UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RAMON PERONDI

IDENTIFICAÇÃO E MONITORAMENTO DE DÍVIDA TÉCNICA NO SETOR DE TECNOLOGIA DA INFORMAÇÃO DA UNIVERSIDADE FEDERAL DA FRONTEIRA SUL - UM ESTUDO DE CASO EXPLORATÓRIO

CHAPECÓ
2014

RAMON PERONDI

IDENTIFICAÇÃO E MONITORAMENTO DE DÍVIDA TÉCNICA NO SETOR DE TECNOLOGIA DA INFORMAÇÃO DA UNIVERSIDADE FEDERAL DA FRONTEIRA SUL - UM ESTUDO DE CASO EXPLORATÓRIO

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientadora: Prof. Me. Graziela Simone Tonin

CHAPECÓ
2014

Perondi, Ramon

Identificação e Monitoramento de Dívida Técnica no Setor de Tecnologia da Informação da Universidade Federal da Fronteira Sul - Um estudo de caso exploratório / por Ramon Perondi. – 2014.
61 f.: il.

Orientadora: Graziela Simone Tonin
Trabalho de conclusão de curso (graduação) - Universidade Federal da Fronteira Sul, Ciência da Computação, Curso de Ciência da Computação, Chapecó SC, 2014.

1. Dívida técnica. 2. Qualidade de software. 3. Modelo de gestão de dívida técnica. 4. Identificação de dívida técnica. 5. Monitoramento de dívida técnica. I. Tonin, Graziela Simone, orient. II. Universidade Federal da Fronteira Sul. III. Título.

E-mail: rpperondi@gmail.com

RAMON PERONDI

IDENTIFICAÇÃO E MONITORAMENTO DE DÍVIDA TÉCNICA NO SETOR DE TECNOLOGIA DA INFORMAÇÃO DA UNIVERSIDADE FEDERAL DA FRONTEIRA SUL - UM ESTUDO DE CASO EXPLORATÓRIO

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Me. Graziela Simone Tonin

Este trabalho de conclusão de curso foi defendido e aprovado pela banca em: 12/12/14

BANCA EXAMINADORA:

(vídeo-conferência)

P.P. Me. Graziela Simone Tonin - UFFS

Dra. Raquel Aparecida Pegoraro - UFFS

Me. Fernando Bevilacqua - UFFS

AGRADECIMENTOS

Agradeço aos meus pais, Laudir e Ilene, a minha irmã, Carla, pela base familiar em que fui criado desde pequeno. Se hoje tenho bons valores são graças a estes que foram, são e sempre serão importantíssimos em minha vida.

Agradeço a melhor parte de mim em outra pessoa e fiel revisora de meus textos, Talita, por ser a namorada mais compreensiva e amável possível.

Agradeço aos amigos, peças extremamente necessárias para que o desânimo não abalasse o emocional em momentos difíceis.

Agradeço a Prof. Dra. Raquel Aparecida Pegoraro e ao Prof. Me. Fernando Bevilacqua, que dispuseram dos seu tempo para as bancas, por tornarem esse trabalho melhor com sugestões e críticas construtivas.

Um agradecimento todo especial a Prof. Me. Graziela Simone Tonin, orientadora que dedicou uma imensa parte de seu tempo e conhecimento para a pesquisa, por grandes ensinamentos e conselhos, que levarei tanto para a vida profissional como pessoal.

Agradeço também ao pessoal do setor de Tecnologia da Informação da UFFS, pelo apoio e pela recepção da pesquisa no local.

Por fim, agradeço a todos que de uma forma ou outra contribuíram para a realização deste trabalho, seja com atos concretos ou com bons pensamentos.

“I’ve been through the desert on a horse with no name
It felt good to be out of the rain
In the desert you can remember your name
’Cause there ain’t no one for to give you no pain”
— America

RESUMO

A metáfora dívida técnica, cunhada por Ward Cunningham em 1992, representa o comprometimento da qualidade de um código ao serem utilizadas práticas imaturas no desenvolvimento de software, justificadas por vários motivos. O perigo de contrair tais dívidas não está no fato de serem adquiridas, mas sim, na ausência de seus pagamentos, ou seja, quando elas não são pagas e acabam esquecidas no código. Neste sentido, o presente trabalho buscou identificar, no setor de Tecnologia de Informação da Universidade Federal da Fronteira Sul, as dívidas técnicas existentes e encontrar um modelo de gestão de dívidas que pudesse auxiliar na identificação e monitoramento das dívidas contraídas. Para que esses objetivos fossem alcançados, a presente pesquisa se propôs a definir o estado da arte buscando trabalhos relacionados a dívida. A presente pesquisa buscou ainda difundir a cultura da dívida técnica no caso estudado, sugerir formas de monitoramento de dívida técnica e identificar o impacto das sugestões dentro da equipe. Para a realização dessas ações, foi realizado um estudo de caso exploratório e utilizaram-se técnicas de coleta de dados qualitativos que foram analisados utilizando a metodologia denominada *Grounded Theory*. Um modelo de gestão de dívida técnica foi construído onde é possível visualizar formas de identificação e monitoramento de dívida técnica. Este modelo foi criado em conjunto com os membros da equipe do caso estudado que o utilizaram durante um semestre. Ao final o mesmo foi validado através de entrevistas onde identificou-se que o modelo auxiliou a equipe a refletir sobre os problemas de qualidade existentes no código bem como a criar um forma de geri-los.

Palavras-chave: Dívida técnica. qualidade de software. modelo de gestão de dívida técnica. identificação de dívida técnica. monitoramento de dívida técnica.

ABSTRACT

The technical debt metaphor, explained by Ward Cunningham in 1992, is compromised quality to be used immature practices in software development, justified by several reasons. The danger of contracting such debt is not in fact be acquired, but in the absence of your payments, i.e., when they aren't paid and end up forgotten in the code. In this sense, the present study sought to identify in Information Technology sector of Federal University of Fronteira Sul existing technical debt and find a debt management model that could help in the identification and monitoring of debts. To achieve these goals this research set out to define the state of the art seeking work related to technical debt. Present research sought to further spread the culture of technical debt in the case studied, suggest ways of technical debt monitoring and identify the impact of the suggestions within the team. To accomplish these actions, it was performed an exploratory case study and used qualitative data collection techniques that were analyzed using the methodology called *Grounded Theory*. A technical debt management model was built where it's possible view forms of identification and monitoring technical debt. This model was created in conjunction with members of the study team that used during a semester. At the end it was validated through interviews where it was identified that the model helped the team to reflect on the existing quality problems in the code and create a way to manage them.

Keywords: Technical debt. software quality. technical debt management model. technical debt identification. techcnical debt monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACM	*Association for Computing Machinery*
CDT	Capital de dívida técnica
DRA	Diretoria de Registro Acadêmico
GT	*Grounded Theory*
ISO	*International Organization for Standardization*
OOPSLA	*Objected Oriented Programming System, Languages & Application*
SETI	Secretaria Especial de Tecnologia e Informação
SQALE	*Software Quality Assessment based on Lifecycle Expectations*
TI	Tecnologia da Informação
UFFS	Universidade Federal da Fronteira Sul

SUMÁRIO

1 INTRODUÇÃO

Gestores e desenvolvedores de software muitas vezes discordam acerca de decisões importantes como a necessidade de investir em qualidade interna do código. Mesmo quando há um acordo entre as partes, ainda pode existir o empecilho de um ou mais executivos que possam vir a questionar para onde estão indo os recursos empregados na melhoria da produção e acabam muitas vezes se recusando a aprovar essas melhorias internas. A situação se agrava em projetos que necessitam equilibrar prazos curtos com manutenibilidade a longo prazo [26].

Quando se fala em desenvolvimento de software frequentemente membros do projeto precisam tomar decisões rápidas. No entanto, a maioria das decisões tomadas pelos gestores de software tem o seu foco na manutenção, em prejuízo a evolução atual do sistema [44]. Manutenções de longa duração podem ser sintomas de dívida técnica, visto que a manutenção é considerada a fase mais custosa da produção, equivalendo a cerca de 90% do custo total do ciclo de vida do software [31].

De acordo com Cavano e McCall [4], para que o software final tenha o resultado esperado é necessário que o processo de produção também tenha os mais altos padrões de qualidade. Nesse sentido, as métricas de qualidade de software devem ser orientadas para a fase de desenvolvimento em substituição ao momento em que o software está em sua versão final.

Um projeto de software flexível, de fácil manutenção, extensível, reutilizável e que possua a inserção de novos requisitos ocorrendo de forma amigável são características de um projeto com a dívida técnica bem gerenciada [46].

Dívida técnica é uma metáfora utilizada pelas empresas de Tecnologia da Informação (TI) em todos os seus níveis de organização [30]. É uma nova forma de se concentrar na gestão a longo prazo de complexidades acidentais causadas por compromissos de curto prazo. Quando não se possui uma gestão correta de dívida técnica, acaba-se por incorrer em problemas a longo prazo como, por exemplo, custos com manutenção [2].

A visão a respeito da metáfora dívida técnica vem ganhando força nas comunidades de desenvolvimento de software como um método inovador para compreender e difundir as questões de qualidade, custo e valor do software [36, 26]. Apesar disso, de acordo com Lim et al. [32] cerca de 75% dos profissionais da área ainda não estão familiarizados com o termo.

Outros estudos realizados comprovam que a dívida técnica está crescendo em todo mundo. Uma pesquisa realizada em 2010 constatou que a dívida técnica mundial era de apro-

ximadamente 500 bilhões de dólares, com potencial para chegar a 1 trilhão de dólares em cinco anos. [18]

Mesmo com uma definição formal, a metáfora dívida técnica pode ser vista nas empresas como ‘os resultados invisíveis de decisões passadas sobre o software que afetam negativamente o seu futuro’ [26]. Um dos benefícios de uma empresa que identifica e gerencia dívida técnica é que ela estará preparada para eventuais problemas que essa dívida possa vir a causar [1]. Os desafios e metas das pesquisas sobre dívida técnica são identificar e quantificar a dívida existente nos projetos para que se encontre métodos eficazes de monitorá-la [53]. Apesar das abordagens de dívida técnica, no geral, seguirem um padrão semelhante de processos, projetos de redução de dívida técnica variam de uma empresa para a outra [19].

No entanto, entender e gerir dívida técnica de modo a conseguir com que o projeto tenha a qualidade mínima necessária é uma tarefa difícil, que só parece possível através de uma combinação de fatores de diferentes áreas da engenharia de software como estudos qualitativos, métricas de software e planejamento [14].

Deste modo, essa pesquisa tem como objetivo geral apresentar um estudo de caso exploratório no setor de TI da Universidade Federal da Fronteira Sul (UFFS) visando identificar as dívidas técnicas existentes no setor e encontrar um modelo de monitoramento de dívidas que melhor se adapte e contribua para a melhoria do processo de desenvolvimento neste caso. Para tanto, apresentam-se os objetivos específicos da pesquisa

- Definição do estado da arte buscando trabalhos relacionados a identificação e monitoramento de dívida técnica;
- Difundir a cultura da dívida técnica no caso estudado;
- Sugerir formas de monitoramento de dívida técnica;
- Identificar o impacto das sugestões dentro da equipe.

2 DÍVIDA TÉCNICA

2.1 A metáfora dívida técnica

A metáfora dívida técnica foi citada pela primeira vez por Ward Cunningham em 1992 no *Objected Oriented Programming System, Languages & Application* (OOPSLA) da *Association for Computing Machinery* (ACM) em 1992. A metáfora está descrita, em tradução livre, a seguir:

> "... a primeira vez que a qualidade do código é comprometida é como se estivéssemos incorrendo em dívida. Uma pequena dívida acelera o desenvolvimento até que seja pago através da reescrita do código. O perigo ocorre quando o dívida não é paga. Cada minuto em que o código é mantido em inconformidade, juros são acrescidos na forma de *refactoring*[1]..." [8]

Para entender melhor a metáfora, pode-se dividi-la em duas partes. Na primeira, Cunningham [8] enfatiza que se a qualidade do código é violada, ou seja, caso o membro da equipe utilize de práticas imaturas de codificação, a dívida é contraída. Na segunda parte, o autor ratifica a tendência das empresas em contrair a dívida para acelerar outros processos. Apesar de parecer interessante adquirir uma pequena dívida para adiantar alguma tarefa, isso é perigoso ao longo do projeto, pois durante o tempo em que essa dívida está no código, existem juros acrescidos e que deverão ser pagos por meio de *refactoring* [8].

Em 2009, Cunningham [9] publicou um vídeo que explica a dívida técnica. Entre outros tópicos, o autor destaca que lhe chamou atenção a forma como a metáfora influenciou os estudos sobre dívida técnica e que a criou para explicar o *refactoring* que estava sendo feito na empresa onde trabalhava.

