MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CENTRO MULTIDISCIPLINAR DE PAU DOS FERROS

CONVOCAÇÃO

O Presidente em exercício do **CONSELHO DO CENTRO MULTIDISCIPLINAR DE PAU DOS FERROS** convoca todos os conselheiros a se fazerem presentes à **5ª Reunião Extraordinária de 2021**, com data e horário abaixo discriminados, para cumprir a seguinte pauta:

1. Indicação de servidores para compor comissão de elaboração e implementação de edital de espaço físico com destinação a alocar projetos/ações de extensão e/ou pesquisa;

2. Apreciação e deliberação sobre renovação de afastamento conforme processo 23091,001267/2018-34;

3. Apreciação e deliberação acerca da destinação dos códigos de vaga docente conforme descrito no MEMORANDO ELETRÔNICO Nº 133/2021 - GR (11.03).

Data: 30 de abril de 2021 (sexta-feira)
Horário: 8h30min
Local: remoto

Pau dos Ferros-RN, 27 de abril de 2021.

Ricardo Paulo Fonseca Melo
Presidente

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO - UFERSA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO - PROPPG
Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

REQUERIMENTO E ANEXOS PARA RENOVAÇÃO DE AFASTAMENTOS DE SERVIDORES DOCENTES DA UFERSA PARA QUALIFICAÇÃO EM INSTITUIÇÕES NACIONAIS OU ESTRANGEIRAS EM NÍVEL DE PÓS-GRADUAÇÃO *STRICTO SENSU*

1. PREENCHIDO PELO REQUERENTE

Nome (completo sem abreviaturas): JOSÉ FERDINANDY SILVA CHAGAS
Identidade: 2003030026003 **Órgão Emissor:** SSP **UF:** CE **Data de emissão:** 21/05/2003
CPF: 014.863.283-18 **Data de Nascimento:** 24/02/1986 **Tel.:** (84) 9 9814-8612
E-mail: ferdinandy@ufersa.edu.br **Departamento/Setor:** Departamento de Engenharias e Tecnologia / Centro Multidisciplinar de Pau dos Ferros (CMPF)
Tipo de Afastamento: Integral: (X) **Parcial:** ( )
Tempo de Serviço Averbado para Aposentadoria: (0) **Anos**
Início de Exercício no Cargo: 08/04/2014 **Total: 6 ano(s) 11 mês(es)** (Anexar Declaração do PRORH).

2. PREENCHIDO PELO REQUERENTE
CURSO: Pós-graduação em Ciência da Computação
Nível: ( ) **Mestrado** ( ) **Doutorado** (X)
Área de concentração: Engenharia de Software
Liberação inicial: Início 21/05/2018 **Término:** 20/05/2019
Período solicitado para (renovação): Início 21/05/2021 **Término:** 01/03/2022
Previsão para término do curso: Início Término: 01/03/2022

ANEXAR (Obrigatório)

I. Lista de verificação própria disponibilizada pela PROPPG (**Check-List**); ***(Anexo I)***

II – Justificativa de seu requerimento; ***(Anexo II)***

III- Relatório de atividades acadêmicas (Anexo III) (quando se tratar do relatório referente ao 3º semestre (mestrado) e 5º semestre (doutorado), deverá ser acompanhado do **projeto de dissertação/Tese)**

IV- Relatório de avaliação de desempenho, feito pelo/a orientador/a (Anexo IV)

V - Declaração de matrícula (Local da pós-graduação) (Anexo V)

VI- Histórico Escolar (Anexo VII) (Disponível na Página da PROPPG**)**

VII- Termo de Compromisso dos docentes que assumirão os componentes curriculares do docente afastado, durante o período de renovação do afastamento, restrito aos casos de indisponibilidade de vaga para contratação de professor substituto; ***(Anexo VII)***

VIII – Termo de Compromisso, devidamente preenchido e assinado com testemunhas; ***(Anexo VIII)***

IX - Parecer da chefia imediata (Departamento acadêmico de lotação do requerente); ***(Anexo IX)***

X - Parecer do Conselho do Centro ao qual o requerente faz parte. ***(Anexo X).***

*Obs. A renovação de afastamento para qualificação em nível de pós-graduação stricto sensu dar-se-á nos termos da legislação em vigor, devendo a manifestação de intenção de renovação do afastamento ser protocolada em **até 60 (sessenta) dias antes do término do afastamento**. Conforme Art. 19. da RESOLUÇÃO CONSUNI/UFERSA Nº 003/2018, de 25/06/2018*

Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

Data: 25/03/2021
(**obrigatória**)

JOSE FERDINANDY SILVA CHAGAS:01486328318

Assinado de forma digital por JOSE FERDINANDY SILVA CHAGAS:01486328318
Dados: 2021.03.25 12:04:25 -03'00'

Assinatura do requerente
(**obrigatória**)

Dúvidas? Leia a: *RESOLUÇÃO CONSUNI/UFERSA Nº 003/2018, de 25 de junho de 2018.*

DECLARAÇÃO

Declaramos, para os fins que se fizerem necessários, que **José Ferdinandy Silva Chagas**, portador(a) do CPF nº 014.863.283-18, matrícula Siape nº 3608635, é servidor(a) do Quadro Permanente desta Universidade, admitido(a) em 08 de abril de 2014, ocupante do cargo de Professor do Magistério Superior, com lotação no(a) Departamento de Engenharias e Tecnologia do Centro Multidisciplinar de Pau dos Ferros.

