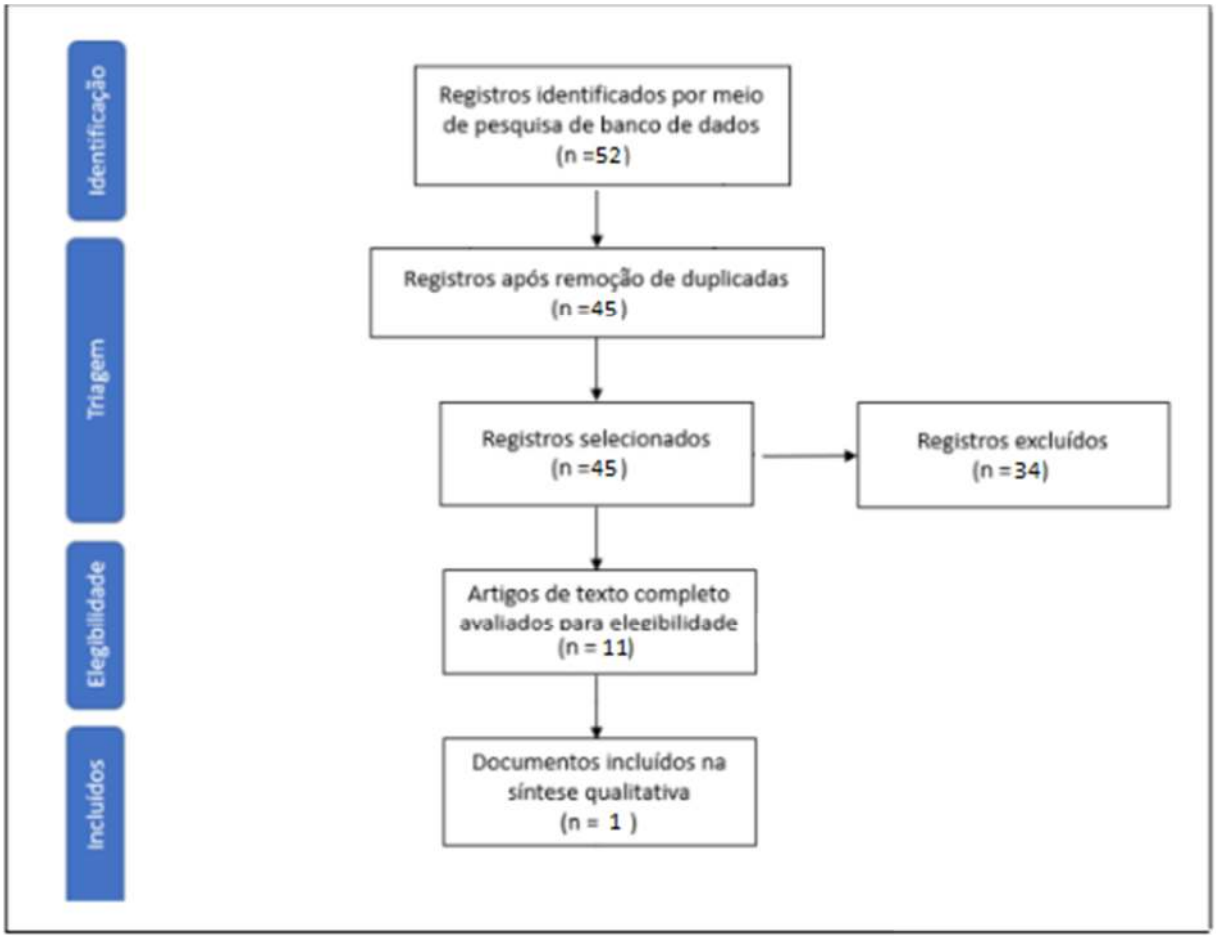
Após analisar o resultado da 1ª bibliometria, que decorreu em 1 artigo, foi constatada a necessidade em expandir a pesquisa a deixando mais ampla sobre o tema em pagamento de dívida técnica, sem restringir por somente dívida técnica de requisitos. Desta forma, foi realizada a 2ª bibliometria para buscar resultados mais abrangentes. Além das bases *Web of Science* e *Scopus*, foi incluída a base Google Acadêmico. Não foi delimitado período de tempo na busca e a sintaxe utilizada foi *("repay" OR "back" OR "payment" OR "pay" OR "return" OR "off" OR "free" OR "out" OR "charge") AND "technical debt" AND "software",* no título, *abstract* e *keywords.* Foram utilizados como parâmetros de exclusão livros, dissertações e teses, filtrando por somente artigos. O resultado da pesquisa foi o indicado abaixo:

- Google acadêmico: 9
- Scopus: 8
- Web of Science: 35

Com base nos artigos encontrados, foi feita a análise de conteúdo com a aplicação baseada no protocolo PRISMA-P, conforme Figura 1. Este protocolo foi desenvolvido como um roteiro para apoiar pesquisadores na realização de revisões que retornem um conjunto mínimo de itens importantes a serem considerados no protocolo de pesquisa (MOHER et al., 2015). Os artigos resultantes da 2ª bibliometria foram tratados na ferramenta Excel, onde houve a remoção de 7 documentos duplicados, restando o total de 45 artigos.

A partir deste resultado, foi aplicado o critério de leitura do título e *abstract* dos 45 artigos para verificar se os mesmos são aderentes ao escopo desta pesquisa. Nesse passo, foram eliminados 34 artigos, resultando em 11. Para complementar este estudo, foi incluído o único artigo encontrado como resultado da 1ª bibliometria deste estudo, totalizando 12 artigos para análise.

Figura 1 – Fluxograma baseado no protocolo PRISMA-P referente à bibliometria.



Fonte: autores (2022)

Os 12 artigos resultantes foram lidos integralmente para análise. No Quadro 1, é ilustrado o título, autores e ano de publicação de cada artigo.

Quadro 1 - Título, autores e ano de publicação dos artigos selecionados para análise

Título do Artigo	Autores	Ano Publicação
The financial aspect of managing technical debt: A systematic literature review	AMPATZOGLOU, Areti et al.	2015
Identification and management of technical debt: A systematic mapping study	ALVES, Nicolli SR et al	2016
Decision criteria for the payment of technical debt in software projects: A systematic mapping study	RIBEIRO, Leilane Ferreira et al	2016
Technical debt reduction using search based automated refactoring	MOHAN, Michael; GREER, Des; MCMULLAN, Paul.	2016
Towards a holistic definition of requirements debt	LENARDUZZI, Valentina; FUC	2019
Surveying software practitioners on technical debt payment practices and reasons for not paying off debt items	FREIRE, Sávio et al.	2020
What are the practices used by software practitioners on technical debt payment: results from an international family of surveys	PÉREZ, Boris et al.	2020
How do technical debt payment practices relate to the effects of the presence of debt items in software projects?	FREIRE, Sávio et al.	2021
Technical debt payment and prevention through the lenses of software architects	PÉREZ, Boris et al.	2021
Does it matter who pays back Technical Debt? An empirical study of self-fixed TD	TAN, Jie; FEITOSA, Daniel; AV	2022
Toward prioritization of self-admitted technical debt: an approach to support decision to payment	DE LIMA, Bruno Santos; GARCIA, Rogerio Eduardo; ELER, Danilo Medeiros.	2022
Do we need to pay technical debt in blockchain software systems?	QU, Yubin et al.	2022

Conforme ilustrado no Gráfico 1, é possível notar q evolução ao longo dos anos quanto as publicações de artigos sobre o tema de pagamento de dívida técnica.

Gráfico 1 - Número de publicações sobre pagamento de dívida técnica ao longo dos anos

RESULTADOS

Com base nas 12 publicações selecionadas, foi realizada a análise do conteúdo sobre pagamento de dívida técnica por meio de leitura integral desses artigos. No trabalho proposto por AMPATZOGLOU et al., no artigo "*The financial aspect of managing technical debt: A systematic literature review"*, os autores trazem a abordagem de que há uma lacuna de comunicação entre as áreas de TI e gerência de projetos no tocante a priorizar o pagamento de dívida técnica. Este artigo teve como objetivo introduzir um vocabulário financeiro sobre dívida técnica. No âmbito da dívida técnica, juros simples custa manutenção contínua do produto e juros compostos aumenta o montante da dívida ao longo do tempo. As abordagens mais populares que os autores indicam para gestão e pagamento de DT são:

- **Análise de Custo-benefício:** analisar a evolução do software os custos relacionados aos juros da dívida técnica em relação ao esforço de desenvolvimento.
- **Gerenciamento de Portfólio:** abordagem iterativa onde a cada interação com o produto pagar alguns itens de dívida técnica do software. Esse método sugere a exclusão da carteira de ativos da dívida técnica que devem ser amortizados, onde a decisão de pagar a dívida e baseada na avaliação de risco.
- **Opções reais:** O valor do ativo como qualidade do serviço. Nessa abordagem, as escolhas de gestão da dívida técnica devem permanecer aberta e livre de qualquer obrigação de pagamento, até que seja inevitável tomar a decisão definitiva de pagar a dívida.