Para Allman[1], a dívida técnica é resultado da lacuna entre as boas práticas de desenvolvimento e fatores como a falta de tempo ou o custo de ferramentas. Em outras palavras, a dívida acontece quando membros da equipe buscam atalhos que ficam aquém das melhores práticas.

A dívida pode se tornar uma âncora que impede o sistema de evoluir de forma eficiente, ocasionando menor produtividade e maior custo de manutenção [13]. Contudo, é necessário diferenciar dívida técnica de defeitos ou falhas. Durante o período de testes, as falhas podem ser sintomas de dívida técnica, mas a maior parte dessas falhas que causam a dívida podem não

[1] *Refactoring* é uma técnica utilizada na fase do desenvolvimento de software para melhorar a qualidade do sistema [46]. É a reestruturação do código, de modo a alterar sua estrutura interna sem modificar sua estrutura externa [2, 53]

ser identificadas durante esse período, deixando o software menos eficiente [11].

O problema da dívida técnica não é assumi-la, mas sim esquecê-la. Muitas empresas assumem a dívida para atingir um objetivo e a esquecem, deixando-a fugir de seu controle. Por esse motivo, o objetivo não deve ser eliminá-la, mas sim, gerenciá-la. Deve-se aceitar a dívida técnica como inevitável, porém um "plano para mudança" deve existir, caso algo necessite ser remediado [1].

Em uma analogia, a dívida técnica pode ser comparada à dívida financeira. Toda vez que se incorre em uma dívida, é como se estivesse adquirindo um empréstimo financeiro. Cada minuto que essa dívida permanece no software, juros são acrescidos na forma de *refactoring*, da mesma forma que a cada momento em que o empréstimo não é quitado, o montante final a ser pago cresce incorporando juros em seu valor [9].

Neste trabalho, todo artefato imaturo identificado dentro do processo de desenvolvimento de software será considerado dívida técnica.

2.2 Classificação de dívida técnica

Para uma melhor compreensão da metáfora alguns autores preferem categorizar as dívidas encontradas em seus estudos. É o caso de McConnell [33] que divide dívida técnica em intencional e sem intenção. Se a dívida for classificada como intencional, o autor ainda faz outra divisão: a dívida a curto e a longo prazo.

2.2.1 Dívida técnica sem intenção

Dívida técnica sem intenção é resultado não-estratégico de um trabalho mal planejado [33]. A dívida sem intenção é contraída de forma involuntária. Como essa dívida não é esperada, pode-se dizer que ela causa um impacto negativo no projeto.

Alguns exemplos de dívida técnica sem intenção são: 'adquirimos uma nova empresa que havia acumulado dívida técnica e estamos lidando com isso', 'algumas funcionalidades não tiveram êxito na hora da apresentação para o cliente pois estas foram desenvolvidas por um programador inexperiente' ou 'havia problemas nos requisitos iniciais e por isso o *design* não ficou como o cliente esperava'.

2.2.2 Dívida técnica intencional

A dívida técnica intencional acontece normalmente por priorização de outras dívidas, mas também pode ser resultado de um cronograma muito rígido. A equipe pode contrair essa dívida numa atitude estratégica para otimizar o projeto.

McConnell [33] cita um exemplo para explicar a dívida técnica intencional: 'Não temos tempo para fazer o *merge* dessas duas bases de dados, por isso vamos utilizar um código improvisado que as mantém sincronizadas e corrigir após o envio'.

Outros exemplos de dívida técnica intencional: 'Hoje priorizaremos uma determinada funcionalidade pois o prazo de entrega está próximo e havíamos prometido essa parte do sistema funcionando para o cliente' ou 'O código será desenvolvido sem testes e depois da entrega faremos os mesmos'.

2.2.3 Dívida técnica a curto prazo

A dívida técnica a curto prazo é assumida quando não se possui uma solução mas se está próxima dela [33]. A dívida a curto prazo é utilizada para cobrir pequenas lacunas de tempo, por exemplo, a necessidade da entrega de uma funcionalidade em um curto período.

Mais exemplos desse tipo de dívida são: 'Hoje não faremos a documentação completa das alterações feitas no código mas amanhã finalizaremos isso' ou 'Todo custo com o café consumido esse mês será pago somente no final do mês'.

2.2.4 Dívida técnica a longo prazo

McConnell [33] vê a dívida técnica a longo prazo como uma decisão estratégica e proativa. Assumida com o intuito de suprir necessidades de grande porte, é uma dívida que, quando bem gerenciada, pode sobreviver anos dentro do projeto sem causar problema.

Um exemplo de dívida técnica a longo prazo: 'Esse algoritmo não é o melhor, mas supre as nossas necessidades pelos próximos 10 anos, então vamos continuar com ele'.

2.2.5 Quadrante de Martin Fowler

Fowler [17] define um quadrante para a classificação da dívida técnica. O autor a classifica em dívida técnica intencional e dívida técnica sem intenção, da mesma forma que McConnell [33]. Acrescenta-se, porém, uma nova divisão: a dívida técnica prudente e a dívida técnica

imprudente, formando assim um quadrante, ilustrado na Figura 2.1.

Figura 2.1: Classificação de dívida técnica [17], em tradução livre

Dívida técnica imprudente é uma sequência de más escolhas que podem resultar em altos juros ou em um longo período para pagamento da dívida. Uma dívida técnica imprudente, se intencional, é uma dívida assumida de forma negligente pela equipe. É perigosa pois, se for esquecida pode gerar juros impossíveis de pagar levando até mesmo à falência do projeto [17].

O exemplo: 'Contrataremos 5 funcionários por no mínimo 5 anos para o setor de manutenção' estaria incluído no quadrante de dívida técnica intencional-imprudente. Já o exemplo: 'Como não sabemos o que é documentação, não faremos' seria classificado no quadrante de dívida técnica sem intenção-imprudente.

A dívida técnica prudente normalmente é gerenciada ao invés de excluída. É uma dívida pequena e que fica em um local pouco acessado, pois causa pouco impacto e pode ser mantida no projeto [17].

O exemplo: 'Amanhã faremos a documentação que não fizemos hoje' é incluído no quadrante de dívida técnica intencional-prudente, no entanto, o exemplo: 'Mostramos o produto hoje para o cliente e descobrimos que acertamos na cor do *layout*' seria classificado como dívida técnica sem intenção-prudente.

2.3 Identificação de dívida técnica

Abordagens de identificação de dívida técnica podem ser classificadas como aquelas que elicitam instâncias dos seres humanos e ferramentas automatizadas dos mais variados tipos para detectar a dívida [54]. Do ponto de vista da gestão de dívida técnica, o objetivo de identificar, medir e monitorar a dívida é facilitar a tomada de decisão [44].

A significativa diferença entre uma abordagem humana e a abordagem computacional, é que a primeira é mais sensível ao fator de importância da dívida para o projeto. A abordagem computacional pode trazer vários resultados não tão importantes, enquanto a abordagem humana pode focar em alguns aspectos que sejam de maior interesse para a pesquisa [54].

Para maior conhecimento, a seguir serão listadas algumas das ferramentas de identificação de dívida técnica que são formas de abordagens computacionais.

O *SonarQube*, ou apenas Sonar, é uma plataforma de código aberto utilizada para inspeção de qualidade de software, possuindo suporte para mais de 25 linguagens de programação. Possui um *plugin* para identificar e estimar a dívida técnica em projetos de software. O *plugin* fornece uma implementação do método SQALE automatizado para melhor compreensão das dívidas identificadas. A visualização da dívida pode ser realizada por severidade, pelas características da *International Organization for Standardization* (ISO), por vencimento, entre outros [48].

FindBugs é um programa de código aberto criado por Bill Pugh e David Hovemeyer que procura por *bugs* em códigos da linguagem Java. Apesar de nem todo *bug* ser uma dívida técnica, alguns definitivamente podem ser. O *FindBugs* é uma ferramenta que tem como principal característica o fato de encontrar possíveis trechos que estejam tornando o código menos eficiente [24].

O *Code Climate* é uma ferramenta que auxilia na revisão automática do código, destacando pontos que possam gerar possíveis falhas. Foi criada inicialmente para a linguagem Ruby mas atualmente possui suporte para PHP, JavaScript e outras linguagens [6].

2.4 Dimensionamento de dívida técnica

As métricas de dimensionamento de dívida técnica podem ser divididas em estáticas e dinâmicas. As ferramentas que dimensionam as métricas estáticas analisam o código sem que esse esteja em execução, por exemplo, o tamanho de uma classe. Já, as ferramentas que

dimensionam métricas dinâmicas analisam o código em execução, por exemplo, o consumo excessivo de memória. Algumas métricas que auxiliam a estimar dívida técnica são: código duplicado, conformidade de regras, comentários ao longo do código, complexidade de classe e cobertura de testes [13].

Dois métodos que buscam dimensionar a dívida técnica em projetos são explicados a seguir. O primeiro é o método denominado *Software Quality Assessment based on Lifecycle Expectations* (SQALE) criado por Letouzey [30]. O segundo é um método utilizado para estimar o capital da dívida técnica (CDT) criado por Curtis [10].

2.4.1 Método SQALE

O método SQALE foi criado para todo tipo de projeto de qualquer tipo de linguagem. Seu objetivo é ser o mais direto possível com o menor número de "falsos positivos" na análise do código [30].

O método define quatro conceitos chaves:

- Qualidade do modelo: é utilizado para organização dos requisitos não-funcionais que se relacionam com a qualidade do código. Possui três níveis hierárquicos. No primeiro nível estão as características. No segundo nível, as sub-características. No terceiro nível, os atributos internos;

- Análise do modelo: são as regras utilizadas para normalizar as medidas e as violações relacionadas ao código;

- Os índices: utilizados para representar os custos. Um representa a dívida técnica da aplicação e os outros são utilizados para a análise da dívida;

- Os indicadores: são definidos para proporcionar uma representação visual da dívida no projeto [30].

Com relação a funcionalidade do método, na fase de qualidade do modelo, a organização define o padrão de qualidade desejado. Esse padrão é um modelo de qualidade requerido e não um conjunto das melhores práticas de codificação que normalmente é definido em múltiplas camadas e inclui a visão do desenvolvedor e a visão do cliente. Esse padrão de requisitos pode ser sobre implementação, nomes ou apresentação [30].

Para estimar a dívida técnica na organização, cada um dos itens do padrão é associado a uma função de remediação que informa o custo para que uma determinada correção seja feita. Este custo de remediação depende do que pode-se chamar de "ciclo de vida da remediação". O ciclo de vida de uma correção com relação a endentação de código é pequena se comparada ao ciclo de uma correção que diz respeito a remover conteúdo redundante. A dívida técnica existente no código é calculada através da soma dos custos de remediação para cada um dos itens, e é chamada de *SQALE Quality Index* [29]. A Figura 2.2 apresenta um exemplo de como a documentação dos itens e suas respectivas funções de remediação são armazenadas.

CARACTERÍSTICA	EVIDÊNCIA	REMEDIAÇÃO MICRO-CICLO	FUNÇÃO DE REMEDIAÇÃO
MANUTENÇÃO	A endentação do código deve seguir um padrão consistente.	Consertar erros com a ajuda da IDE.	2 minutos por arquivo, independente do número de violações.
CONFIANÇA	Não há variáveis de iteração modificadas no corpo de um *loop*.	Reescrever o código e o teste associado.	40 minutos por ocorrência.
CONFIANÇA	Todos os arquivos têm o teste de unidade com cobertura de código em pelo menos 70% do seu conteúdo.	Escrever testes adicionais.	20 minutos por arquivo.
TESTABILIDADE	Não existe um método com uma complexidade ciclomática maior que 12.	Corrigir com IDE e escrever testes.	1 hora por ocorrência, se a medida for menor que 24 e 2 horas se for maior.
TESTABILIDADE	Não há partes clonadas no código com 100 *tokens* ou mais.	Corrigir com IDE e escrever testes.	20 minutos por ocorrência.

Figura 2.2: Modelo exemplo de identificação de dívida técnica [29], adaptado pelo autor

Para analisar a dívida, o método SQALE identifica características de qualidade, como testabilidade, eficiência, segurança e portabilidade. Cada um dos itens do padrão também são associados a uma característica. Existe um indicador para representar a distribuição de dívida técnica em cada característica. Esse indicador é chamado de pirâmide [29].

A partir disso, é necessário identificar qual é o tipo de projeto em que a dívida técnica está sendo dimensionada. Projetos ágeis, por exemplo, trabalham constantemente com mudanças no código e as características necessárias para apoiar estes desenvolvimentos são testabilidade, confiabilidade e mutabilidade. Se as dívidas técnicas nessas três características são altas demais, o projeto não é ágil o suficiente [29].

