DOI: 10.36558/rsc.v12i1.7569

Análise Quantitativa da Aquisição de Dívida Técnica em Sistemas Legados

Gabriela P. Serafim
Universidade do Oeste de Santa Catarina
Unoesc
Chapecó/SC
gabriela.pianezzola@unoesc.edu.br

Ana M. Debiasi Duarte
Universidade do Oeste de
Santa Catarina - Unoesc
Chapecó/SC
ana.duarte@unoesc.edu.br

Denio Duarte
Universidade Federal da
Fronteira Sul - UFFS
Campus Chapecó - SC
duarte@uffs.edu.br

ABSTRACT

Software development companies that work with legacy systems face challenges typical of these systems, *e.g.*, maintenance carried out for decades, outdated concepts, and a system heavily coupled with the hardware. Conversely, technical debt describes the development of low-quality systems to be delivered quickly but with the view that the system will be improved in the future. Such situations occur when the delivery of a module has to be fast to meet market demand. Thus, legacy systems and technical debt are firmly linked, as the maintenance that occurs in the former are, in most cases, technical adjustments to modernize the system, both in terms of requirements and technology. The management of technical debts and legacy systems is a challenge. Therefore, this work aims to identify the factors of technical debt acquisition in legacy software in the Western Region of Santa Catarina, according to classification patterns already identified in the literature. The research followed an applied approach, from identifying characteristics of companies in the area and the relationship with their programming practices and prioritization of corrections. The results confirmed the presence of factors already described in the literature, such as process, attitude, pragmatism, and prioritization, present in the development of the researched products. Nevertheless, there was no significant relationship between the characteristics of the companies and the acquisition of technical debts.

CCS Concepts

· **Information systems** → *Process control systems; Process control systems;*

Keywords

technical debt; legacy systems; software maintenance; quantitative analysis

1. INTRODUÇÃO

A manutenção representa de 85 a 90% do custo total de um software legado, considerando seu ciclo de vida [16]. O termo manutenção, utilizado neste trabalho, compreende as manutenções de caráter corretivo e evolutivo. As manutenções corretivas são necessárias para corrigir defeitos inseridos durante o processo de desenvolvimento. As manutenções evolutivas são aquelas necessárias para manter o software (aplicação) atualizado para atender novas necessidades de um grupo de usuários. Dentre os dois tipos de manutenções, a que tem custo mais alto é a corretiva, pois erros inseridos durante o ciclo de vida do software tendem a ser muito maior do que durante a fase de desenvolvimento [7].

Sistemas legados são sistemas que implementam funções críticas para o negócio ou são importantes para a organização. Tendem a ter uma longa vida útil e, geralmente, são muito caros ou cr´ıticos para serem substitu´ıdos [36]. Geralmente, são desenvolvidos com tecnologias que não estão mais em uso, tanto *hardware* (*e.g.*, monitores sem capacidade gráfica) quanto software (*e.g.*, linguagens de programação como Cobol). Assim, o custo de manutenção, tanto financeiro quanto de recursos humanos, é alto. Segundo uma pesquisa da Accenture [4], 72% dos grandes executivos concordam que sistemas legados prejudicam a capacidade de migração para novas tecnologias. De acordo com pesquisa do *MIT Sloan School of Management*, 67% dos executivos gostariam de substituir seus principais sistemas legados [19].

Apesar de existirem várias definições de sistemas legados, a definição de quando um sistema se torna legado não é trivial. Os autores em [9], afirmam não terem encontrado estudos em que definições de sistemas legados tenham sido organizadas, correlacionadas e sintetizadas de forma sistemática. Assim, neste trabalho, o tempo de vida do sistema foi utilizado como parâmetro para caracterizar um sistema como legado. Baseado nas discussões em [35], sistemas com mais de cinco anos, independente de tecnologia utilizada, foram considerados como legado.

Em algumas situações, o balanceamento entre a implementação de um código com a menor quantidade de erros possível e o prazo de entrega, faz com que desenvolvedores e usuários finais optem pelo prazo de entrega. Esse cenário foi descrito por [11], que afirma que um código imaturo deve ser reescrito futuramente gerando dívida técnica (DT). Dívida técnica é similar a uma dívida tradicional: permite-se um desenvolvimento rápido cujo custo será pago mais tarde. Um estudo similar apresentado em *37+ mostrou que a d´ıvida técnica, nos EUA, foi estimada em U$ 511 bilhões.

De acordo com [18], a dívida técnica pode ser pensada como uma dívida financeira: o esforço para reescrita do có-

digo equivale aos juros pagos sobre a d´ıvida. Segundo [3], a dívida técnica pode ser causada por itens como um processo, uma decisão, uma determinada ação ou falta dela, ou por um evento que desencadeia a dívida, como pela pressão para cumprimento de cronograma, indisponibilidade de uma pessoa-chave ou, ainda, pela falta de informações sobre um recurso técnico.

Diante dos cenários de sistemas legados e dívida técnica, este trabalho propõe uma análise quantitativa da relação entre ambos em empresas de desenvolvimento de software da região Oeste de Santa Catarina. O objetivo é identificar se o tempo de existência do código-fonte, a idade da empresa ou o tamanho da empresa (quantidade de funcionários) são elementos que influenciam na geração da dívida técnica. Também, pretende-se identificar se as empresas costumam pagar as dívidas técnicas contraídas durante o desenvolvimento. A pesquisa pretende identificar características de análise estática de código, violações de modularidade e *code smells* nos códigos-fonte que compõem os sistemas legados.

Profissionais que trabalham nas empresas alvo da pesquisa responderam a um questionário com perguntas voltadas para caracterizar a empresa, o software desenvolvido e entender o processo de desenvolvimento. Após a aplicação, os dados foram tabulados e analisados quantitativamente. Os resultados indicam que empresas com mais idade e mais funcionários possuem mais dívida técnica do que as menores ou mais jovens, e que sistemas legados são mais propensos a terem dívida técnica, conforme comparação entre os dois tipos de sistemas considerados: legados e mais recentes.

O restante do trabalho está organizado da seguinte forma: a próxima seção apresenta o referencial teórico deste trabalho. A Seção 3 apresenta a metodologia da condução da pesquisa e a Seção 4 os resultados da proposta. Na Seção 5 são apresentados alguns trabalhos relacionados ao tema proposto neste artigo. Finalmente, a Seção 6 conclui este trabalho bem como apresenta algumas direções futuras.

2. REFERENCIAL TEÓRICO

Durante o ciclo de vida do software, as necessidades dos usuários sofrem modificações; consequentemente os sistemas devem se adaptar para permanecerem em uso [31]. Essas necessidades de manutenções acarretam custos, então é preciso atentar aos critérios de qualidade no código-fonte para o custo do desenvolvimento não ser maior que o esperado. O custo da manutenção não está relacionado apenas aos aspectos financeiros diretos. Ele pode, por exemplo, gerar situações em que um código mal descrito leva o desenvolvedor a demorar mais para resolver um problema em um código [26].