Esse artigo trouxe essas abordagens de gestão com ênfase nos tipos de DT: código, arquitetura e design. Não foi tratado especificamente quanto a dívida técnica de requisitos.

No artigo "*Identification and management of technical debt: A systematic mapping study*", foi feito um mapeamento sistemático onde foram avaliados 100 estudos, datados de 2010 a 2014. O objetivo foi caracterizar os tipos de dívida técnica, identificar indicadores que podem ser utilizados para encontrar dívida técnica e estratégias de gestão. O artigo traz 3 estratégias para gestão de dívida técnica:

- **Análise de Custo-benefício:** Avalia se os juros esperados são altos os suficientes para justificar o pagamento da dívida técnica.
- **Abordagem de Portfólio:** Criar lista de itens de dívida técnica, com descritivo do motivo que é considerada dívida técnica, estimativa do montante principal e dos juros. Durante o planejamento de cada incremento do software, deve ser realizada uma análise do que deve ser pago e/ou que deve ser adiado da dívida técnica
- **AHP (*Analytic Hierarchy Process*):** Processo de hierarquia analítica que pode fornecer método para estrutura problemas, comparando alternativas em relação a critérios especificados. Quando aplicado à dívida técnica, as alternativas de decisão seriam as várias instâncias identificadas de dívida técnica e o resultado seria uma classificação priorizada desses itens.

Dentre essas três estratégias, os autores indicam que as mais utilizadas são a Análise de Custo-benefício e Abordagem de Portfólio. A estratégia de AHP é pouco utilizada como gestão de dívida técnica. O artigo não especifica para qual tipo de dívida técnica essas estratégias são eficientes, trazendo uma visão generalista quanto a aplicação.

No trabalho *"Decision Criteria for the Payment of Technical Debt in Software Projects: A Systematic Mapping Study"*, RIBEIRO et al., os autores trouxeram alguns critérios, como por exemplo, severidade da dívida técnica, visibilidade da dívida, esforço para implementar a correção, dentre outros. Os critérios trazidos com o propósito de ajudar na tomada de decisão para pagamento de dívida técnica. Não houve ênfase e especificidade no tipo de dívida técnica de requisitos.

O artigo "*Technical debt reduction using search based automated refactoring*", traz a abordagem de refatoração de código como estratégia de pagamento de dívida técnica. Refatorar é útil para manter a dívida técnica baixa. Em softwares que possuem dívida técnica, com o passar do tempo, se torna difícil adicionar novas funcionalidades. Os autores ressaltam que pagamentos regulares da dívida técnica podem ser feitos por refatoração do código, porém o tempo gasto na refatoração poderia ser usado para adicionar novas funcionalidades no software. Neste artigo a ênfase foi em dívida técnica de código, não havendo menção sobre dívida técnica de requisitos.

Lenarduzzi e Fucci, apresentam no artigo "*Towards a holistic definition of requirements debt*", uma definição holística de dívida técnica de requisitos que inclui dívida incorrida durante o ciclo de identificação, formalização e implementação de requisitos. Neste artigo, os autores informam que desafiaram a definição clássica (Ernst, Neil 2012) de dívida técnica de requisitos como distância entre o que foi especificado e o que foi implementado, trazendo luz para a abordagem deste tipo de dívida em atividades centradas no usuário, como elicitação de requisitos. Foram classificados três tipos de dívida técnica de requisitos:

- **Tipo 0: *Incomplete User´s Needs* (necessidades incompletas dos usuários):** Representa a dívida contraída ao negligenciar as necessidades de alguns stakeholders ou de um grupo específico de stakeholders.
  - Estratégia de pagamento sugerida: Uma vez identificada a necessidade dos usuários negligenciados, ela deve ser formalizada e incluída no documento de especificação de requisitos de software.
- **Tipo 1: *Requirements smells* (cheiro de requisito)**, esse tipo se caracteriza por requisitos mal elaborados, com linguagem subjetiva ou termos vagos. *Requirements smells* não levam necessariamente a um defeito, podem ser detectados por meio do documento de especificação. A implementação de um *requirement smells*, remete a risco de juros na dívida técnica.
  - Estratégia de pagamento sugerida: Uma vez implementado esse tipo de dívida técnica de requisitos, a refatoração no código precisa ser aplicada para correção.
- **Tipo 2: *Mismatch implementation* (divergência de implementação)*:*** Representa a dívida incorrida quando os desenvolvedores implementam uma solução de requisito incompatível com o que foi especificado. Este tipo de dívida também pode ser incorrido quando o requisito descrito no documento de especificação funcional muda enquanto a implementação não muda em conformidade com a especificação alterada. Nesta situação, os juros podem ser entendidos como a quantidade de mudança entre a implementação atual e o documento de requisitos.
  - Estratégia de pagamento sugerida: As ações de pagamento deste tipo de DT consistem na implementação da melhor nova solução que corresponda à documentação de requisitos atualizada.

É possível observar que as estratégias de pagamento de dívida técnica de requisitos apresentadas não indicam o uso de uma ferramenta sistêmica, mas em boa parte ajustar o documento de especificação funcional. Os autores informam que os três tipos de dívida técnica de requisitos ainda não foram totalmente validados e o próximo passo é validar preliminarmente a conceituação de dívida na engenharia de requisitos.

No artigo "*Surveying software practitioners on technical debt payment practices and reasons for not paying off debt items*", foi realizado um *survey* com 432 profissionais do Chile, Brasil, Colômbia *e EUA* sobre as práticas de pagamento de dívida técnica e os motivos para não pagar. FREIRE et al., trazem as razões para o não pagamento de dívida técnica como: falta de interesse organizacional, baixa prioridade da dívida técnica, foco em metas de curto prazo custos e falta de tempo. Identificaram que refatoração de código, refatoração de design e atualização de documentação de sistema são as práticas mais utilizadas quanto ao pagamento de dívida técnica. A maioria das práticas de pagamento são de natureza técnica, enquanto a maioria dos motivos para não pagar dívida técnica, estão associados a questões não técnicas. O resultado desta pesquisa indica que os itens de dívida técnica não foram pagos na maioria dos casos, parecendo a prática de pagamento ainda não ser uma prática comum nas organizações. Não foi apresentado uma estratégica específica para pagamento de dívida técnica de requisitos, no entanto a questão de melhorar o processo de elicitação de requisitos pode ser visto como uma prática de prevenção de dívida técnica.

No trabalho "*What are the practices used by software practitioners on technical debt payment: results from an international family of surveys*" foi realizado um *survey* parte do projeto *InsigthTD* com 432 participantes, onde foi identificado que as práticas mais utilizadas para pagamento de dívida técnica são refatoração, melhoria nos testes e melhoria no design. Sistemas de grande e pequeno porte, juntamente com sistemas jovens (menos de um ano de implementação), tendem a usar mais refatoração. Os tipos de dívida técnica explorados nesse estudo foram de código, teste e design. Não houve abordagem de prática de pagamento específica para dívida técnica de requisitos.

No artigo "*How do technical debt payment practices relate to the effects of the presence of debt items in software projects?* ", trouxe 10 efeitos da dívida técnica nos projetos, sendo os com mais evidências: baixa qualidade do software, atraso na entrega e retrabalho. Conhecer esses efeitos pode ajudar os times de desenvolvimento de projetos na priorização de quais itens de dívida técnica deve-se priorizar o pagamento. Como prática de pagamento de dívida técnica, a mais utilizada é refatoração de código. O artigo sugere usar a lista de efeitos de dívida técnica para apoio na priorização de pagamento de dívida técnica. Não houve enfoque em dívida de requisitos, no entanto entendo que requisitos mal especificados poderiam causar o efeito "retrabalho", listado nesse estudo

PÉREZ et al., no artigo "*Technical debt payment and prevention through the lenses of software architects*", investigaram as práticas mais utilizadas para quitar e prevenir dívida técnica em projetos de software do ponto de vista do arquiteto do sistema. Foi indicado que decisões arquitetônicas são consideradas uma das fontes de dívida técnica. Quanto a prática de pagamento de dívida técnica a refatoração de código é a prática mais utilizada, seguida de melhoria de design. No quesito de atitudes preventivas para não gerar dívida técnica, ter arquitetura e design bem definidos foi a prática mais citada, seguido de ter escopo e requisitos bem definidos. O artigo não abordou

especificamente pagamento de dívida de requisitos, mas trouxe o ponto de requisitos como atitude de prevenção.