2.4.2 Método para estimar o CDT

No contexto de dívida técnica, o CDT é o custo necessário para remediar violações no código de produção que caracterizam a dívida [10]. Segundo Falessi et al. [15], o CDT refere-se ao valor que deve ser pago para remover a dívida por completo.

O método busca estimar o CDT em função de três variáveis: o número de violações que devem ser consertadas na aplicação, as horas necessárias para consertar cada violação e o custo

do trabalho. A equação para o cálculo é:

$$\text{CDT por violação} = \sum vdas \cdot percent \cdot average \cdot value$$

Onde *vdas* são as violações de uma determinada severidade, *percent* é a porcentagem de violações a serem consertadas, *average* é a média de horas necessárias para consertar as violações e *value* é o valor da hora [10]. A severidade é definida pela equipe como valores máximos de dívida técnica permitidos em um determinado projeto.

A equação é aplicada para violações de alta, média e baixa severidade e o resultado total do CDT é dado pela expressão abaixo:

$$\text{CDT total} = CDTalta + CDTmedia + CDTbaixa$$

Onde CDT alta é o capital para violações de alta severidade, CDT media é o capital para violações de média severidade e CDT baixa é o capital para violações de baixa severidade [10].

2.5 Gestão de dívida técnica

De acordo com Buschmann [3], existem três maneiras possíveis de tomada de decisão perante uma dívida técnica: pagar os juros, pagar a dívida ou converter a dívida. Uma outra solução ainda pode ser acrescentada que é a estratégia de continuar com a dívida.

Quando os juros são pagos, sofrem-se as consequências da dívida e para lidar com ela incorrem-se em custos adicionais [3]. Por exemplo, se existe um programador na equipe que sempre programa da mesma forma e isso implica na contração de dívida, toda vez que outro membro da equipe precisar dispender seu tempo para arrumar o código feito pelo programador causador da dívida se estará pagando juros dessa dívida.

Se a opção for pagar a dívida, livra-se de vez dessa dívida ao custo de significativo esforço de *refactoring* extra [3]. No caso descrito anteriormente uma solução seria delegar alguém para treinar o programador a codificar no padrão desejado pela organização.

Caso a escolha seja converter a dívida, substitui-se sua fonte com uma solução que implicará em uma nova dívida, porém, normalmente, menor. Por exemplo, quando se opta por pagar a dívida do *design* e a documentação é atrasada por conta disso.

Por fim, pode-se ainda escolher continuar com a dívida técnica. De forma consciente, a dívida pode ser mantida no sistema, se bem monitorada. Como exemplo, pode-se imaginar um

caso onde um método não foi codificado da forma que deveria, mas supre as necessidades. A organização pode monitorar esse trecho de código e optar por continuar com ele enquanto ele continuar cumprindo com suas obrigações.

3 METODOLOGIA

Neste capítulo são apresentados os métodos utilizados neste trabalho: o método de pesquisa, as formas de coleta e o método para análise dos dados.

3.1 Método de Pesquisa

Um método de pesquisa é um conjunto de princípios de organização em torno do qual os dados empíricos são obtidos e analisados. Existe uma variedade de métodos que podem ser aplicados a qualquer problema de pesquisa e diversas vezes é necessário que o pesquisador combine vários desses métodos para compreender totalmente o problema. A escolha dos métodos depende da postura do pesquisador e do acesso aos recursos, por exemplo [12].

Após uma breve análise sobre os métodos de pesquisa, elaborada no projeto deste trabalho de conclusão, o estudo de caso foi o método de pesquisa escolhido. Lakatos e Andrade Marconi [28] definem o estudo de caso como: "refere-se ao levantamento com mais profundidade de determinado caso ou grupo humano sob todos os seus aspectos".

3.1.1 Estudo de caso

O método de pesquisa chamado estudo de caso é tradicionalmente utilizado em pesquisas que possuem caráter qualitativo, as quais procuram pesquisar e analisar dados com caráter mais profundo, descrevendo a complexidade e fornecendo maiores detalhes sobre as investigações, hábitos, tendências etc [28].

Neste sentido, Yin [52] conceitua estudo de caso como uma investigação empírica que procura entender um fenômeno contemporâneo em profundidade e no seu contexto real, especialmente em situações em que o fenômeno e o seu contexto não são claramente evidentes. Os estudos de caso oferecem compreensão profunda do como e do porquê certos eventos ocorrem, e podem revelar os mecanismos de causa e efeito desse evento [12].

A tipo do estudo de caso utilizado será o exploratório. Easterbrook et al. [12] relata que o estudo de caso exploratório é, normalmente, utilizado como investigação inicial de alguns fenômenos para derivar novas hipóteses e construir teorias.

Uma pré-condição para a elaboração de um estudo de caso é a definição clara da questão de pesquisa. É essa questão que elenca precisamente o que o estudo pretende mostrar, orienta a

seleção de casos e os tipos de dados a serem coletados [12].

Foram definidas duas questões para o projeto:

- De que forma podemos aplicar identificação e monitoramento de dívida técnica no setor de TI da UFFS?
- Qual o impacto adquirida pelo equipe após a utilização do modelo?

Após definidas as questões de pesquisa, inicia-se a seleção dos casos que serão estudados. Esta seleção foi realizada através da amostragem não-probabilística que é utilizada quando os entrevistados serão escolhidos porque são facilmente acessíveis ou porque os pesquisadores possuem algum motivo para crer que eles representarão a população. Este é um tipo de amostragem onde se corre o risco de ser parcial. No entanto, Kitchenham e Pfleeger [25] citam três motivos para a escolha dessa amostragem: quando a população alvo é difícil de identificar, quando a população é específica e de disponibilidade limitada e quando o exemplo é um estudo piloto. Esse tipo de amostragem ainda possui subdivisões e a presente pesquisa faz uso da seleção por conveniência.

Yin [52] ainda afirma que um estudo de caso pode analisar apenas um caso ou diversos. A presente pesquisa analisa apenas um único caso. O estudo de um único caso é justificável sobre diversos aspectos: é um caso que permite um teste crucial a respeito de uma teoria existente; é um evento raro ou exclusivo; é um caso representativo ou típico; é um caso revelador; ou o caso representa um propósito longitudinal. Dentro de um caso individual, ainda podem ser estudados sub unidades, o que permite um *design* mais complexo [52].

3.2 Técnicas de Coleta de Dados

Após a definição do método de pesquisa a ser utilizado, inicia-se a escolha da técnica de coleta de dados para ser possível obter com maior facilidade e precisão as informações desejadas.

Algumas das principais técnicas de coleta de dados que existem são classificadas como documentação indireta, documentação direta, observação direta intensiva e observação direta extensiva [27].

A documentação indireta é subdividida em pesquisa documental (ou de fontes primárias) e pesquisa bibliográfica (ou de fontes secundárias). Já a documentação direta “constitui-se, em

geral, no levantamento de dados no próprio local onde os fenômenos ocorrem. Esses dados podem ser obtidos de duas maneiras: através da pesquisa de campo ou da pesquisa de laboratório". A observação direta intensiva é realizada através de duas técnicas: observação e entrevista, enquanto a observação direta extensiva é realizada através de questionário, formulário, medidas de opinião e atitude ou de técnicas mercadológicas [27].

Este projeto fez uso das técnicas de observação e entrevista, componentes da observação direta intensiva, e de questionários, componente da observação direta extensiva.

3.2.1 Observação

Na utilização da técnica de observação, os sentidos do corpo humano são responsáveis pela coleta de dados de determinados aspectos da realidade [39]. A observação é uma técnica que não se restringe a ver e ouvir, ela busca também examinar fatos ou fenômenos que se deseja estudar. Ajuda o pesquisador a identificar e obter provas a respeito de eventos em que os envolvidos não têm consciência, mas que orientam seu comportamento. A técnica de observação possui diversas modalidades, as quais são utilizadas de acordo com as circunstâncias [27].

Quanto aos meios utilizados, a observação utilizada foi a assimétrica ou assistemática que 'consiste em recolher e registrar os fatos da realidade sem que o pesquisador utilize meios técnicos especiais ou precise fazer perguntas diretas' [39]. O que caracteriza esse tipo de observação é o fato de que os dados são obtidos de um experimento casual, sem que se tenha determinado previamente quais aspectos deverão ser observados [41].

Em relação à participação do observador, esta pesquisa utilizou-se da forma não participante, onde o pesquisador possui contato com o grupo ou realidade estudada, mas sem que haja interação com o mesmo [27]. Ainda pode-se categorizar as observações realizadas como individual, visto que estas foram realizadas apenas por um observador [39].

3.2.2 Entrevista

A entrevista é uma técnica de levantamento de dados primários que dá grande importância à descrição verbal fornecida pelo entrevistado. Possui a característica de ser realizada frente a frente com a pessoa fonte das informações [39].

Esta técnica pode ser caracterizada como padronizada ou estruturada, quando o pesquisador segue um roteiro preestabelecido, não padronizada ou não estruturada, onde existe um roteiro, no entanto, sem rigidez, ou seja, o pesquisador pode explorar mais amplamente algumas

das questões e em geral, as perguntas são abertas [39].

Nesta pesquisa foram utilizadas entrevistas semiestruturadas com perguntas abertas e fechadas visando maior eficiência na coleta de informações.

3.2.3 Questionário

O questionário é uma técnica de coleta de dados que consiste em uma série ordenada de questões que devem ser respondidas por escrito pelo participante, sem que haja a presença do pesquisador [27]. A linguagem do questionário deve ser simples e direta para que suas questões sejam compreendidas com facilidade pelo respondente. O questionário também deve ser objetivo, limitado em extensão e estar acompanhado de instruções que expliquem a sua natureza e a importância e a necessidade das respostas [39].

O pesquisador que desenvolver o questionário deve estar ciente dos tipos de perguntas que podem ser feitas e qual desses tipos trará melhores resultados. Elas podem ser abertas ou fechadas. A primeira permite ao informante responder livremente e emitir opiniões, a segunda, onde o respondente escolhe uma opção entre duas fornecidas, ou de múltipla escolha, as quais são perguntas fechadas mas que em sua resposta, apresentam mais do que apenas duas opções, abrangendo várias facetas do mesmo assunto [27].

Os questionários utilizados no projeto foram híbridos mesclando perguntas abertas e fechadas.

3.3 Técnica de Análise de Dados

Tendo em vista que este projeto de pesquisa utilizou técnicas de coleta de dados qualitativas, a técnica para análise de dados utilizada foi *Grounded Theory*.

A técnica chamada *Grounded Theory* (GT) é conceituada por Strauss e Corbin como:

> [...] "teoria que foi derivada de dados, sistematicamente reunidos e analisados por meio de processo de pesquisa. Neste método, coleta de dados, análise e eventual teoria mantêm uma relação próxima entre si. Um pesquisador não começa um projeto com uma teoria preconcebida em mente (a não ser que seu objetivo seja elaborar e estender a teoria existente). Ao contrário, o pesquisador começa com uma área de estudo e permite que a teoria surja a partir dos dados."[49]

As ideias iniciais e fundamentais da GT proposta por Glaser e Strauss, em 1967, são:

- O desenvolvimento simultâneo na coleta e na análise dos dados;
- A construção de códigos e categorias analíticas a partir dos dados, e não de hipóteses

preconcebidas e logicamente deduzidas;

- A utilização do método comparativo constante, que compreende a elaboração de comparações durante cada etapa da análise;
- O avanço no desenvolvimento da teoria em cada passo da coleta e da análise dos dados;
- A redação de memorandos para elaborar categorias, especificar as suas propriedades, determinar relações entre as categorias e identificar lacunas;
- A amostragem dirigida à construção da teoria, e não visando à representatividade populacional;
- A realização da revisão bibliográfica após o desenvolvimento de uma análise independente [5].

O GT é um método de pesquisa qualitativo que busca gerar teoria através de três elementos principais: conceitos, categorias e proposições. Os conceitos são as unidades básicas de análise, uma vez que é a partir deles, e não dos dados reais (brutos), que a teoria é desenvolvida. Os eventos percebidos através dos dados reais considerados como indicadores potenciais do fenômeno recebem rótulos conceituais. O pesquisador deve comparar eventos e nomear com o mesmo termo se estes eventos forem semelhantes [37].

O segundo elemento, as categorias, são mais elevadas em nível e mais abstratas. São geradas pelo mesmo processo de análise para destacar diferenças e semelhanças que é utilizado para a criação dos conceitos. A criação das categorias inicia-se da análise dos próprios conceitos desenvolvidos anteriormente. Analisam-se e agrupam-se os conceitos que são voltados para um mesmo processo, com o mesmo nome. O terceiro elemento da GT, proposições, é o processo de indicar relações entre as categorias e seus conceitos e entre categorias distintas [37].