O custo da manutenção relatado, também pode ser chamado dívida técnica. Essa metáfora é definida por uma abordagem de projeto ou construção conveniente a curto prazo. No entanto, em um contexto técnico, o mesmo trabalho custará mais para ser executado tardiamente do que custaria para ser executado no momento [2].

Embora a metáfora da dívida técnica tenha sido proposta há duas décadas, apenas nos últimos anos vem recebendo atenção significativa [23]. Adquirir dívida técnica é inevitável; por isso, o objetivo principal não é eliminá-la, mas mantê-la sob controle e gerenciá-la [21].

2.1 Manutenção em softwares legados

Há cerca de 30 anos, a capacidade de armazenamento de software disponível era pequena, o que levava a restrições no projeto de software. Os programadores precisavam economizar espaço utilizando *aliasing* e estruturas grandes de dados globais; a eficiência era importante, pois a clareza e a estrutura eram trocadas pela velocidade. O resultado é sistemas legados dif´ıceis de entender [5].

Sistemas legados são compostos por softwares antigos utilizados pelas empresas. Estes sistemas agregam valor ao negócio por cinco ou mais anos, além de resistir a alterações evolutivas e corretivas [6], tendem a ter tecnologia ou métodos obsoletos, e, mesmo com as dificuldades de manutenção, são essenciais para os negócios e precisam ser mantidos [36].

Sistemas legados trazem impasses no cotidiano das empresas por frequentemente possuir projetos não expansíveis, código intrincado, documentação pobre ou inexistente, casos de teste que nunca foram salvos e histórico de manutenção mal administrado. Estes fatores são considerados maus hábitos de desenvolvimento de software. Mesmo com esses maus hábitos, o software precisa evoluir para se adaptar e atender novas necessidades, ou ser rearquitetado para ser mantido com tecnologias viáveis [31].

Por isso, no cenário de desenvolvimento de sistemas legados, surge um dilema. Prosseguir com a manutenção do sistema se torna caro e ineficaz para as alterações necessárias, e leva a perdas de oportunidades de negócios. Porém, o sistema é muito valioso e substituí-lo traz novos custos de desenvolvimento e implantação, além do risco da perda de informações críticas dos negócios. Tal perda não é aceitável para as organizações [5, 41].

2.2 Dívida Técnica

Dívida técnica (DT) retrata cenários em que, para acelerar o desenvolvimento de software, é necessário o desenvolvimento de um código imaturo a ser reescrito futuramente, gerando uma dívida [12]. Mais recentemente, a dívida técnica foi definida por [2] como uma abordagem de projeto ou construção, conveniente a curto prazo, mas que, em um contexto técnico, custe mais para ser executado tardiamente do que custaria no momento, incluindo aumento de custo ao longo do tempo. A metáfora facilitou a discussão entre as partes técnicas e não técnicas em meio ao processo de desenvolvimento de software, pois forneceu uma estrutura e vocabulário conhecidos do domínio financeiro [38].

A dívida técnica pode ser intencional e não-intencional *22+. A d´ıvida intencional ocorre quando pessoas envolvidas no projeto decidem inserir a d´ıvida, ou seja, sabem que aquela não é a melhor técnica, mas tomam a decisão pensando no presente e não no futuro. A dívida não intencional é adquirida de forma involuntária, sem planos estratégicos. Pode ser causada por uma equipe mal preparada ou até mesmo por um processo mal descrito. Por não ser adquirida propositalmente, essa categoria de d´ıvida pode ser a mais danosa para o ciclo de vida do software, pois não é visível para o gerenciamento e pode se tornar fora de controle [2].

Em [17], é encontrada uma classificação semelhante: dívida intencional ou imprudente. A d´ıvida intencional ocorre quando a equipe possui ciência e assume a dívida. Nesse caso, a equipe realiza uma entrega com problemas técnicos, mas já se planeja para realizar o pagamento. Esse tipo de d´ıvida ocorre principalmente pela falta de tempo ou por pressão para entregar algum desenvolvimento com rapidez. É considerada uma dívida perigosa, pois terá grandes consequências e resultará em juros técnicos caso seja esquecida.

Dívida imprudente (ou não-intencional), segundo [22], ocorre sem intenção, ou seja, a equipe desconhece ou ignora as boas práticas de projeto de *software*. Quando equipes possuem profissionais que aprendem boas práticas de programação durante o desenvolvimento do sistema, mesmo que acreditem estar fazendo um bom trabalho, após adquirirem mais conhecimento notam que poderiam ter feito melhor trabalho [17].

O principal perigo ocorre quando a dívida técnica é inserida no software e não é paga, pois, a cada minuto em que o código é mantido em inconformidade, juros são acrescidos. Isso torna a d´ıvida cada vez mais dif´ıcil de ser paga futuramente [22].

2.3 Métricas de código-fonte

Métricas de código-fonte podem auxiliar no desenvolvimento de um código claro, simples e flexível. Assim, o monitoramento do código pode ser uma forma de identificar implementações com problemas [27]. A seguir, são apresentadas as descrições das métricas de código-fonte utilizadas para elaboração do questionário aplicado na pesquisa.

Primeiramente, são apresentadas as métricas de tamanho LOC, AMLOC e Total Number of Modules or Classes.

- LOC (*Lines of Code*) ou SLOC (*Source lines of code*): trata-se de quantidade de linhas executáveis do código, por isso é uma métrica mais perceptível de custo do software [30].
- AMLOC (*Average Method LOC*): essa métrica preza pela boa distribuição dos métodos no código fonte [27]. A métrica pretende identificar como os métodos são distribu´ıdos de acordo com o tamanho em linhas de código.
- *Total Number of Modules or Classes* (TNMoC): total de módulos ou classes, essa métrica é menos influenciada por linguagens de programação, nível dos desenvolvedores e estilos de codificação [39].

Além das métricas de tamanho também é possível identificar má qualidade de código fonte em métricas estruturais:

- NOM (*Number of Methods*): trata-se da medição dos números dos métodos. Essa métrica permite identificar o potencial de reuso de uma classe, pois classes com um grande quantidade de métodos são mais difíceis de serem reutilizadas [25].
- ACCM (*Average Cyclomatic Complexity per Method*): o termo complexidade ciclomática se refere a um grande número de componentes que interagem em um sistema [29]. Essa métrica fornece dados quantitativos de complexidade lógica de um programa [31].