No artigo "*Does it matter who pays back Technical Debt? An empirical study of self-fixed TD",* os autores estudaram 5 tipos de dívida técnica: código, defeito, design, documentação e teste. Esse estudo focou em como os desenvolvedores pagam as dívidas técnicas que os mesmos criaram (dívida técnica autofixada) em projetos de software com linguagem Java e Python. Estudar os fatores humanos no pagamento de dívida técnica também é importante, pois eles podem ter impacto significativo na qualidade do software, por exemplo, um forte sentimento de propriedade sobre um pedaço de código pode aumentar o orgulho de realização e contribuir para que o desenvolvedor escreva um código de melhor qualidade, minimizando gerar dívida de código. Esse estudo não abordou dívida técnica de requisitos.

O trabalho "*Toward prioritization of self-admitted technical debt: an approach to support decision to payment",* traz a abordagem da dívida técnica auto admitida, que é quando intencionalmente esse tipo de dívida é confirmada e documentada por meio de comentários no código fonte. Esse artigo explora o cenário de priorização deste tipo de dívida técnica. A abordagem de priorização de pagamento de dívida técnica auto admitida foca na criação e associação de dívida técnica auto admitida e problemas encontrados no código fonte. O foco desse estudo é na parte técnica, e dívida de código. Não houve tratativas de dívida técnica de requisitos.

QU et al., no estudo *"Do we need to pay technical debt in blockchain software systems? ",* investigou a questão de dívida técnica em 6 projetos de softwares de *blockchain* mais populares no GitHub (plataforma de hospedagem de código-fonte e arquivos com controle de versão). O foco deste artigo foi em identificar dívida técnica de código em sistemas de *blockchain.* Não houve menção sobre dívida técnica de requisitos.

Como análise do resultado é possível observar que as abordagens de pagamento de dívida técnica abordam como a estratégia mais utilizada a refatoração de código, utilizada para o tipo de dívida técnica de código. O único artigo que trouxe abordagem de pagamento de dívida de requisitos foi o "*Towards a holistic definition of requirements debt*", que indica como estratégia principal de pagamento deste tipo de dívida, ter a documentação de requisitos atualizada, com requisitos bem definidos e escritos.

O surgimento do tipo de dívida técnica de requisito foi formalizado na literatura por Ernst (2012). Por ter surgido há 10 anos do período desta pesquisa, sendo relativamente recente, pode ser indicativo de ainda não ter muitas pesquisas sobre o tema. O baixo resultado de estratégias de pagamento de dívida técnica de requisitos, pode indicar uma lacuna de pesquisa nesse âmbito.

CONCLUSÃO

Este artigo teve como propósito identificar se há na literatura estratégias para pagamento de dívida técnica de requisitos. Para essa finalidade, foi realizada a revisão bibliográfica na literatura sem restrição de período, ou seja, considerando todos os anos. O resultado desta pesquisa confirma a proposição P2 em que há oportunidade de

aprofundamento de pesquisa na literatura sobre estratégias para pagamento de dívida técnica de requisitos e refuta a proposição P1 em que não há oportunidade de aprofundamento a literatura sobre esse tema.

Foi identificado somente um artigo específico sobre dívida técnica de requisitos, que propõe algumas estratégias de pagamento. No entanto, a aplicação dessas técnicas, até o momento, não atingiu a uma esfera de validações consideráveis para comprovação de efetividade. Portanto, o resultado indica que há uma possível lacuna de pesquisa a ser explorada por pesquisadores futuros.

REFERÊNCIAS

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

Ampatzoglou, A., Ampatzoglou, A., Chatzigeorgiou, A., & Avgeriou, P. (2015). The financial aspect of managing technical debt: A systematic literature review. *Information and Software Technology*, *64*, 52-73.

Bhatti, M. W., Hayat, F., Ehsan, N., Ishaque, A., Ahmed, S., & Sarwar, S. Z. (2010, October). An investigation of changing requirements with respect to development phases of a software project. In *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)* (pp. 323-327). IEEE.

Buschmann, F. (2011). To pay or not to pay technical debt. *IEEE software*, *28*(6), 29-31.