O processo de desenvolvimento da GT ocorre em cinco fases (desenho da pesquisa, coleta dos dados, ordenação dos dados, análise dos dados e comparação com a literatura) subdivididas em nove passos, os quais são descritos por Pandit [37], na Figura 3.1 a seguir:

Assim, pode-se concluir que os fundamentos da GT, propostos por Glaser e Strauss em 1967 são de que a proposta inicial do método é a construção da teoria, e não somente a codificação dos dados coletados. Além disso, o pesquisador deve, como regra geral, definir um quadro conceitual que antecede o início da pesquisa, premissa que é definida para garantir que os conceitos possam imergir sem viés conceitual pré-definido. A análise e a conceitualização são

Fase		Atividade
FASE DESENHO DA PESQUISA		
Passo 1	Revisão da literatura	Definição da questão de pesquisa
Passo 2	Seleção de casos	Amostra teórica, não aleatória
FASE COLETA DE DADOS		
Passo 3	Desenvolver protocolo para coleta de dados rigoroso	Empregar múltiplos métodos para coleta de dados
Passo 4	Ir à campo	Sobrepor coleta e análise de dados
		Métodos de coleta de dados flexíveis
FASE ORDENAÇÃO DE DADOS		
Passo 5	Ordenar dados	Colocar os dados coletados em ordem cronológica
FASE ANÁLISE DE DADOS		
Passo 6	Analisar os dados do primeiro caso	Usar codificação
Passo 7	Amostra teórica	Replicação teórica através dos casos (Reiniciar no Passo 2 até a saturação teórica)
Passo 8	Fechamento	Atingir a saturação teórica, se possível
FASE COMPARAÇÃO DE LITERATURA		
Passo 9	Comparar a teoria emergente com a literatura	Comparações com quadros referenciais similares e conflitantes

Figura 3.1: Processo de Construção do *Grounded Theory* [37], adaptado pelo autor

obtidas através do processo principal de coleta e comparação constante dos dados, no qual cada fatia de informação é comparada com construtos e conceitos existentes, visando enriquecer uma categoria já existente, formar uma nova ou estabelecer novos pontos de relação entre categorias [38].

4 DESENHO DO ESTUDO DE CASO

4.1 Caracterização do caso

O presente estudo foi realizado na Secretaria Especial de Tecnologia e Informação (SETI) da UFFS e teve como objetivo sugerir um modelo de gestão das dívidas técnicas no setor. Para tanto, tornou-se necessário um maior conhecimento a respeito de como é organizado o setor e quais áreas poderiam receber a pesquisa.

A Universidade Federal da Fronteira Sul (UFFS) foi criada no ano de 2010 e junto com o início de suas atividades de ensino, pesquisa e extensão começaram os primeiros trabalhos do setor de TI.

Contendo 16 unidades funcionais, a SETI possui uma diretoria geral, a qual conta com três órgãos de *staff*, sendo elas: Divisão de Segurança da Informação, Setor de Administração de Banco de Dados e Serviço de Governança de TI. Subordinados a estes estão a Diretoria de Infraestrutura de TI e a Diretoria de Sistemas de Informação. É possível verificar o organograma completo da SETI na Figura 4.1.

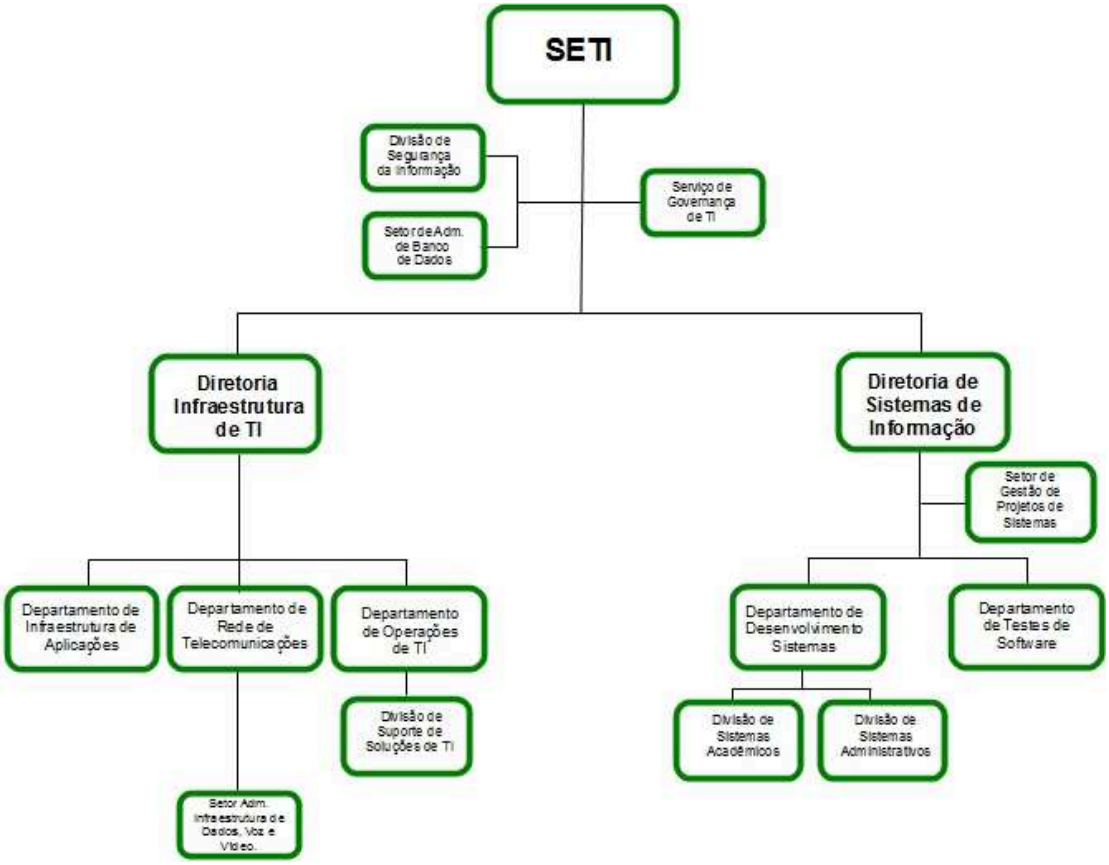

Figura 4.1: Organograma da SETI da UFFS [45]

A unidade de infraestrutura possui cinco departamentos, os quais não foram objetos de estudo da presente pesquisa. Já a Diretoria de Sistemas de Informação possui um órgão de *staff* chamado Setor de Gestão de Projetos de Sistemas. Como unidades subordinadas, existem o Departamento de Testes de Software e o Departamento de Desenvolvimento de Sistemas o qual possui a Divisão de Sistemas Acadêmicos e a Divisão de Sistemas Administrativos. Todas as unidades funcionais dessa diretoria juntas quantificam um total de doze funcionários e dois estagiários.

Analisando-se somente o organograma da Figura 4.1 não é possível perceber onde a Divisão de Segurança da Informação, o Setor de Administração de Banco de Dados e o Serviço de Governança de TI estão localizados fisicamente. O órgão Serviço de Governança de TI está alocado juntamente com a Diretoria de Infraestrutura, enquanto os órgãos Divisão de Segurança da Informação e a Administração de Banco de Dados estão localizados juntamente à Diretoria de Sistemas de Informação, se tornando assim parte do objeto de estudo.

O processo de trabalho da equipe é elaborado de acordo com o *framework Scrum*. De acordo com Schwaber [42], o *Scrum* possui quatro eventos formais: a reunião de planejamento da *sprint*, a reunião diária, a reunião de revisão da *sprint* e a retrospectiva da *sprint*. A SETI realiza a reunião de planejamento e a reunião diária, já a reunião de revisão é realizada sempre que necessário, mas não ao fim de cada *sprint* como é sugerido no *Scrum*, enquanto a reunião de retrospectiva não é realizada.

As *sprints* na SETI tem a duração de 10 dias úteis e possuem alguns passos bem definidos. O conjunto desses passos é denominado na presente pesquisa como "Fluxo da *Sprint*" e é apresentado na Figura 4.2.

No primeiro passo, o setor recebe conhecimento do pedido do requisitante (cliente) após este ter encaminhado a sua solicitação para a Diretoria de Registro Acadêmico (DRA). Em seguida é agendada uma reunião para que o requisitante possa discutir detalhes acerca dos requisitos enviados via DRA com a equipe de desenvolvimento.

A reunião de coleta de requisitos da *sprint* realizada pela equipe de desenvolvimento de sistemas acadêmicos reúne para a discussão o(s) requisitante(s) e uma parte da equipe, além de um membro da equipe de gerenciamento de projetos. Com os requisitos apresentados e, supostamente, compreendidos pela equipe de desenvolvimento, segue-se ao próximo passo: a realização da reunião de planejamento da *sprint*. Nessa reunião todos os membros da equipe de desenvolvimento de sistemas acadêmicos, acompanhados pela equipe de testes, fazem a

Figura 4.2: Fluxo da *Sprint* da SETI da UFFS

verificação da viabilidade do pedido e a estimativa de horas para cada requisito utilizando-se da técnica do *Planning Poker*.

Quando todas as tarefas têm seus tempos de execução definidos, passa-se a fase de codificação que ocorre em paralelo a fase de testes. Quatro desenvolvedores trabalham na codificação, enquanto um ou dois *testers* planejam e executam os testes. Esse ciclo ocorre até que todas as tarefas sejam concluídas ou até que o tempo de codificação/testes na *sprint* termine.

Após isso, o sistema desenvolvido é disponibilizado para homologação, onde os requisitantes recebem o sistema para ser testado no ambiente real de trabalho. Caso necessite ainda de alguma correção ou o sistema ainda possua alguma falha que não foi encontrada na fase de testes, e que possa ser resolvida até o final da *sprint*, ela é consertada e o software final é entregue ao requisitante.

4.2 Processo de aplicação do estudo no caso

Após evidenciar interesse em ser aplicada a pesquisa no setor, passou a ser executada a etapa que consistiu em explorar e compreender o funcionamento do setor. Quando a presente pesquisa começou, a equipe de desenvolvimento dos sistemas administrativos estava no decorrer de uma *sprint*, enquanto a equipe de desenvolvimento dos sistemas acadêmicos começaria a sua na próxima semana. Optou-se então por acompanhar a *sprint* da equipe de sistemas acadêmicos.

O acompanhamento iniciou com uma reunião de coleta de requisitos realizada na sede da Secretaria Especial de Gestão de Pessoas da UFFS no dia 9 de setembro. Para a coleta dos

dados da reunião optou-se por utilizar a observação assistemática não participante.

Cada requisitante teve uma maneira diferente de pautar e esclarecer suas necessidades para a equipe. Enquanto os membros do setor do primeiro módulo chegaram na reunião sem os requisitos especificados e com divergências entre eles, os requisitantes do segundo e do terceiro módulo haviam encaminhado previamente um documento para a equipe de desenvolvimento, o que facilitou a discussão, pois assim a equipe de desenvolvimento já possuía certo conhecimento a respeito das necessidades do requisitante de modo a encurtar consideravelmente o tempo.

Após a reunião de coleta, participou-se da reunião de planejamento da *sprint* com as estimativas de horas para cada tarefa. Novamente optou-se por coletar os dados na forma de observação assistemática não participante.

A reunião teve a participação da equipe de desenvolvimento, da equipe de testes e de um membro da equipe de gestão de projetos, além de dois observadores. Percebeu-se que alguns problemas constatados na reunião como a ideia de basear a estimativa das horas das tarefas na experiência sem executar o *Planning Poker*, como ele é sugerido nos mais diversos artigos acadêmicos [20, 23, 34], ou tomar conhecimento dos requisitos por parte da equipe de testes apenas na hora da reunião poderiam gerar dívidas técnicas futuras.

Após a etapa de estimativa de horas, o fluxo da *sprint* mostra que a próxima fase seria a codificação. No entanto, antes dessa fase ser iniciada foi realizada uma apresentação formal sobre o assunto dívida técnica a equipe do setor afim de proporcionar treinamento aos mesmos.

A apresentação dividiu-se em duas fases, sendo que a primeira teve como foco a explicação do termo dívida técnica, onde buscou-se expor a definição da metáfora cunhada por Cunningham. Mostrou-se também as possíveis classificações e noções acerca da gestão. A segunda etapa teve seu foco voltado para a presente pesquisa, onde pretendeu-se expor pontos como objetivos, justificativas e a metodologia.

Definiu-se nesta reunião o que seria considerado dívida técnica nos projetos do setor: qualquer artefato imaturo no processo de desenvolvimento. Além disso, foram apresentados alguns fatores como os tipos de dívida técnica que poderiam ser encontrados e dados que comprovaram a importância de identificar e monitorar dívida técnica no desenvolvimento de software.