O acoplamento é uma característica do software usada como métrica de qualidade de código. Mede o grau de ligação entre as classes do software. Quanto mais acoplamento maior a dificuldade para alterar uma classe do sistema, pois a alteração de uma classe pode impactar todas as classes acopladas [27]. A seguir, são apresentas três métricas associadas ao acoplamento [10]:

- ACC (*Afferent Connections per Class*): essa métrica mede a conectividade de uma classe a partir da contagem do número de classes do sistema que acessam um atributo ou método da classe em análise. Caso o valor desta métrica seja alto, é possível identificar que as modificações na estrutura das classes podem afetar as demais classes acopladas.
- CBO (*Coupling Between Objects*): está relacionada a métrica de ACC. Mede quantas classes são utilizadas pela classe analisada.
- COF (*Coupling Factor*): é responsável por medir a interconexão do software. Um software com alto grau de conexão possui maior COF, isso ocasiona alta complexidade e difíceis entendimento e manutenção.

A seguir são apresentas duas métricas comumente utilizadas para análise de coesão. A coesão mede a diversidade de assuntos (diferentes regras) que uma classe implementa. Boas abstrações de classes geralmente exibem alta coesão [8, 31].

- LCOM4 (*Lack of Cohesion in Methods*): calcula a quantidade de componentes conectados em uma classe. Um componente é considerado conectado quando um conjunto de métodos estão relacionados. Há relação entre dois métodos quando ambos acessam as mesmas variáveis da classe ou um método invoca outro. [31].
- SC (*Structural Complexity*): quanto maior a complexidade do software, mais difíceis são as alterações e evoluções. Tanto o acoplamento quanto a coesão não estão relacionados ao esforço de manutenção de software. Apesar disso, são considerados em conjunto, pois devem estar em equil´ıbrio na arquitetura do software. Quando usados como uma métrica, o produto de acoplamento e coesão pode ser correlacionado ao esforço da manutenção [31].

No presente trabalho foram utilizadas as métricas LOC, AMLOC, TNMoC, NOM, ACCM, ACC, CBO, COF, LCOM4 e SC como base para a criação do questionário. Foi adotada a estratégia de criar um conjunto de perguntas para atender cada métrica abordada. Para facilitar a identificação do uso de métricas pelos profissionais, as perguntas elaboradas não citaram as métricas diretamente e sim as práticas adotadas que compreendem cada uma das métricas. A Tabela 1 apresenta as questões e as métricas que inspiraram suas criações. Por motivo de clareza foram colocados apenas os números das questões (o Apêndice A apresenta detalhes do questionário aplicado).

Tabela 1: Lista de perguntas associadas às métricas apresentadas.

Questão	Métricas
Q15	LCOM4 e SC
Q16	NOM e ACCM
Q17	NOM e ACCM
Q20	ACC, CBO, COF, LOCM4 e SC
Q26	Todas
Q27	AMLOC e TNMoC
Q28	SC
Q29	NOM, ACCM e SC
Q30	SC

3. COLETA DE DADOS

Esta seção apresenta a forma com que foi realizada a pesquisa, apresentando a população e amostra em que foi conduzida a coleta de dados.

A população utilizada para amostragem neste trabalho é de desenvolvedores, arquitetos e gestores das empresas de software da região do Oeste de Santa Catarina. A região contabiliza um total 1.192 empresas de software [1]. O questionário encaminhado teve como foco colaboradores que atuam nestas empresas.

Para avaliar o tamanho da amostra a ser pesquisada foi utilizado o cálculo de confiabilidade proposto por [32]. O cálculo utilizou o nível de confiança (s^2), com um valor de 90%, uma margem de erro amostral (E^2) de 5% e, para o tamanho da população (N), um número equivalente a todas as empresas de software da região Oeste de Santa Catarina, ou seja, 1.192. A Equação 1 apresenta a fórmula para calcular o tamanho da amostra:

$$TA = \frac{\frac{z^2 \times (p \times (p-1))}{E^2}}{1 + \frac{z^2 \times (p \times (p-1))}{E^2 \times N}} \quad (1)$$

onde z é 1,65 (*i.e.*, *z-score* de 90% de confiança) e p é o desvio padrão que foi considerado como 0,5 (*i.e.*, 50%). Optou-se por um respondente por empresa e, assim, aplicando a Equação 1, o tamanho da amostra deve ser de 221,63, ou seja, 222.

A coleta de dados foi realizada entre 4 de setembro de 2020 e 3 de março de 2021. Foram obtidas 75 respostas dos 222 questionários de diferentes profissionais, o que reduziu a margem de erro amostral e o nível de confiança. Sendo ajustados, então, para a margem de erro de 7% com um nível de confiança de 80%. Um respondente foi selecionado de cada empresa, além do respondente avaliar apenas um dos produtos da empresa, caso a empresa tivesse mais de um em produção.

O questionário é composto por 34 questões e é caracterizado como semiaberto. Algumas perguntas têm alternativas – SIM e NÃO (seis questões), outras permitem que o respondente descreva sua opinião (cinco questões) e outras utilizam como base de resposta a escala de frequência de Likert [24] (16 questões). As sete primeiras questões são para caracterizar a empresa. O Apêndice A apresenta o questionário aplicado.

4. RESULTADOS

Para análise dos dados, inicialmente buscou-se identificar se as empresas, nas quais os respondentes atuam em projetos, possuem ou não dívida técnica. Essa análise aconteceu por meio das seguintes perguntas do questionário:

- Encontro problemas nos projetos de arquitetura que devem ser resolvidos antes de iniciar o desenvolvimento do software (Q13).
- Ao desenvolver me deparo com acoplamentos desnecessários, falta de abstração e duplicação de código no projeto de código fonte (Q20).
- Durante o meu cotidiano entregas rápidas são cobradas, o que prejudica a qualidade do código (Q25).
- Na empresa que trabalho é um habito aplicar padrões de projetos nos códigos (Q30).
- Na empresa em que trabalho costuma-se realizar refatoração de código (Q31).

Os seguintes fatores foram utilizados para identificar um respondente em uma empresa que possui projetos com d´ıvida técnica: codificação e padrões de arquitetura, padrões de projeto e refatoração de código. Esses fatores foram capturados pelas perguntas apresentadas anteriormente. As perguntas foram consideradas positivas (Q30 e Q31) ou negativas (Q13, Q20 e Q25). Conforme a polaridade da pergunta (*i.e.*, positiva ou negativa), as respostas recebem um peso conforme apresenta a Tabela 2. As empresas que contabilizarem no máximo 15 pontos são classificadas como possuintes de projetos com dívidas técnicas. Ou seja, nas cinco perguntas dos fatores considerados, a empresa recebeu 5 em todas. Das 75 respostas obtidas (*i.e.*,75 empresas), 45 configuraram como empresas que possuem projetos com d´ıvida técnica. A caracterização dos projetos como possuidores de dívida técnica e as perguntas selecionadas são apresentadas a seguir.