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

de Lima, B. S., Garcia, R. E., & Eler, D. M. (2022). Toward prioritization of self-admitted technical debt: an approach to support decision to payment. *Software Quality Journal*, 1-27.

Ernst, N. A. (2012, June). On the role of requirements in understanding and managing technical debt. In *2012 Third International Workshop on Managing Technical Debt (MTD)* (pp. 61-64). IEEE.

Fowler, M. (2009). Technical debt quadrant, 2009. *URL: http://martinfowler. com/bliki/TechnicalDebtQuadrant.html*.

Freire, S., Rios, N., Gutierrez, B., Torres, D., Mendonça, M., Izurieta, C., ... & Spínola, R. O. (2020). Surveying software practitioners on technical debt payment practices and reasons for not paying off debt items. In *Proceedings of the Evaluation and Assessment in Software Engineering* (pp. 210-219).

Freire, S., Rios, N., Pérez, B., Torres, D., Mendonça, M., Izurieta, C., ... & Spínola, R. (2021, March). How do technical debt payment practices relate to the effects of the presence of debt items in software projects? In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 605-609). IEEE.

Guo, Y., Spínola, R. O., & Seaman, C. (2016). Exploring the costs of technical debt management–a case study. *Empirical Software Engineering*, *21*(1), 159-182.

Hussain, A., & Mkpojiogu, E. O. (2016, August). Requirements: Towards an understanding on why software projects fail. In *AIP Conference Proceedings* (Vol. 1761, No. 1, p. 020046). AIP Publishing LLC.

Hussain, A., Mkpojiogu, E. O., & Kamal, F. M. (2016). The role of requirements in the success or failure of software projects. *International Review of Management and Marketing*, *6*(7), 306-311.

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

Lenarduzzi, V., & Fucci, D. (2019, September). Towards a holistic definition of requirements debt. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (pp. 1-5). IEEE.

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, *101*, 193-220.

Li, Z., Liang, P., & Avgeriou, P. (2014). Architectural debt management in value-oriented architecting. In *Economics-Driven Software Architecture* (pp. 183-204). Morgan Kaufmann.

McConnell, S. (2008). Managing technical debt. *Construx Software Builders, Inc*, 1-14.

McManus12, J., & Wood-Harper, T. (2007). Understanding the sources of information systems project failure.

Melo, A., Fagundes, R., Lenarduzzi, V., & Santos, W. B. (2022). Identification and measurement of Requirements Technical Debt in software development: A systematic literature review. *Journal of Systems and Software*, 111483.

Mohan, M., Greer, D., & McMullan, P. (2016). Technical debt reduction using search based automated refactoring. *Journal of Systems and Software*, *120*, 183-194.

Pérez, B., Castellanos, C., Correal, D., Rios, N., Freire, S., Spínola, R., & Seaman, C. (2020, June). What are the practices used by software practitioners on technical debt payment: results from an international family of surveys? In *Proceedings of the 3rd International Conference on Technical Debt* (pp. 103-112).

Pérez, B., Castellanos, C., Correal, D., Rios, N., Freire, S., Spínola, R., ... & Izurieta, C. (2021). Technical debt payment and prevention through the lenses of software architects. *Information and Software Technology*, *140*, 106692.

Qu, Y., Bao, T., Chen, X., Li, L., Dou, X., Yuan, M., & Wang, H. (2022). Do we need to pay technical debt in blockchain software systems? *Connection Science*, *34*(1), 2026-2047.

Ribeiro, L. F., de Freitas Farias, M. A., Mendonça, M. G., & Spínola, R. O. (2016). Decision Criteria for the Payment of Technical Debt in Software Projects: A Systematic Mapping Study. *ICEIS (1)*, 572-579.

Rios, N., Spínola, R. O., Mendonça, M., & Seaman, C. (2019, May). Supporting analysis of technical debt causes and effects with cross-company probabilistic cause-effect diagrams. In *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)* (pp. 3-12). IEEE.

Seaman, C., & Guo, Y. (2011). Measuring and monitoring technical debt. In *Advances in Computers* (Vol. 82, pp. 25-46). Elsevier.

Tan, J., Feitosa, D., & Avgeriou, P. (2022). Does it matter who pays back Technical Debt? An empirical study of self-fixed TD. *Information and Software Technology*, *143*, 106738.

Van Vliet, H., Van Vliet, H., & Van Vliet, J. C. (2008). *Software engineering: principles and practice* (Vol. 13). Hoboken, NJ: John Wiley & Sons.