Posterior a apresentação, ouviu-se a opinião dos funcionários do setor. Após um momento de discussão e esclarecimento de dúvidas sobre o trabalho, todos confirmaram estar dispostos a colaborar para a presente pesquisa. Acertou-se que as dívidas técnicas encontradas pela equipe seriam cadastradas num quadro de dívidas e a presente pesquisa continuaria a

analisar o setor para que pudesse encontrar um modo de prosseguir com o próximo passo.

O próximo passo do estudo foi a realização de entrevistas com membros da equipe de desenvolvimento e de testes onde utilizou-se de entrevistas não estruturadas com perguntas abertas e fechadas. Objetivou-se compreender aspectos do setor e dos membros relativos ao controle de qualidade de software aplicado no caso.

A próxima etapa do estudo se deu com a criação de um padrão para o cadastro das dívidas técnicas. Em seus estudos, Seaman [54] sugere um padrão para o cadastro que foi adotado pela equipe nesse caso. A Figura 4.3 mostra o exemplo de padrão que foi inserido no quadro do setor.

<table>
<tr><td colspan="2">Nome da Dívida: Segurança desativada</td><td>Data: 30/09/2014</td></tr>
<tr><td>Identificador: Ramon</td><td colspan="2">Local: Componente de segurança</td></tr>
<tr><td colspan="3">Descrição: Por problemas de performance parte da solução de segurança está desativada.</td></tr>
<tr><td>Tempo para pagamento: 6h</td><td colspan="2" rowspan="2">Probabilidade de gerar problemas: Alta</td></tr>
<tr><td>Valor real do pagamento: 9h</td></tr>
</table>

Figura 4.3: Padrão para cadastro de dívida técnica

No campo **Nome da Dívida** devem ser inseridas uma ou duas palavras que facilitem a identificação da dívida. A **Data** mostra quando a dívida foi identificada e o **Identificador** quem foi que a cadastrou. O **Local** da dívida é a localização desta dentro do sistema enquanto que no campo **Descrição**, deve-se realizar uma breve explicação sobre a dívida encontrada. A **Probabilidade de gerar problemas** é o campo onde o identificador expressa um nível que pode ser baixo, médio ou alto para a possibilidade da dívida gerar problemas futuros. O **Tempo para pagamento** é a quantia de horas que o identificador acredita que serão necessárias para o pagamento da dívida, enquanto o valor real do pagamento é o campo que será preenchido no momento em que a dívida é paga. A diferença entre o valor real e o tempo estimado para pagamento é a margem de erro da previsão inicial feita pelo identificador.

Após identificado e definido o padrão em que as dívidas seriam cadastradas, o próximo passo foi a confecção do quadro. A versão final foi fixada na parede do setor, como sugerido pelos próprios funcionários com o objetivo de que as dívidas ficassem expostas para todos do setor. A versão final do quadro já com algumas dívidas cadastradas pode ser vista na Figura 4.4.

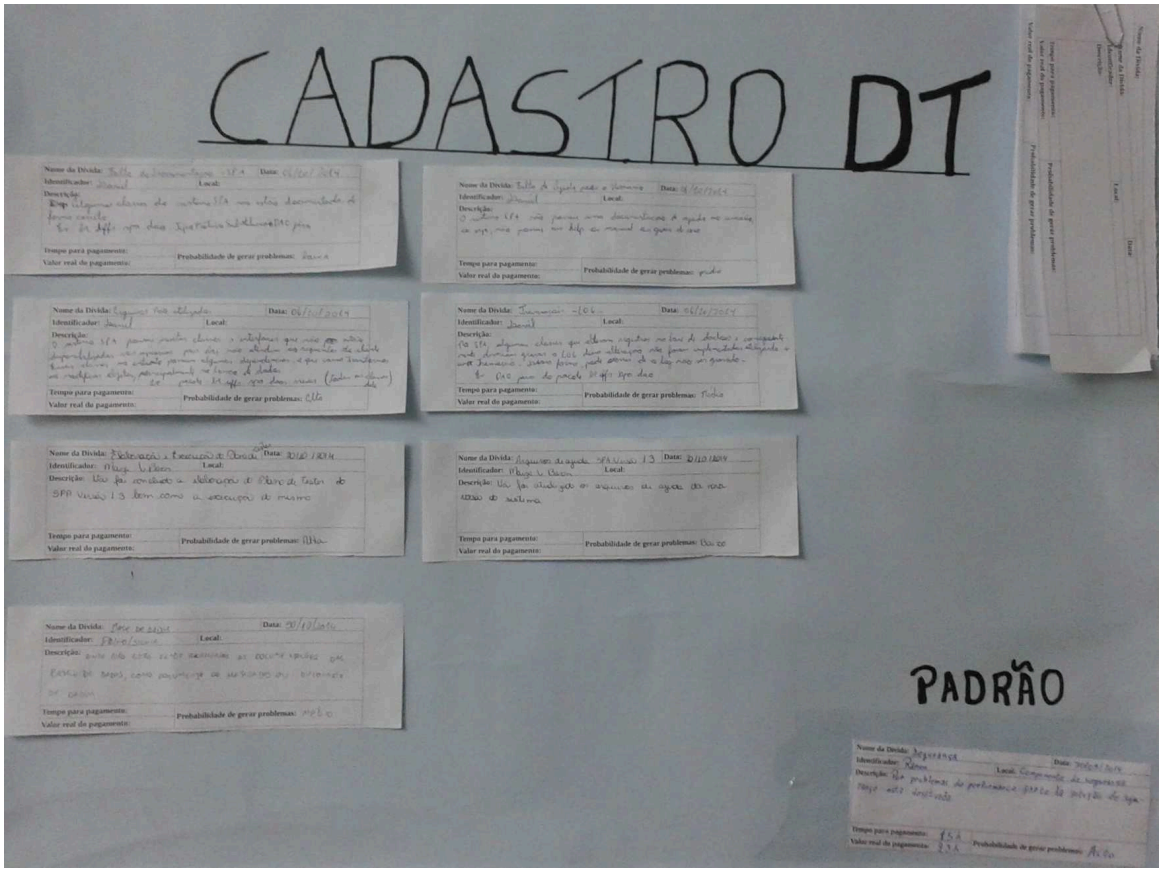

Figura 4.4: Quadro de dívidas técnicas do setor de TI da UFFS

A próxima etapa do estudo de caso teve o intuito de aplicar um questionário no setor com o objetivo de verificar se os membros compreenderam o que é dívida técnica e se as dívidas encontradas estavam sendo cadastradas por todos no setor.

No entanto, ao receber os resultados do questionário, percebeu-se a necessidade de um novo local para que as dívidas pudessem ser armazenadas principalmente por dois motivos: para que fosse possível gerir todo o histórico de dívida ao longo do ciclo de vida do projeto e pelo tamanho de armazenamento do quadro que é limitado. Entretanto, o quadro foi mantido para o cadastro das dívidas técnicas recentes e relevantes encontradas no setor, além destas serem replicadas no novo local de armazenamento.

Visando encontrar uma forma de manter o histórico das dívidas técnicas e armazenar todas as dívidas identificadas no setor, buscou-se analisar o processo e as ferramentas já utilizados pela equipe. Em conjunto com os membros, chegou-se a conclusão que as mesmas poderiam ser registradas na ferramenta de gestão de projetos já utilizada pela equipe: o *Redmine*.

O *Redmine* é uma aplicação web flexível utilizada para gerenciamento de projetos. Criado com o uso do *framework Ruby on Rails*, é multi-plataforma e multi-banco de dados. Possui

seu código aberto e é liberado segundo os termos da *GNU General Public License v2*[40]. O setor de TI da UFFS utiliza o *Redmine* principalmente para o cadastro de suas tarefas da *Sprint* e no caso de alguma alteração no sistema, seja ela evolutiva ou corretiva.

É inserido um identificador no momento do cadastro para que todos entendam que aquela informação é uma dívida e não uma tarefa. Esse identificador é a *string* 'dívida técnica', sendo inserida no campo **tipo** na hora do cadastro. A Figura 4.5 exibe a tela inicial do *Redmine* e a Figura 4.6 apresenta um exemplo de dívida técnica cadastrada no software.

Figura 4.5: Página inicial do *Redmine*, adaptado pelo autor

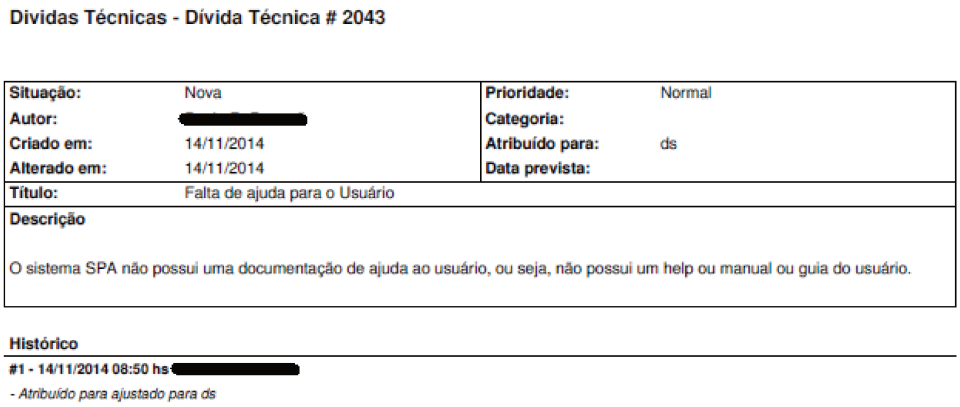

Figura 4.6: Exemplo de dívida técnica identificada e cadastrada no *Redmine*, adaptado pelo autor

As informações extras que não possuem campos específicos no software podem ser incluídas no campo descrição. A Figura 4.6 expõe dados importantes que não existiam no quadro de dívidas, como o histórico de alterações das dívidas e uma possível categoria associada

a dívida.

A pesquisa ainda buscou estudar a causa das dívidas mais comuns dentro do setor e se haveria alguma forma de evitá-las. Para que isso fosse possível, buscou-se as estatísticas das dívidas cadastradas no *Redmine* e analisou-se quais eram as dívidas que ocorreram com maior frequência na ferramenta.

Após coletados esses dados, compreendeu-se que as dívidas mais cadastradas tinham relação com documentação e com a fase de testes. Para investigar as causas desse tipo de dívida, realizaram-se duas entrevistas semiestruturadas com perguntas abertas e fechadas com dois membros do setor, um membro da equipe de desenvolvimento e um membro da equipe de testes. Ambos escolhidos já haviam cadastrados dívidas técnicas.

Os dados coletados foram ordenados e organizados na ferramenta *NVivo 10* para a posterior análise, utilizando-se da técnica de analise de dados qualitativos *Grounded Theory*.

O *NVivo 10* é um software utilizado para análise de dados qualitativos, independente do estágio em que a pesquisa se encontra. Dentre suas características estão a facilidade de codificar os dados de mídia social de forma rápida para visualizar os resultados e os padrões de auto codificação e para codificar grandes volumes de texto rapidamente [35]. A Figura 4.7 apresenta a ferramenta sendo utilizada para a análise dos dados da presente pesquisa.

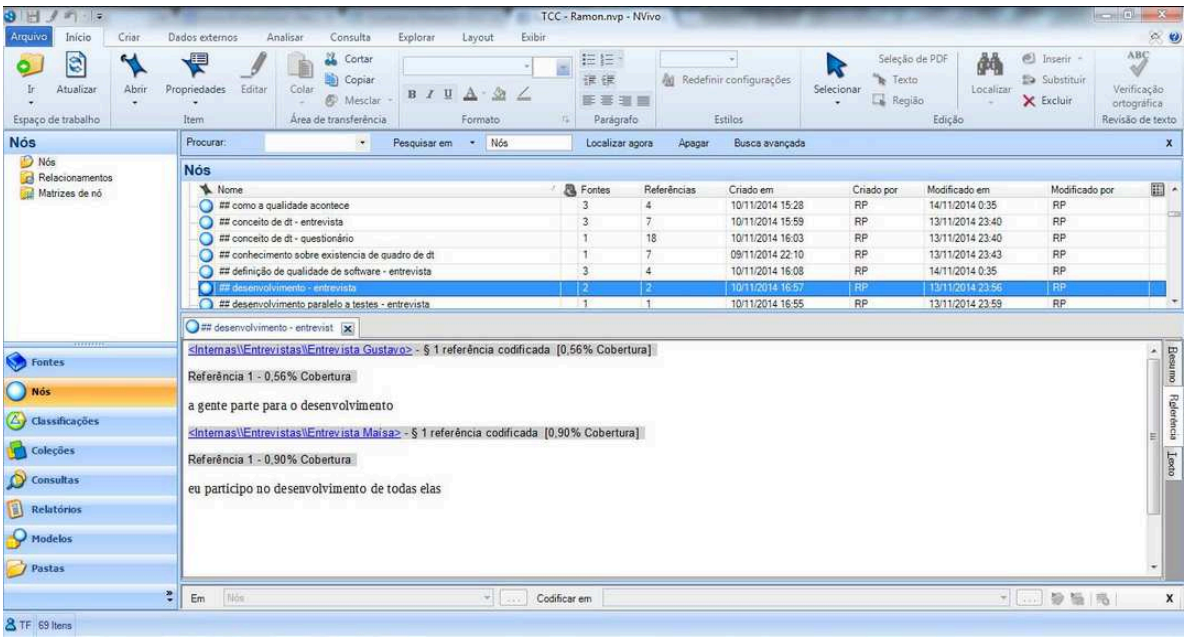

Figura 4.7: Utilização da ferramenta *NVivo 10* pela presente pesquisa

Fez-se necessário a transcrição de todos os dados em forma de texto. As observações e respostas de questionários ou entrevistas foram incluídas num documento de texto que posteri-

ormente foi carregado na ferramenta NVivo 10 para que a análise pudesse ser realizada em três fases de codificações que são abordadas nos resultados dessa pesquisa.