Tabela 2: Pontuação de acordo com o tipo de concordância

Grau de concordância	Positivo	Negativo
Muita frequência	5	1
Frequentemente	4	2
Ocasionalmente	3	3
Raramente	2	4
Nunca	1	5

Códigos que estão fora dos padrões de *design* de arquitetura já durante a sua criação são chamados de *grimes*. Porém, além do código estar em inconformidade no momento da sua criação, também é possível que haja inconformidade de *design* durante a sua alteração. Esta inconformidade é chamada de *rot*. Ambos os casos são considerados dívidas técnicas porque prejudicam a qualidade do código [42].

Padrões de *design* prometem que o código seja mais sustentável e menos propenso a defeitos [14]. Desse modo, se a empresa possui hábito de aplicar padrões de projetos, o código fica menos propenso a defeitos e com manutenibilidade mais simples, o que evita a aquisição dívida técnica.

Assim, para avaliar se um projeto possui ou não dívida técnica, o fator de aplicação de padrões de projeto foi considerado positivo. Ao aplicar esses padrões, a empresa se preocupa com a não inserção da dívida.

A Seção 2.3 apresentou algumas métricas que são consideradas boas práticas de programação. Presume-se que o não uso dessas métricas pode ser considerado um fator negativo, pois impacta na qualidade de código. Também levou-se em consideração se a empresa faz inserção da dívida conforme a classificação apresentada na Seção 2.2. Assim, caso a empresa cobre entregas rápidas de forma a prejudicar a qualidade do código, ela faz inserção de dívida técnica de forma intencional ou não intencional.

Quarenta e cinco dentre os 75 respondentes foram caracterizados por possuírem projetos com dívida técnica. Assim, neste trabalho, os 30 respondentes sem dívida técnica não serao analisados. Entre os 45 respondentes, que foram caracterizados por fazerem parte de projetos com dívida técnica, buscou-se identificar alguma relação entre dívida técnica e demais caracter´ısticas da empresa, como tempo de software ou aplicação, idade da empresa ou quantidade de funcionários.

A pergunta Q5 é relativa à idade do software e a Figura 1 apresenta o resultado. Percebe-se que aplicações de mais de 7 anos correspondem a 48,89% (12 de 7 a 10 anos e 10 de mais de 10 anos). Somado com as aplicações de 5 a 7 anos, esse percentual atinge quase 69% (9 empresas). É uma quantidade alta, pois existem apenas duas faixas das cinco possíveis de resposta. Também, percebe-se que nove empresas (20%) possuem projetos com dívidas técnicas mesmo com aplicações mais novas (0 a 3 anos).

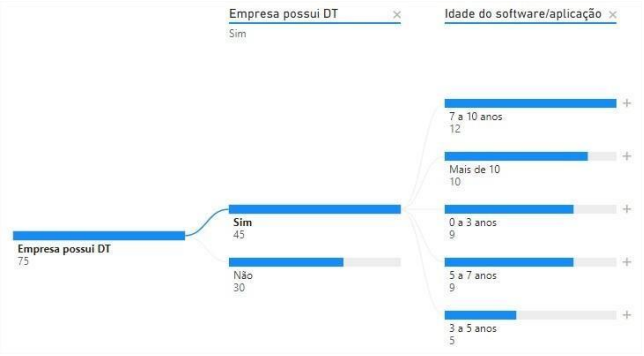

Figura 1: Relação entre dívida técnica e idade do software.

As dívidas técnicas dos sistemas legados podem ter sido adquiridas há muito tempo e o seu acúmulo faça que, talvez, elas nunca sejam pagas [28]. O percentual encontrado nesse levantamento (69%) aponta para essa tendência.

A análise seguiu tentando identificar a idade da empresa e a aquisição da dívida técnica (Q4). Os resultados apontam que empresas mais jovens (0 a 5 anos) tendem a ter menos dívida técnica. Em média, 40% dessas empresas possuem projetos com d´ıvida (8 empresas no total de 20). Em empresas mais velhas, o percentual chega a 62% (37 de 55 empresas). A Figura 2 apresenta as quantidades por faixa de idade. Essa tendência é esperada, pois, empresas mais antigas tendem a contrair mais d´ıvida durante o ciclo de vida de seus produtos.

Figura 2: Relação entre dívida técnica e idade das empresas.

Em relação à quantidade de funcionários (Q6), foi percebido que a maioria das empresas nas faixas de quantidade de funcionários proposta no questionário tiveram resultados parecidos. Por exemplo, dos 44 respondentes que se encaixam na faixa *Até 50 funcionários*, 24 possuem projeto com dívida técnica (ou seja, 54,54%). O mesmo padrão segue para as outras faixas sendo que aquelas com *Mais de 1000 funcionários* o percentual chega a 72,73% (*i.e.*, de 11 com mais de 1000 funcionários, 8 possuem projeto com dívida técnica). Isso pode ser causado por diversos fatores, como, por exemplo, o investimento insuficiente da empresa no desenvolvimento de seus colaboradores, ou até mesmo a falta de um gerenciamento mais presente para avaliar os códigos fontes gerados por seus desenvolvedores. A Figura 3 apresenta a quantidade de empresas com dívida técnica por faixa de número de funcionários definida.

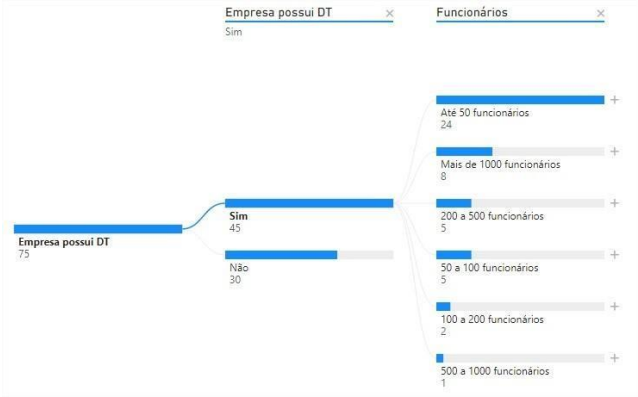

Figura 3: Relação entre dívida técnica e quantidade de funcionários.

As próximas seções apresentam a análise sobre a aquisição da dívida técnica e as percepções dos respondentes, e, em seguida, a preocupação com o pagamento da dívida técnica, destacando os sistemas legados, no sentido de identificar diferenças de comportamento quando existem sistemas que estão em produção há 5 anos ou mais.