5 RESULTADOS

Neste capítulo serão apresentados os resultados da presente pesquisa.

5.1 Codificação aberta

O método de codificação consiste em extrair fragmentos dos dados transcritos (informações-chave) e codificá-los em *strings* que representem aquele determinado trecho. Normalmente é utilizada uma ferramenta para a codificação e na presente pesquisa, utilizou-se o programa *NVivo* para a análise dos dados de forma qualitativos.

A Tabela 5.1 apresenta os nós mais frequentes agrupados nas categorias, que posteriormente serão agrupados em coleções.

Coleção	**Ocorrência de nós**
Identificação de Dívida Técnica	20
Monitoramento de Dívida Técnica	14
Requisitos	11
Desenvolvimento-Teste	7
Qualidade de Software	5
Homologação	2

Tabela 5.1: Nós com maior ocorrência

5.2 Codificação axial

A partir do momento em que todos os dados coletados foram transformados em nós, iniciou-se para a segunda fase de codificação. A codificação axial consiste no agrupamento dos nós referentes ao mesmo assunto em grupos chamados de coleção. Para que esses nós sejam agrupados em uma mesma coleção, esses devem interagir entre si. A Figura 5.1 apresenta o resultado da codificação axial.

A Figura 5.1 representa as coleções de nós e como elas se relacionam. É possível que um nó, caso possua relação com outra coleção, seja replicado para esta, fazendo parte então de duas coleções simultaneamente. As relações entre as coleções mostradas na Figura 5.1 são representações de como elas estão interligadas.

A coleção Requisitos se relaciona com a coleção Desenvolvimento-Teste, pois toda coleta e análise de requisitos é executada com qualidade para que não haja dúvida na hora do

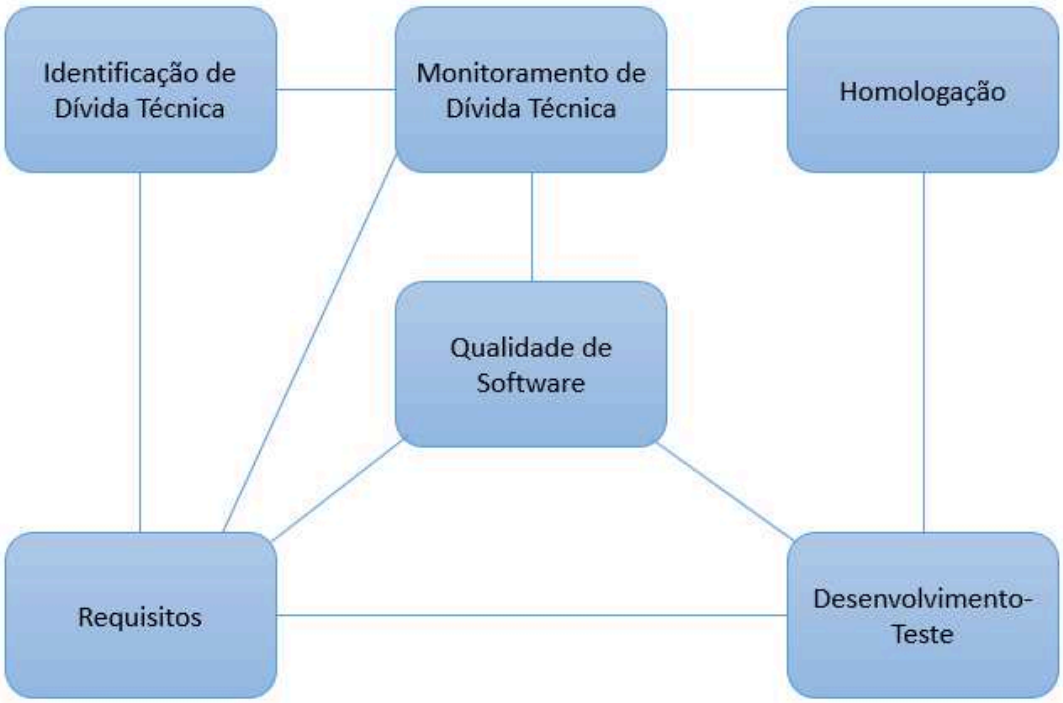

Figura 5.1: Resultado da codificação axial sobre os primeiros dados - Coleções e suas relações

desenvolvimento. Além disso, todo requisito deve seguir normas especificadas pela área de qualidade de software, para que não haja inconsistências e divergências sobre o objetivo do cliente em relação ao produto final.

Toda fase de coleta, análise e execução dos requisitos ainda se associa a Identificação e ao Monitoramento da dívida técnica, visto que essas são fases importantes na produção do software e costumam contrair uma significativa quantidade de dívida para o projeto.

A coleção Identificação de Dívida Técnica, além de possuir relação com a coleção Requisitos, relaciona-se com a coleção Monitoramento da Dívida Técnica, tendo em vista que, toda dívida que é identificada dentro do processo de produção deve ser monitorada posteriormente para que se saiba o seu estado atual, além da probabilidade de futuros problemas para o projeto.

A coleção Monitoramento de Dívida Técnica, afora as relações já citadas com as coleções Requisitos e Identificação de Dívida Técnica, possui relação com a coleção Qualidade de Software e com a Homologação. Para que se obtenha a qualidade não só no produto final, mas também em todo o processo de produção, a equipe precisa que o monitoramento de sua dívida técnica funcione. Com a garantia da qualidade durante o processo proporcionada pelo monitoramento da dívida, o produto irá para homologação com um número reduzido de erros.

A coleção Desenvolvimento-Teste tem relações com a coleção Requisitos (já citada), com a coleção Qualidade de Software e com a coleção Homologação. Toda fase de desenvolvimento que ocorra em paralelo com a fase de testes deve ocorrer seguindo o padrão máximo de qualidade para que se contraia o mínimo possível de dívidas técnicas. A relação da coleção Desenvolvimento-Teste com a coleção Homologação se justifica pelo fato de que, uma vez que o sistema sai da fase de desenvolvimento e parte para ser homologado, ele ainda pode retornar à fase de desenvolvimento para a correção de eventuais erros encontrados pelo cliente quando o sistema estava sendo testado pelos requisitantes em funções executadas no dia a dia.

5.3 Codificação seletiva

A terceira fase de codificação, a seletiva, tem o objetivo de integrar todas as coleções numa categoria central. Neste caso o objetivo não é gerar uma teoria, mas identificar as principais coleções da percepção do setor de TI da UFFS no modelo de identificação e monitoramento da dívida técnica aplicados. O resultado portanto permanece sendo o da Figura 5.1 após essa fase de codificação.

Apesar disso, é percebível que todas as coleções são direta ou indiretamente impactadas pela qualidade de software e por isso pode-se considerá-las membros de uma coleção chave Qualidade de Software.

5.4 Modelo de identificação e monitoramento

A análise dos dados qualitativos das entrevistas e questionários ao longo da pesquisa ainda resultou no modelo final de identificação e monitoramento de dívida técnica do setor. Para que o modelo pudesse ser produzido e posteriormente executado, constatou-se que a grande maioria dos membros compreenderam o conceito de dívida técnica que foi sugerido e evoluído pelos participantes do treinamento. Como já citado na revisão bibliográfica, todo artefato imaturo encontrado no processo de desenvolvimento de software do setor seria considerado dívida técnica.

O modelo ilustra o acontecimento do treinamento sobre dívida aos funcionários e como funciona a identificação e monitoramento desta no caso estudado. É possível observar o modelo na Figura 5.2.

Quando a presente pesquisa foi proposta, já era notório que a gestão de dívida téc-

Figura 5.2: Modelo resultante da análise de dados representando a gestão da dívida técnica

nica nos casos que fossem estudados seria dividida entre a identificação e o monitoramento, excluindo-se o dimensionamento, tendo em vista que esta fase está fora do escopo do presente trabalho.

A identificação tem o seu suporte na inclusão das dívidas no quadro e no *Redmine*. Toda dívida técnica que é encontrada pelos funcionários do setor deve ser cadastrada em um desses locais. Caso o quadro seja o local de cadastro, a dívida deve ser replicada posteriormente na ferramenta para que se tenha o histórico das dívidas presentes nos projetos do setor. A partir disso, três itens são gerados: As dívidas relevantes, as dívidas recentes e as estatísticas sobre as dívidas.

As dívidas relevantes existem para retratar dívidas que devem ser pagas na *sprint* e que foram diagnosticadas, na hora da identificação, com uma alta probabilidade de gerar problemas. Essas dívidas merecem uma atenção especial, pois podem prejudicar um projeto inteiro, com custos elevados e emprego de demasiado esforço para o pagamento. Logo, as mesmas devem ser discutidas na reunião de planejamento da *sprint* visando analisar se é o caso de pagá-las ou não.

Para as dívidas recentes, os funcionários receberam orientações para que essas, além de mantidas no *Redmine*, fossem incluídas no quadro. Essa orientação surgiu em uma resposta recebida no questionário de acompanhamento, tendo em vista que esta prática proporcionaria facilidade de acesso e visibilidade das dividas existentes nos projetos a todos os membros do setor. Desta forma, quando os membros do projeto chegam no setor e olham para o quadro já se tem uma visão de como está a saúde do software. Além disso, quando o quadro começa

a ficar saturado de dívidas, pode haver um certo desconforto para a equipe, pois todos podem comprovar que a qualidade do código pode estar comprometida.

As estatísticas sobre as dívidas consta nesse modelo para explicar a situação das destas até o momento. Os dados variam desde a quantidade de dívidas cadastradas por um usuário até o valor total de horas para o pagamento de todas as dívidas do setor. É fator importante para que se possa quantificar certos aspectos acerca da dívida técnica no setor.

No modelo, o monitoramento da dívida funciona como um pequeno ciclo, onde apenas o primeiro item não é executado novamente. Para que o monitoramento da dívida ocorra de forma eficiente, criou-se uma cultura de conversa sobre as dívidas dentro do setor. Ao passo que os membros do setor discutem sobre as dívidas, as decisões tomadas sobre o seu pagamento e sua influência terão um maior grau de acerto.

O monitoramento continua com as definições de quais dívidas serão pagas na *sprint* e quanto tempo será alocado para o pagamento dessas dívidas. Dentro dessas discussões envolve-se também o fato de que existem dívidas que podem ser consideradas sob controle e que não crescem seu valor de pagamento há um tempo considerável. Com isso, o grupo pode considerar essa dívida como estável e não prejudicial para o desempenho do projeto.

Após isso, o pagamento das dívidas, último item do pequeno ciclo de monitoramento ocorre, seguindo as diretrizes definidas nas reuniões de planejamento da *sprint*.

Com esses dois modelos definidos, buscou-se analisar os dados das últimas entrevistas aplicadas, onde tinha-se como objetivo entender o motivo das dívidas serem contraídas. Para tanto, codificou-se as entrevistas que estavam em texto para nós no intuito de facilitar o entendimento e seguir os passos do GT. Com todo o texto codificado, fez-se para o agrupamento de nós em coleções e com a análise destas chegou-se a dois fatores principais que contribuíam para que essas dívidas fossem contraídas. A Figura 5.3 apresenta o resultado da codificação axial no processo de procura das causas para a contração das dívidas.

O primeiro fator encontrado foi a falta de tempo. Os membros alegaram existir uma certa pressão para que os sistemas sejam entregues em tempo e que por vezes a responsabilidade de datas acaba por afetar a eficiência do trabalho e consequentemente do sistema.

O segundo fator resultante da análise dos dados, que foi taxado como grave no quesito contração de dívidas, foi a falta de pessoas. Os membros entrevistados entendem que com um maior número de funcionários no setor, as tarefas poderiam ser melhor divididas e erros que frequentemente acontecem poderiam ser reduzidos.

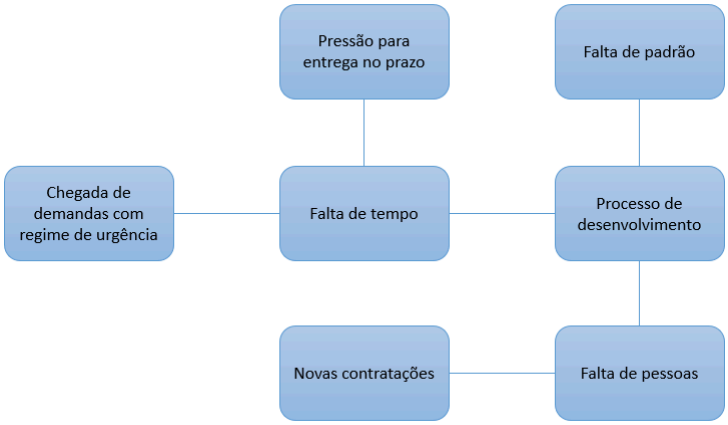

Figura 5.3: Resultado da codificação axial sobre as últimas entrevistas - Coleções e suas relações

Buscou-se ainda compreender se havia algum modo de evitar que essas fossem contraídas. Identificou-se que a contratação de novos funcionários e a instituição de um padrão de trabalho para que processo de desenvolvimento seja melhor definido foram os fatores identificados como possíveis ações necessárias para evitar a contração das dívidas.

O modelo resultante foi validado com um membro do setor com um questionário sugerindo que fossem apontados os pontos positivos e negativos do mesmo.

Com a análise dos dados, ficou notório que a experiência foi bastante proveitosa do ponto de vista de identificar, documentar e antecipar futuros problemas. Além disso, a conscientização a respeito dos problemas deixados para trás e que podem gerar consequências foi outro ponto importante perceptível nas respostas. Um membro da equipe argumentou que "de certa forma, isso contribui para que as pessoas minimizem as dívidas técnicas, aumentando a qualidade do trabalho realizado".

No entanto, foi possível perceber após a análise das entrevistas que para tirar melhor proveito do modelo, é preciso ter um processo de gestão e desenvolvimento menos flexível para que o modelo possa ser aplicado em sua totalidade, uma vez que não é possível identificar no setor o responsável pela atualização das informações ou mesmo pela cobrança do pagamento das dívidas e isso pode levar o processo ao ostracismo. Sugeriu-se ainda que apesar do fato que as dívidas devem ser identificadas e monitoradas por todos no setor, alguém poderia ficar responsável por buscar mais informações sobre a metáfora, fazendo uso desse conhecimento para orientar os outors membros em caso de dúvidas ou novas ações a ser tomadas para evolução do modelo.

O modelo final de identificação e monitoramento de dívida técnica encontrado nessa

pesquisa é de fácil entendimento e pode ser aplicado em qualquer equipe de desenvolvimento de software, sem grandes alterações no seu processo e pelo menos parte dele pode ser adotado sem a adição de novas ferramentas, apenas com a adição do quadro de cadastro de dívidas. O quadro auxilia a criar uma cultura de qualidade na produção do software, uma vez que todos podem ter acesso ao mesmo com facilidade. Além disso, toda vez que um membro do time olhar para o quadro imediatamente consegue visualizar a saúde do projeto.

6 TRABALHOS RELACIONADOS E DISCUSSÃO

Algumas pesquisas já buscaram fazer a identificação e o monitoramento de dívida técnica. Este capítulo destina-se aos trabalhos relacionados com a presente pesquisa, focado no quesito da identificação e monitoramento da dívida.

O estudo elaborado por Zazworka et al. [54] também utiliza uma lista de itens de dívida técnica para monitorá-las, mas na fase de identificação vários métodos são testados, incluindo questionários como abordagens humanas e o resultado da análise dos dados realizado encontrados com o *CodeVizard* e o *FindBugs* como abordagens computacionais. As ferramentas buscavam por *code smells* (potenciais violações de bons princípios na orientação a objeto) e *issues* (violações de práticas recomendadas na programação e potenciais defeitos que possam causar problemas no futuro).

Autores como Carolyn Seaman [43], Yuepu Guo [22] e Martin Fowler [16] também já utilizaram ferramentas que reconhecem *code smells* no código para identificar dívida técnica. Adicionalmente, um estudo que utilizou ferramentas que reconhecem *issues* para identificação da dívida foi a pesquisa de Antonio Vetro [50].

Apesar da abordagem humana do trabalho de Zazworka et al. [54] se assemelhar com a presente pesquisa, a abordagem computacional do trabalho é utilizada para identificar as dívidas no código-fonte diferentemente da presente pesquisa que utiliza do *Redmine* para o cadastro das dívidas identificadas pelos membros após o treinamento para a compreensão da equipe sobre o que é dívida técnica.

Outro fator que assemelha o trabalho de Zazworka et al. [54] é o padrão utilizado para o cadastro da dívida. Uma vez que o padrão usado na presente pesquisa foi baseado no modelo apresentado no trabalho de Zazworka et al., grande parte dos campos do padrão da presente pesquisa é igual aos campos do modelo de Zazworka et al. O modelo utilizado por ele é representado na Figura 6.1 e pode ser comparado com o padrão utilizado na presente pesquisa que é ilustrado na Figura 4.3

No trabalho realizado por Wiklund et al. [51] foi efetuado um estudo de caso buscando investigar a dívida técnica em testes automatizados. Foram procurados fatores que contribuíam para a ineficiência na utilização, manutenção e desenvolvimento dos testes. A pesquisa utilizou-se de entrevistas semiestruturadas combinados com a inspeção de uma série de documentos (regras de *design* para os *scripts* de teste, diretrizes de qualidade, entre outros) para identificar

ID	*TD identification number*
Responsible	*Person or role who should fix this TD item*
Type	*design, documentation, defect, testing, or other type of debt*
Location	*List of files/classes/methods or documents/pages involved*
Description	*Describes the anomaly and possible impacts on future maintenance*
Estimated principal	*How much work is required to pay off this TD item on a three point scale:* ***High/Medium/Low***
Estimated interest amount	*How much extra work will need to be performed in the future if this TD item is not paid off now on a three point scale:* ***High/Medium/Low***
Estimated interest probability	*How likely is it that this item, if not paid off, will cause extra work to be necessary in the future on a three point scale:* ***High/Medium/Low***
Intentional?	***Yes/No/Don't Know***

Figura 6.1: *Template* para cadastro de dívidas [54]

e estimar a dívida. Após o estudo, foi concluído que a área de testes automatizados não possuía orientações evidentes e detalhadas sobre como projetar e executar para manter um teste automatizado em um padrão de dívida técnica aceitável. No entanto, a pesquisa ainda conclui que essa é uma demanda grande e precisa ser explorada no futuro.

A pesquisa realizada por Codabux e Williams [7] consistiu em um estudo feito durante a implementação de práticas de desenvolvimento ágil em uma indústria com o objetivo de entender como a dívida técnica é caracterizada e adquirida pelas empresas, além do modo como cada fator influenciava a tomada de decisões. A coleta de dados do trabalho também foi feita através de entrevistas semiestruturadas, observação etnográfica e questionários. Para a identificação da dívida técnica, a pesquisa determinou que os desenvolvedores considerariam sua própria taxonomia de dívida técnica com base no tipo de trabalho que lhes eram atribuídos e sua compreensão pessoal do termo. Seguindo a linha de raciocínio da forma de identificação da dívida, o estudo conclui que os participantes da entrevista criaram com várias definições e categorias de dívida técnica com base no tipo de trabalho que eles são responsáveis.

A pesquisa de Codabux e Williams [7] além disso ultrapassa o escopo da presente pes-

quisa pois dimensiona a dívida técnica no projeto. Um item de dívida técnica para a pesquisa representa uma tarefa de manutenção que foi prorrogada e pode gerar problemas no futuro. Cada item de dívida técnica possui a informação sobre o que cada dívida é, por que e quando foi assumida, além do local e quem foi o responsável pela mesma. Um item de dívida técnica também possui o CDT estimado e os juros da dívida. O CDT e os juros são estimados com a principal medida sendo o esforço. Após isso, a dívida técnica é estimada, quantificada e está pronta para o uso no processo de tomada de decisão [21]. Na fase de coleta de dados da pesquisa foi utilizada a ferramenta *Unified Code Count* para identificar o tamanho do sistema antes e depois de cada ponto de cronograma de evolução. A métrica utilizada foi o número de linhas de código [21].

O trabalho de Siebra et al. [47] teve o objetivo de analisar um projeto de uma indústria na perspectiva das suas decisões e eventos relacionados para que a existência de dívida técnica no projeto fosse melhor evidenciada e a evolução de seus parâmetros melhor acompanhada. Os dados foram coletados para análise através de e-mails, documentos, *logs* CVS, arquivos de código e entrevistas. Concluiu-se que, de acordo com os cenários analisados na pesquisa, o uso de decisões surge como uma abordagem significativa para caracterizar a evolução da dívida técnica. De acordo com o autor, quando busca-se identificar dívida técnica, existem outros fatores, além de aspectos de implementação, que devem ser analisados, como por exemplo, as relações humanas da equipe [47].

Uma pesquisa elaborada por Guo et al. [21] explora o efeito da dívida técnica em uma tarefa de manutenção atrasada em um projeto de software real e simula como a gestão da dívida pode alterar seus resultados durante o ciclo de vida do projeto. Ao rastrear essa dívida e medir o impacto dela no projeto, o trabalho em questão demonstra evidências e números concretos sobre o quão sério é contrair uma dívida com um CDT alto. Além disso, é possível verificar o quanto uma decisão sem uma análise prévia e cuidadosa pode ser prejudicial e agravar o efeito da dívida. Para monitorar a dívida técnica da equipe um quadro é proposto, assim como na presente pesquisa. O quadro visa monitorar uma lista de dívidas técnicas do projeto em que a pesquisa é aplicada. A lista contém todos os itens de dívidas atuais do sistema, diferentemente do quadro colocado no setor de TI da UFFS onde somente as dívidas mais recentes e relevantes são mantidas.

A Tabela 6.1 mostra um breve resumo de comparações entre a presente pesquisa e os trabalhos citados.

	Trabalho Zazworka	**Trabalho Wiklund**	**Trabalho Codabux**	**Trabalho Siebra**	**Trabalho Guo**
Estudo de Caso	Sim	Sim	Sim	Sim	Sim
Identificação	Sim	Sim	Sim	Sim	Não
Monitoramento	Sim	Não	Sim	Sim	Sim
Dimensionamento	Não	Não	Sim	Não	Não
Padrão de cadastro	Sim	Não	Não	Não	Sim
Quadro de dívidas	Não	Não	Não	Não	Sim
Treinamento para equipe	Não	Não	Não	Não	Não
Análise de código-fonte	Sim	Não	Sim	Não	Não
Encontrar causas da dívida	Não	Sim	Sim	Sim	Sim

Tabela 6.1: Comparação dos trabalhos existentes com a presente pesquisa

Estas pesquisas sugerem a utilização de abordagens humanas, através de entrevistas, questionários e quadros, e abordagens computacionais, através de ferramentas, para a identificação, dimensionamento e monitoramento de dívida técnica. A definição de quais abordagens serão utilizadas, porém, varia de caso para caso, dependendo do ambiente em que o estudo é aplicado. Adicionalmente, em cada estudo ferramentas e formas diferentes de identificar e monitorar dívida técnica são utilizados.

Nesta pesquisa foi proposto um modelo de gestão de dívida técnica que foi evoluido ao longo de todo o processo, como por exemplo com a automatização da forma de cadastro e monitoramento das dívidas, replicando as dívidas do quadro cadastral para uma ferramenta automatizada. Posteriormente, verificou-se a necessidade da introdução de uma ferramenta no contexto do setor para que se pudesse manter o histórico das dívidas, além de automatizar o processo de inserção de novas dívidas no cadastro. A presente pesquisa diferencia-se das outras, pois sugere um modelo completo e simples de gestão com sugestão de ferramentas, quadros e formas de gestão de dívida técnica no processo de desenvolvimento de software de um caso real.

7 CONCLUSÃO

O trabalho apresentou significantes resultados em resposta as questões impostas para a pesquisa, mesmo com alguns empecilhos durante a aplicação do estudo de caso, como a troca do diretor geral do setor durante a análise de viabilidade deste como um caso do estudo. Com a aplicação do *Grounded Theory* em todas as suas fases para a análise de dados qualitativos, ficou mais fácil compreender as áreas mais relevantes para a identificação e o monitoramento da dívida técnica.

Apesar de recente, a metáfora da dívida técnica vem ganhando significativa força na comunidade de desenvolvimento ágil como uma maneira de compreender e medir a qualidade do processo de desenvolvimento de software. Este trabalho propôs um modelo para gestão da dívida técnica para uma parte da Secretaria Especial de Tecnologia e Informação da Universidade Federal da Fronteira Sul.