4.1 Ponto de vista dos participantes sobre a aquisição da dívida técnica

Conforme apresentado anteriormente, o tempo da empresa e a idade da aplicação estão relacionados com a aquisição da dívida técnica. Porém, analisando a Figura 1, percebe-se que aplicações *jovens* (de 0 a 5 anos) são responsáveis por 31% das empresas com dívidas técnicas. Apesar dos sistemas serem recentes no mercado, há alguns fatores que podem contribuir para o acumulo da d´ıvida, conforme definições de [40]:

- Pragmatismo: código mal descrito e inúmeras inconsistências, mesmo que sejam pouco relevantes. O pragmatismo leva ao adiamento ou implementação abaixo do ideal em tarefas maiores. Em [2], o autor traz como um exemplo pequenas empresas em um nicho de negócio, nesses casos, lançar por primeiro o produto no mercado é a prioridade e problemas a longo prazo não importam naquele momento.
- Priorização: entregas são priorizadas ao invés da boa qualidade do código [40].
- Processos: quando não há um processo bem definido nem uma boa comunicação entre os colaboradores, então torna-se mais fácil acumular dívida técnica [40].
- Atitudes: trata-se da hesitação do programador e da gerência em melhorar o código, por medo de introdu-

zir novos problemas: ”se não está quebrado, não conserte” *15+. Isso se caracteriza como um dos motivos mais comuns para a não-identificação da dívida técnica.

Para identificar se os desenvolvedores não tem conhecimento sobre as boas práticas de programação foram analisadas duas perguntas:

- Busco aplicar boas práticas de programação durante meu dia a dia (Q26).
- Você já codificou um método muito longo por não saber como abstraí-lo em métodos menores? (Q27)

Dos respondentes que trabalham em empresas que possuem projeto com dívida técnica, 28 já codificaram um método muito longo por não saber como abstraí-lo em métodos menores. Isso interfere na métrica de tamanho AMLOC (veja Seção 2.3) e pode ser um fator que influencia a dívida técnica. Destes 28 respondentes, todos responderam que ocasionalmente, frequentemente ou com muita frequência buscam aplicar boas práticas de programação.

Fatores que podem ser analisados nas respostas obtidas na pesquisa são os fatores de *atitude* e *processo* apontados por [40]. Atitude, pois o método em que o desenvolvedor está trabalhando pode já estar muito longo e ele não planeja fazer essa correção, apesar de tentar aplicar boas práticas de programação em seu cotidiano. Processo, pois caso houvesse um processo estabelecido e específico para refatoração, aos poucos, durante as manutenções, o problema no método estaria sendo corrigido.

Para analisar se o respondente não possui conhecimento no projeto todo, foi analisada a Q28 (*Já precisei criar duplicações no código por não ter conhecimento de todo o projeto?*). Entre os respondentes, 31 responderam ocasionalmente, frequentemente e com muita frequência, o que demonstra que já precisaram criar duplicações no código por não ter conhecimento de todo o projeto. Desses 31 respondentes, 23 atuam em empresas que possuem sistema legado e 8 em empresas com sistema com menos de cinco anos de idade. Esses fatores podem estar relacionados a um problema de processo conforme proposto por [40]. Além disso, a falta de conhecimento do desenvolvedor em relação ao código que altera pode ser ocasionada pela falta de documentação das classes e métodos.

A Q19 (*Você acha que o fato de a dívida técnica não ser facilmente identificada e comunicada é uma causa do seu acúmulo?*) tem o intuito de identificar se é dada baixa visibilidade da dívida técnica adquirida. Para esta pergunta, não houve muitos respondentes que acham que a dificuldade de identificação e comunicação da dívida técnica é uma causa do seu acúmulo. Desse modo entende-se que isso não é um fator relevante para sua aquisição na região.

Dos 45 respondentes, 19 responderam que a empresa nunca ou raramente realiza a refatoração de código (Q17). Dentre eles, 13 atuam em empresas de sistema legado e seis trabalham em empresas com sistema de menos de cinco anos de idade. Percebe-se que empresas de sistema legado que possuem dívida técnica não possuem hábito de refatorar o código, fator que pode impactar diretamente em novos desenvolvimentos e aumentar o valor do software em manutenção, conforme apresentado na Seção 2.1. Grande parte do esforço da fase de manutenção do software se dá na manutenção perfectiva e adaptativa. Assim, enquanto não for realizada a refatoração do código, haverá juros sobre a dívida existente, isso pode levar à problemas no longo prazo.

Desse modo, é possível identificar também um problema de processo, por não haver um aumento da visibilidade da dívida técnica inserida. Em conjunto com o fator de pragmatismo, que pode impactar em uma implementação abaixo do ideal de tarefas maiores.

Por fim, foram analisadas as respostas obtidas para a Q25 (*Durante o meu cotidiano, entregas rápidas são cobradas, o que prejudica a qualidade do código?*). Das 45 empresas que possuem projeto com dívida técnica, 39 dos respondentes disseram que ocasionalmente, frequentemente ou com muita frequência são cobradas entregas rápidas durante o cotidiano, o que prejudica a qualidade do código. De acordo com [13] essa é uma das principais razões para aquisição de dívida técnica. Para [40], isso está em acordo com o fator de priorização.

4.2 Preocupação com o pagamento da dívida técnica nos sistemas legados

Como citado anteriormente, foi considerada a definição de sistemas legados proposta por [35]: sistemas que estão em produção a cinco anos ou mais. Partindo desta definição, foram contabilizados 47 respondentes que trabalham em empresas com sistema legado. Para avaliar se a empresa preocupa-se com o pagamento de dívida técnica foram levadas em consideração as perguntas:

- A empresa em que trabalho costuma se preocupar em resolução de problemas/defeitos conhecidos (Q21).
- Na empresa em que trabalho costuma-se realizar refatoração de código (Q31).
- Após inseridas as dívidas técnicas, existe organização da equipe para que seja realizado pagamento da d´ıvida (Q33).

As três perguntas devem contabilizar, no máximo, 9 pontos, dentro do critério de pontuação de pesos elaborado, para os respondentes que trabalham em empresas que não se preocupam com a dívida técnica. Esse critério de pontuações é aquele apresentado na Tabela 2. Como as perguntas são de cunho positivo, todas as respostas devem estar na faixa de *Nunca* e *Ocasionalmente*.

Percebeu-se que, das 47 empresas que possuem sistema legado, 13 não se preocupam com o pagamento da dívida técnica conforme é apresentado na Figura 4. Em relação às 28 empresas que não possuem sistema legado, apenas duas delas não se preocupam com o pagamento.

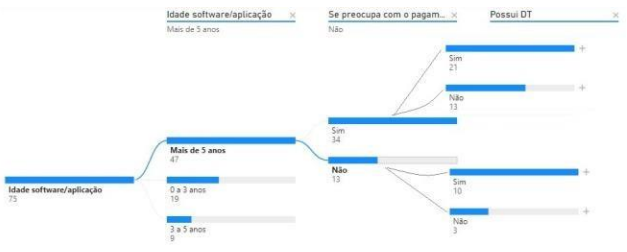

Figura 4: Relação da preocupação com o pagamento da dívida técnica

Dentre as 13 empresas que não se preocupam com o pagamento da dívida técnica, 10 foram caracterizadas como possuindo projeto com dívida técnica. Portanto, exclui-se a hipótese de que as empresas não se preocupam com o pagamento por não haver dívida técnica.

As empresas que possuem projeto com dívida técnica e se preocupam com o pagamento podem inserir e usar a d´ıvida a seu favor de forma intencional, para fazer entregas mais rápidas. De acordo com [13], a razão da contração da dívida técnica se dá, em sua maioria, por pressão por parte do negócio, pois há necessidade de atender as demandas de negócio em um curto espaço de tempo. Essas empresas que inserem dívida técnica se preocupam com o planejamento para o pagamento, seja este parcial ou por completo.

Desse modo, no geral, é possível perceber que as empresas de sistema legado e colaboradores se preocupam com o pagamento da dívida técnica, mesmo que o pagamento fique em segundo plano e as necessidades de negócio venham primeiro. Somente com as perguntas realizadas, não é possível concluir se o pagamento ocorre de forma parcial ou por completo, pois a atividade pode variar de acordo com a forma de gerenciamento da empresa. Vale ressaltar que, apesar do termo dívida técnica ter sido proposto a mais de duas décadas, muitas empresas não o conhecem muito bem [23]. Assim, pesquisas quantitativas na área são desafiadoras e complexas, principalmente no reconhecimento se uma manutenção foi gerada ou não por uma dívida técnica. Neste trabalho, o uso de conceitos de métricas de códigos-fonte auxiliou neste desafio.

4.3 Ameaças à validade

Este estudo buscou associar a dívida técnica com as caracter´ısticas das empresas em que os respondentes trabalham e utilizam os fatores de aquisição de dívida técnica abordados na literatura. Um questionário foi utilizado para identificar as respostas necessárias para a elaboração do estudo. Uma validação prévia do questionário foi proposta para garantir a qualidade das perguntas e também o tempo médio de resposta. Para isso, foram selecionados especialistas na área de desenvolvimento de software: quatro desenvolvedores e um gerente de desenvolvimento. Durante a avaliação prévia do questionário, observou-se uma dúvida sobre o conceito de dívida técnica. Desta forma houve um ajuste no questionário e o conceito de dívida técnica foi incluído antes da pergunta número 13. Esse ajuste reduziu a ameaça da falta de compreensão correta dos termos tratados na pesquisa.

A seleção dos respondentes foi definida para garantir a maior segurança em relação a qualidade das respostas. Por isso, o questionário foi direcionado aos desenvolvedores que fazem parte do processo de desenvolvimento, mais especificamente a fase de programação, em empresas que produzem software.

A construção do questionário, embasado na literatura, a seleção dos respondentes, sendo profissionais atuantes no desenvolvimento de software, além da amostra representativa foram critérios aplicados e contribuem para a validade interna do estudo.

Não houve a análise dos códigos fonte das empresas pesquisadas porque existem muitas restrições de acesso a essas informações por parte das empresas desenvolvedoras. Para garantir a qualidade dos dados coletados, esta pesquisa considerou que os desenvolvedores conheciam os códigos-fonte aos quais se referiam durante as respostas às questões apresentadas.

Apesar desta pesquisa ter sido aplicada em uma região específica do estado de Santa Catarina, no Brasil, empresas de outras regiões podem ter as mesmas características identificadas neste trabalho, e desta forma os resultados podem ser extrapolados para estas outras empresas representando a sua validade externa.

5. TRABALHOS RELACIONADOS

A literatura apresenta alguns trabalhos que traçam o tratamento das dívidas técnicas em sistemas legados. Por exemplo, em [20], uma pesquisa foi realizada em empresas para identificar a relação de métodos ágeis e dívidas técnicas. O objetivo do trabalho foi identificar como as dívidas técnicas são inseridas em projetos utilizando métodos ágeis. Porém, como um dos resultados da pesquisa realizada, sistemas legados foram identificados como origem das dívidas técnicas. Os autores identificaram que, apesar de não ser conclusivo, muitas dívidas técnicas residem em sistemas legados.

Em [28], os autores realizaram uma pesquisa com 16 empresas de grande porte. Foram realizadas 16 entrevistas para o levantamento dos dados. Como um dos resultados, foi identificado que os participantes percebem sistemas legados como sistemas desenvolvidos com tecnologias antigas sobre *hardware* antigo. E que tais sistemas estão perto do fim de suas vidas úteis. Foi identificado que gerenciar dívidas técnicas em sistemas legados é bastante complexo e classificaram essas d´ıvidas como *Ecosystem Debt*, *i.e.*, um conjunto de sistemas que envolve os sistemas legados e outros criados para suportar tais sistemas.

Os trabalhos de [33] e [34] aplicaram questionários para identificar a relação da dívida técnica e desenvolvimento de software no Brasil. No primeiro trabalho, os autores tentam identificar os diferentes problemas relacionados à criação e ao pagamento de dívidas técnicas na perspectiva dos desenvolvedores, sem focar no gerenciamento da DT, apenas na causa da inserção. Foram coletados respostas de 74 participantes entre fevereiro e março de 2017. Os participantes foram convidados utilizando uma lista de estudantes da UFRGS, pesquisadores da área e *posts* publicados em redes sociais. Os resultados foram agrupados de duas formas: o entendimento de o que é DT e a adoção de boas práticas de desenvolvimento. Na primeira forma, 89% conhecem o termo mas com denominações diferentes, por exemplo: custo de manutenção, código de baixa qualidade e custo × benefício de entregar código em desenvolvimento. Seis participantes disseram não ter a menor ideia de o que era TD. Quanto às boas práticas, a revisão de código foi considerada a mais relevante.

O segundo trabalho apresenta resultados obtidos após a análise de 40 respostas de um questionário para analisar a caracterização e gerenciamento da dívida técnica em empresas de software. Entre os achados, foi conclu´ıdo que uma pequena quantidade de empresas possui gerenciamento de dívidas técnicas em seus processos. Os participantes foram selecionados a partir de eventos relacionados a software: Simpósio Brasileiro de Qualidade de Software (SBQS) e Congresso Brasileiro de Software: Teoria e Prática (CBSoft), entre 2017 e 2018.

A análise proposta neste estudo se aproxima mais dos dois últimos trabalhos citados por terem sidos aplicados no Brasil. Porém, esta proposta não considerou que os participantes tivessem o perfil de pesquisador. Apesar de não existir

nenhuma pergunta no questionário para identificar se o participante fazia pesquisa na área, foi identificado empiricamente que todos não tinham contato com a academia para o desenvolvimento de seus sistemas. Assim, acredita-se que a análise realizada neste trabalho está mais próxima da caracter´ıstica do mercado brasileiro.