É importante salientar que esse modelo não pode ser generalizado uma vez que este não foi aplicado em múltiplos casos. No entanto, o modelo possui simplicidade em seu processo e pode ser replicado em outros casos, para continuar sendo testado, evoluído e então poder ser generalizado.

O modelo não visa alterar o processo da equipe e procura ajudar na criação de uma cultura voltada a qualidade no desenvolvimento de software. Contatou-se ainda que o modelo pode ser empregado independente da tecnologia ou equipe, podendo ser utilizado desde o membro júnior até o membro sênior.

7.1 Trabalhos futuros

A partir da presente pesquisa, abre-se um leque de oportunidades para pesquisas futuras no setor de TI da UFFS. O código-fonte do setor, por exemplo, não foi analisado e pode conter dívidas que não aparecem na identificação realizada pelo presente trabalho. Ao passo que a presente pesquisa somente buscou identificar e monitorar as dívidas, sugere-se que essas dívidas poderiam ser dimensionadas, com o método SQALE por exemplo, para facilitar a tomada de decisão, principalmente no momento do pagamento das dívidas. Além disso, esta pesquisa pode ser acompanhada por mais tempo no futuro, com a intenção de avaliar os pontos positivos e negativos a longo prazo, tais como evoluções das dívidas ou ineficiência parcial do modelo.

REFERÊNCIAS

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

[2] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. Ozkaya, et al. Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 47–52. ACM, 2010.

[3] F. Buschmann. To pay or not to pay technical debt. *Software, IEEE*, 28(6):29–31, 2011.

[4] J. P. Cavano and J. A. McCall. A framework for the measurement of software quality. *ACM SIGSOFT Software Engineering Notes*, 3(5):133–139, 1978.

[5] K. Charmaz. *A construção da teoria fundamentada: guia prático para análise qualitativa*. Bookman, 2009.

[6] C. Climate. Code climate, a better experience for creating software. https://codeclimate.com/about, 2011. [Online; Acesso em: 02/07/2014].

[7] Z. Codabux and B. Williams. Managing technical debt: An industrial case study. In *Managing Technical Debt (MTD), 2013 4th International Workshop on*, pages 8–15. IEEE, 2013.

[8] W. Cunningham. The wycash portfolio management system. http://c2.com/doc/oopsla92.html, 1992. [Online; Acesso em: 11/06/2014].

[9] W. Cunningham. Ward explains debt metaphor. http://c2.com/cgi/wiki?WardExplainsDebtMetaphor, 2009. [Online; Acesso em: 11/06/2014].

[10] B. Curtis, J. Sappidi, and A. Szynkarski. Estimating the principal of an application's technical debt. *IEEE software*, 29(6):0034–42, 2012.

[11] B. Curtis, J. Sappidi, and A. Szynkarski. Estimating the size, cost, and types of technical debt. In *Managing Technical Debt (MTD), 2012 Third International Workshop on*, pages 49–53. IEEE, 2012.

[12] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer, 2008.

[13] R. J. Eisenberg. A threshold based approach to technical debt. *ACM SIGSOFT Software Engineering Notes*, 37(2):1–6, 2012.

[14] D. Falessi, P. Kruchten, R. L. Nord, and I. Ozkaya. Technical debt at the crossroads of research and practice: report on the fifth international workshop on managing technical debt. *ACM SIGSOFT Software Engineering Notes*, 39(2):31–33, 2014.

[15] D. Falessi, M. A. Shaw, F. Shull, K. Mullen, and M. S. Keymind. Practical considerations, challenges, and requirements of tool-support for managing technical debt. In *Managing Technical Debt (MTD), 2013 4th International Workshop on*, pages 16–19. IEEE, 2013.

[16] M. Fowler. *Refactoring: improving the design of existing code*. Pearson Education India, 1999.

[17] M. Fowler. Technicaldebtquadrant. http://www.martinfowler.com/bliki/TechnicalDebtQuadrant.html, 2009. [Online; Acesso em: 11/06/2014].

[18] Gartner. Gartner estimates global 'it debt' to be $500 billion this year, with potential to grow to $1 trillion by 2015. http://www.gartner.com/newsroom/id/1439513, 2010. [Online; Acesso em: 09/06/2014].

[19] I. Gat and J. D. Heintz. From assessment to reduction: how cutter consortium helps rein in millions of dollars in technical debt. In *Proceedings of the 2nd Workshop on Managing Technical Debt*, pages 24–26. ACM, 2011.

[20] J. Grenning. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*, 3, 2002.

[21] Y. Guo, C. Seaman, R. Gomes, A. Cavalcanti, G. Tonin, F. Q. Da Silva, A. L. M. Santos, and C. Siebra. Tracking technical debt—an exploratory case study. In *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*, pages 528–531. IEEE, 2011.

[22] Y. Guo, C. Seaman, N. Zazworka, and F. Shull. Domain-specific tailoring of code smells: an empirical study. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 167–170. ACM, 2010.

[23] N. C. Haugen. An empirical study of using planning poker for user story estimation. In *Agile Conference, 2006*, pages 9–pp. IEEE, 2006.

[24] IBM. Findbugs, part 1: Improve the quality of your code. http://www.ibm.com/developerworks/java/library/j-findbug1/, 2004. [Online; Acesso em: 29/06/2014].

[25] B. A. Kitchenham and S. L. Pfleeger. Personal opinion surveys. In *Guide to Advanced Empirical Software Engineering*, pages 63–92. Springer, 2008.

[26] P. Kruchten, R. L. Nord, and I. Ozkaya. Technical debt: from metaphor to theory and practice. *IEEE Software*, 29(6):18–21, 2012.

[27] E. M. Lakatos and M. de Andrade Marconi. *Fundamentos da Metodologia científica*. Atlas São Paulo, 2003.

[28] E. M. Lakatos and M. de Andrade Marconi. *Metodologia científica*. Atlas São Paulo, 2006.

[29] J. Letouzey. Managing technical debt with sqale method. *Software, IEEE*, 29(6):44–51, 2012.

[30] J.-L. Letouzey. The sqale method for evaluating technical debt. *MTD@ ICSE*, 2012.

[31] J. Li, T. Stålhane, J. M. Kristiansen, and R. Conradi. Cost drivers of software corrective maintenance: An empirical study in two companies. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–8. IEEE, 2010.

[32] E. Lim, N. Taksande, and C. Seaman. A balancing act: what software practitioners have to say about technical debt. *Software, IEEE*, 29(6):22–27, 2012.

[33] S. Mcconnell. Technical debt. http://www.construx.com/10x_Software_Development/Technical_Debt/, 2007. [Online; Acesso em: 06/06/2014].

[34] K. Molokken-Ostvold and N. C. Haugen. Combining estimates with planning poker–an empirical study. In *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian*, pages 349–358. IEEE, 2007.

[35] S. NVivo. nvivo - features and benefits. http://www.qsrinternational.com/products_nvivo_features-and-benefits.aspx, 2012. [Online; Acesso em: 28/11/2014].

[36] I. Ozkaya, P. Kruchten, R. L. Nord, and N. Brown. Managing technical debt in software development: report on the 2nd international workshop on managing technical debt, held at icse 2011. *ACM SIGSOFT Software Engineering Notes*, 36(5):33–35, 2011.

[37] N. R. Pandit. The creation of theory: A recent application of the grounded theory method. *The qualitative report*, 2(4):1–14, 1996.

[38] M. Petrini and M. Pozzebon. Usando grounded theory na construção de modelos teóricos. *Gestão & Planejamento-G&P*, 10(1), 2009.

[39] C. C. Prodanov and E. C. d. Freitas. Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico. *Novo Hamburgo: Feevale*, 2013.

[40] S. RedMine. Redmine. http://redmine.org, 2006. [Online; Acesso em: 28/11/2014].

[41] F. V. Rudio. *Introdução ao projeto de pesquisa científica*. Vozes, 1979.

[42] K. Schwaber and J. Sutherland. The scrum guide. *Scrum. org, October*, 2011.

[43] C. Seaman and Y. Guo. Measuring and monitoring technical debt. *Advances in Computers*, 82:25–46, 2011.

[44] C. Seaman, Y. Guo, C. Izurieta, Y. Cai, N. Zazworka, F. Shull, and A. Vetrò. Using technical debt data in decision making: Potential decision approaches. In *Managing Technical Debt (MTD), 2012 Third International Workshop on*, pages 45–48. IEEE, 2012.

[45] Seti. Organograma da seti. http://uffs.edu.br/index.php?option=com_content&view=article&id=1313&Itemid=1217&site=seti, 2010. [Online; Acesso em: 21/11/2014].

[46] T. Sharma. Quantifying quality of software design to measure the impact of refactoring. In *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, pages 266–271. IEEE, 2012.

[47] C. S. A. Siebra, G. S. Tonin, F. Q. Silva, R. G. Oliveira, A. L. Junior, R. C. Miranda, and A. L. Santos. Managing technical debt in practice: An industrial report. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 247–250. ACM, 2012.

[48] S. Source. Technical debt evaluation. http://www.sonarsource.com/products/features/technical-debt-evaluation/, 2008. [Online; Acesso em: 01/07/2014].

[49] A. L. Strauss and J. Corbin. *Pesquisa qualitativa: técnicas e procedimentos para o desenvolvimento de teoria fundamentada*. Artmed, 2008.

[50] A. Vetro. Using automatic static analysis to identify technical debt. In *Proceedings of the 34th International Conference on Software Engineering*, pages 1613–1615. IEEE Press, 2012.

[51] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist. Technical debt in test automation. In *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*, pages 887–892. IEEE, 2012.

[52] R. K. Yin. *Case study research: Design and methods*. Sage publications, 2009.

[53] N. Zazworka, M. A. Shaw, F. Shull, and C. Seaman. Investigating the impact of design debt on software quality. In *Proceedings of the 2nd Workshop on Managing Technical Debt*, pages 17–23. ACM, 2011.

[54] N. Zazworka, R. O. Spínola, A. Vetro, F. Shull, and C. Seaman. A case study on effectively identifying technical debt. In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, pages 42–47. ACM, 2013.

APÊNDICES

Perguntas das entrevistas realizadas com 3 membros do setor onde objetivou-se identificar métricas de qualidade já aplicadas no local, além de verificar a compreensão da metáfora dívida técnica pela equipe.

Sua função.

Como tu definiria “qualidade de software”.

Vocês trabalham com o conceito de qualidade de software aqui no setor?

Como isso acontece?

O que você entende por dívida técnica?

Você acha importante monitorar dívida técnica dentro do processo de desenvolvimento de software?

Por que?

Você acredita que o setor contraia dívida técnica?

Dessas dívidas, tem alguma que tu considera a mais crítica? Que se fosse pra monitorar a partir de hoje, qual seria?

Como você definiria o processo, desde o surgimento da demanda, até a entrega do produto final para o cliente.

Numa escala de 1 a 5, sendo 1 pouquíssima importância e 5 muitíssima importância, como a equipe trata cada um dos intens abaixo.

- Levantamento de requisitos
- Análise de requisitos
- Documentação
- Codificação
- Testes
- Recodificação
- Gestão do projeto

- Implantação do sistema
- Interação com o requisitante
- Interação com o usuário

Vocês utilizam alguma ferramenta para o controle de qualidade de software?

Existe alguma métrica pré definida para o monitoramento?

Se sim, quem é o responsável?

Perguntas do questionário realizado com o setor onde objetivou-se realizar um acompanhamento do cadastro de dívidas. Após esse questionário, percebeu-se a necessidade da ferramenta.

Função no setor.

Defina dívida técnica.

Você tem conhecimento que o setor possui um local para o cadastro de dívidas técnicas?

Você já cadastrou alguma dívida técnica?

Se não, por que?

Caso um dia se precise dos históricos dessas dívidas, como você gostaria que essas dívidas técnicas fossem armazenadas? Uma ferramenta poderia ser uma solução?

Você acredita que apenas as dívidas mais recentes poderiam ser mantidas no quadro?

Perguntas da entrevista realizada no setor onde objetivou-se compreender as causas da dívida contraída e buscar alternativas para que estas pudessem ser evitadas.

A falta de tempo pode ser um dos motivos que contribuíram para que as dívidas do setor sejam contraídas? Por que?

A chegada de demandas específicas que tem prioridade sobre as outras podem contribuir para a contração da dívida? Por que?

Além disso, podem existir outros motivos para a contração da dívida?

Na sua opinião, a dívida poderia ser evitada de alguma maneira? Como?

O contrainte da dívida está a disposição para retirada de dúvidas sobre a dívida?

Questão aplicada no setor para validação da pesquisa

Analisando a pesquisa em sua totalidade, gostaria que você citasse os pontos positivos e negativos do trabalho.