6. CONCLUSÃO

Sistemas precisam estar em constante atualização para satisfazer necessidades do negócio, e por isso, o custo de manutenção do software é considerado parte significativa de todo o ciclo de vida. Para isso, é preciso desenvolver de forma rápida e desconsiderar algumas boas práticas de programação, o que pode levar a inserção da dívida técnica.

Entretanto, mesmo que haja a exigência de entregas rápidas, a dívida técnica deve ser gerenciada. Caso as complicações sejam tratadas e gerenciadas, entende-se que é feito o pagamento da dívida técnica. O não pagamento pode acarretar aumento da complexidade do código-fonte a médio e longo prazo e, consequentemente, aumentar o custo de manutenção.

Este trabalho abordou, especificamente, uma análise de dívida técnica de código-fonte com foco em sistemas legados. Uma pesquisa foi conduzida com empresas do Oeste Catarinense para identificar a relação dívida técnica e sistemas legados. Observou-se que sistemas legados têm mais propensão a ter dívida técnica, o que era esperado, pois tais sistemas estão há mais tempo em produção.

A idade dos sistemas não indica que apenas sistemas legados estão sujeitos à dívida técnica e isso deve ser analisado mais detalhadamente em pesquisas futuras. Sugere-se, então, como trabalhos futuros abordar outras categorias de artefatos de software, não somente de código-fonte, e avaliar se há uma relação destas outras categorias com sistemas legados.

7. REFERÊNCIAS

[1] ACATE. Tech report 2019 - panorama do setor de tecnologia catarinense, 2018.

[2] E. Allman. Managing technical debt. *Communications of the ACM*, 55(5):50–55, 2012.

[3] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman. Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162). *Dagstuhl Reports*, 6(4):110–138, 2016.

[4] K. Awalegaonkar, R. Berkey, G. Douglass, and A. Reilly. AI: Built to scale - from experimental to exponential. https://www.accenture.com/us-en/insights/artificial-intelligence/ai-investments, 2019.

[5] H. K. A. Bakar, R. Razali, and D. I. Jambari. Implementation phases in modernisation of legacy systems. In *2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS)*, pages 1–6. IEEE, 2019.

[6] S. Cetin, N. I. Altintas, H. Oguztuzun, A. H. Dogru, O. Tufekci, and S. Suloglu. Legacy migration to service-oriented computing with mashups. In *International Conference on Software Engineering Advances (ICSEA 2007)*, pages 21–21. IEEE, 2007.

[7] N. Chapin, J. E. Hale, K. M. Khan, J. F. Ramil, and W.-G. Tan. Types of software evolution and software maintenance. *Journal of software maintenance and evolution: Research and Practice*, 13(1):3–30, 2001.

[8] C.-Y. Chen, K.-Y. Tai, and S.-S. Chong. Quality evaluation of structural design in software reverse engineering: A focus on cohesion. *IEEE Access*, 9:109569–109583, 2021.

[9] A. S. Chervenski and A. S. Bordin. Understanding legacy systems in the light of grounded theory. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*, pages 344–353, 2020.

[10] S. Chidamber and C. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

[11] W. Cunningham. The WyCash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30, 1992.

[12] W. Cunningham. The wycash portfolio management system. *SIGPLAN OOPS Mess.*, 4(2):29–30, Dec. 1992.

[13] R. G. de Oliveira. Caracterização e conceituação teórica da metáfora de débito técnico através de um estudo exploratório. Master's thesis, Universidade Federal de Pernambuco, 2011.

[14] D. N. Z. e Dra. Carolyn Seaman. Identifying and managing technical debt, 2012.

[15] J. Elm. Design debt economics: A vocabulary for describing the causes, costs, and cures for software maintainability problems, 2009.

[16] L. Erlikh. Leveraging legacy system dollars for e-business. *IT Professional*, 2(3):17–23, 2000.

[17] M. Fowler. Technical debt quadrant. www.martinfowler.com/bliki/TechnicalDebtQuadrant.html, 2009.

[18] M. Fowler. Technical debt. martinfowler.com/bliki/TechnicalDebt.html, 2019.

[19] J. Grubbs. Are technical debt and legacy systems affecting your digital transformation?, 2018.

[20] J. Holvitie, S. A. Licorish, R. O. Sp´ınola, S. Hyrynsalmi, S. G. MacDonell, T. S. Mendes, J. Buchan, and V. Leppänen. Technical debt and agile software development practices and processes: An industry practitioner survey. *Information and Software Technology*, 96:141–160, 2018.

[21] P. Kruchten, R. Nord, and I. Ozkaya. *Managing Technical Debt: Reducing Friction in Software Development*. Addison-Wesley Professional, 2019.

[22] V. Lenarduzzi, T. Besker, D. Taibi, A. Martini, and F. A. Fontana. A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools. *Journal of Systems and Software*, 171:110827, 2021.

[23] Z. Li, P. Avgeriou, and P. Liang. A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101, 12 2014.

[24] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):65–68, 1932.

[25] M. Lorenz and J. Kidd. *Object-Oriented Software Metrics: A Practical Guide*. Prentice-Hall, Inc., USA, 1994.

[26] R. C. Martin. *Código limpo*. Alta Books - Rio de Janeiro, Rio de Janeiro, 2011.

[27] P. Meirelles. *Monitoramento de métricas de código-fonte em projetos de software livre*. PhD thesis, Universidade de São Paulo, São Paulo, SP, 2013.

[28] B. D. Monaghan and J. M. Bass. Redefining legacy: a technical debt perspective. In *International Conference on Product-Focused Software Process Improvement*, pages 254–269. Springer, 2020.

[29] G. J. Myers. An extension to the cyclomatic measure of program complexity. *ACM Sigplan Notices*, 12(10):61–64, 1977.

[30] V. Nguyen, S. Deeds-rubin, T. Tan, and B. Boehm. A sloc counting standard. In *COCOMO II Forum 2007*, California, EUA, 2007. Center for Systems and Software Engineering.

[31] R. S. Pressman and B. R. Maxim. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Higher Education, Rio de Janeiro, RJ, 2014.

[32] R. J. Richardson, J. A. Peres, et al. *Pesquisa social: métodos e técnicas*. Atlas São Paulo, São Paulo, SP, 1985.

[33] J. C. Rocha, V. Zapalowski, and I. Nunes. Understanding technical debt at the code level from the perspective of software developers. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*, pages 64–73, 2017.

[34] V. M. Silva, H. J. Junior, and G. H. Travassos. A taste of the software industry perception of technical debt and its management in brazil. *Journal of Software Engineering Research and Development*, 7:1–1, 2019.

[35] H. M. Sneed. Encapsulation of legacy software: A technique for reusing legacy software components. *Annals of Software Engineering*, 9(1):293–313, 2000.

[36] I. Sommerville. *Engenharia de Software*. Pearson Universidades, São Paulo, SP, 2019.

[37] T. Soroker. Technical debt is costing you more than you think, 2019.

[38] R. Sp´ınola, N. Zazworka, A. Vetro, C. Seaman, and F. Shull. Investigating technical debt folklore: Shedding some light on technical debt opinion. In *International Workshop on Managing Technical Debt (MTD)*. IEEE, 2013.

[39] E. Tempero. On measuring java software. In *Proceedings of the Thirty-First Australasian Conference on Computer Science*, ACSC '08. Australian Computer Society, Inc., 2008.

[40] E. Tom, A. Aurum, and R. Vidgen. An exploration of technical debt. *Journal of Systems and Software*, 86(6):1498–1516, 2013.

[41] I. Warren and J. Ransom. Renaissance: A method to support software system evolution. In *Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International*, pages 415– 420, Oxford, UK, 02 2002. IEEE Xplore.

[42] N. Zazworka, A. Vetro, C. Izurieta, S. Wong, Y. Cai, C. Seaman, and F. Shull. Comparing four approaches for technical debt identification. *Software Quality Journal*, 22:1–24, 09 2013.

APÊNDICE

A. QUESTIONÁRIO

1. Qual é sua atual função?
   (a) Arquiteto de software
   (b) Desenvolvedor
   (c) Analista de software
   (d) Outros (Resposta aberta)
2. Há quanto tempo você trabalha com desenvolvimento de software?
   (a) 0 a 3 anos
   (b) 3 a 5 anos
   (c) 5 a 10 anos
   (d) Mais de 10 anos
   (e) Outros
3. Em qual cidade se localiza a empresa em que você trabalha?
   (a) Inclusas para seleção todas as 120 cidades pertencentes ao Oeste de Santa Catarina
4. Quantos anos possui a empresa que você trabalha? Se não souber indique uma idade aproximada
   (a) 0 a 3 anos
   (b) 3 a 5 anos
   (c) 5 a 7 anos
   (d) Outros
5. Quanto tempo tem o software/aplicação com que você trabalha? Indique o mais antigo.
   (a) 0 a 3 anos
   (b) 3 a 5 anos
   (c) 5 a 7 anos
   (d) 7 a 10 anos
   (e) Outros
6. Quantos funcionários possui a empresa que você trabalha? Se não souber indique um número aproximado.
   (a) Até 50 funcionários
   (b) 50 a 100 funcionários
   (c) 100 a 200 funcionários
   (d) 200 a 500 funcionários
   (e) 500 a 1000 funcionários
   (f) Mais de 1000 funcionários
7. Quais as tecnologias de desenvolvimento (linguagem de programação, *frameworks* utilizada na empresa ou projeto que você trabalha?
   Resposta descritiva
8. A sua empresa utiliza softwares para inspeção de qualidade do código? (SonarQube, Codacy, Code Climate, outros)
   (a) Sim
   (b) Não
9. Se você respondeu "Sim", qual o *software* utilizado? Resposta descritiva
10. Você conhece o conceito de dívida técnica?
    (a) Sim (Será direcionado para a pergunta 11)
    (b) Não (Será direcionado para a pergunta 12)

11. Como você descreveria o que é dívida técnica?
    Resposta descritiva
12. Só de ouvir o conceito ”Dívida técnica”, o que você acha que dívida técnica pode significar?
    Resposta descritiva
    As perguntas 13 a 15 tratam-se de perguntas com respostas de escala Likert para indicar seu grau de concordância com as afirmações: 1 - Nunca, 2 - Raramente, 3 - Ocasionalmente, 4 - Frequentemente, 5 - Muita frequência

    O conceito de dívida técnica é uma metáfora a dívida financeira. Quando o desenvolvimento é acelerado e o código fica imaturo e desorganizado, é necessário reescrevê-lo. O custo da reescrita é tratado como dívida técnica.

    Com base nesta definição responda as perguntas a seguir:
13. Encontro problemas nos projetos de arquitetura que devem ser resolvidos antes de iniciar o desenvolvimento do software.
14. Comunico os problemas de arquitetura aos responsáveis pelo projeto sempre que os encontro.
15. Ao encontrar problemas de arquitetura durante o desenvolvimento de software, eu ou alguém de minha equipe corrige esses problemas.
16. Tendo em vista o conceito de dívida técnica você acha que código de má qualidade, como por exemplo, acoplamentos desnecessários, falta de abstração, duplicação de código e soluções rápidas em vez de elegantes, constitui dívida técnica?
    (a) Sim
    (b) Não
17. Você acha que a arquitetura de sistema degradada por falta de refatoração estrutural constitui dívida técnica?
    (a) Sim
    (b) Não
18. Você acha que problemas e defeitos conhecidos que não foram resolvidos constituem dívida técnica?
    (a) Sim
    (b) Não
19. Você acha que o fato de a dívida técnica não ser facilmente identificada e comunicada é uma causa do seu acúmulo?
    (a) Sim
    (b) Não

    As perguntas 20 a 23 tratam-se de perguntas com respostas de escala Likert para indicar seu grau de concordância com as afirmações: 1 - Nunca, 2 - Raramente, 3 - Ocasionalmente, 4 - Frequentemente, 5 - Muita frequência
20. Ao desenvolver me deparo com acoplamentos desnecessários, falta de abstração e duplicação de código no projeto de código fonte.
21. A empresa que trabalho costuma se preocupar em resolução de problemas/defeitos conhecidos.
22. Na empresa em que trabalho há preocupação em identificar a dívida técnica.
23. Quando a dívida técnica é identificada e comunicada ao responsável há um planejamento para que a dívida técnica seja paga.
24. Você acha que a dívida técnica se torna mais cara para pagar com o passar do tempo?
    Respostas:
    (a) Sim
    (b) Não

    As perguntas 25 a 33 tratam-se de perguntas com respostas de escala Likert para indicar seu grau de concordância com as afirmações: 1 - Nunca, 2 - Raramente, 3 - Ocasionalmente, 4 - Frequentemente, 5 - Muita frequência
25. Durante o meu cotidiano entregas rápidas são cobradas prejudicando a qualidade do código.
26. Busco aplicar boas práticas de programação durante meu dia a dia.
27. Você já codificou um método muito longo por não saber como abstraí-lo em métodos menores?
28. Já precisei criar duplicações no código por não ter conhecimento de todo o projeto.
29. Preciso realizar duplicação de código por não haver classes/métodos reutilizáveis.
30. Na empresa que trabalho é um habito aplicar padrões de projetos nos códigos.
31. Na empresa em que trabalho costuma-se realizar refatoração de código.
32. A empresa que trabalho faz a inserção de dívida técnica de forma consciente
33. Após inseridas as dívidas técnicas, existe organização da equipe para que seja realizado pagamento da d´ıvida.
34. Existe algo mais que você gostaria de compartilhar sobre suas experiências com desenvolvimento de software?
    Resposta descritiva