Eu, Laura Maria Araújo Mendes Pereira, ocupante do cargo de Assistente em Administração, digitei e conferi a presente declaração, conforme dados extraídos do Sistema Integrado de Administração de Recursos Humanos – SIAPE, nesta data.

Pau dos Ferros/RN, 23 de março de 2021.

DHOUGO ARAGONES AMARO DA SILVA:01031095446

Assinado de forma digital por DHOUGO ARAGONES AMARO DA SILVA:01031095446
Dados: 2021.03.23 19:12:46 -03'00'

D'hougo Aragonês Amaro da Silva
Diretor

DECLARAÇÃO

Declaramos, para os fins que se fizerem necessários, que **José Ferdinandy Silva Chagas**, portador(a) do CPF nº 014.863.283-18, é servidor(a) do Quadro Permanente desta Universidade, admitido(a) em 08 de abril de 2014, ocupante do cargo de Professor do Magistério Superior, com lotação e exercício no(a) Departamento de Engenharias e Tecnologia do Centro Multidisciplinar de Pau dos Ferros.

Eu, Laura Maria Araújo Mendes Pereira, ocupante do cargo de Assistente em Administração, digitei e conferi a presente declaração, conforme dados extraídos do Sistema Integrado de Administração de Recursos Humanos – SIAPE, nesta data.

Pau dos Ferros/RN, 23 de março de 2021.

DHOUGO ARAGONES AMARO DA SILVA:01031095446

Assinado de forma digital por DHOUGO ARAGONES AMARO DA SILVA:01031095446
Dados: 2021.03.23 19:11:29 -03'00'

D´hougo Aragonês Amaro da Silva
Diretor

DECLARAÇÃO

Declaramos, para os fins que se fizerem necessários, que **José Ferdinandy Silva Chagas**, Matrícula SIAPE nº **3608635**, foi admitido(a) nesta Universidade em 08 de abril de 2014, ocupante do cargo de Professor do Magistério Superior.

Declaramos, outrossim, que o(a) servidor(a) possui de efetivo exercício prestado neste Órgão, no referido provimento, até a presente data, o tempo de contribuição de **2542** dias, correspondente a **6** anos, **11** meses e **22** dias, entre o período de 08/04/2014 a 23/03/2021.

Em dias

TEMPO BRUTO	2542
Faltas	- -
Licenças	- -
Licenças sem vencimentos	- -
Suspensões	- -
Disponibilidades	- -
Outras	- -
TEMPO LÍQUIDO	2542

Eu, Laura Maria Araújo Mendes Pereira, ocupante do cargo de Assistente em Administração, digitei e conferi a presente declaração, conforme dados extraídos dos assentamentos funcionais do servidor(a) e do Sistema Integrado de Administração de Recursos Humanos – SIAPE, e em observação a legislação vigente nesta data.

Pau dos Ferros/RN, 23 de março de 2021.

DHOUGO ARAGONES AMARO DA SILVA:01031095446

Assinado de forma digital por DHOUGO ARAGONES AMARO DA SILVA:01031095446
Dados: 2021.03.23 19:10:50 -03'00'

D'hougo Aragonês Amaro da Silva
Diretor

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO - UFERSA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO - PROPPG
Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

(Anexo I)

Check-List – Renovação de Afastamento para qualificação
(**obrigatório**)

Nome do solicitante: JOSÉ FERDINANDY SILVA CHAGAS	
Local da Qualificação*:*	
X No País ☐ No exterior	
Período solicitado para renovação do afastamento: 21/05/2021 a 01/03/2022	
Documentos Anexados – Processo de Renovação:	**Número da página (Preenchido pela PROPPG):**
I. Lista de verificação própria disponibilizada pela PROPPG (Check-List); ***(Anexo I)***	
II. Justificativa de seu requerimento; (***Anexo II***)	
III. Relatório de atividades acadêmicas (***Anexo III***)	
IV. Relatório de avaliação de desempenho, feito pelo orientador (***Anexo IV***)	
V. Declaração de Matrícula ***(Anexo V)***	
VI. Histórico Escolar – Atualizado ***(Anexo VI)***	
VII – Termo de Compromisso, devidamente preenchido e assinado com testemunhas; (Anexo VIII)	
VIII. Documentação que formalize a substituição do(a) interessado: ***(Anexo VIII)*** **X** Utilização de vaga ou disponibilidade de professor substituto a ser contratado(a) ☐ Termo de Compromisso dos docentes que assumirão as disciplinas	
IX**.** Parecer da chefia imediata (Departamento acadêmico de lotação do requerente); ***(Anexo IX)***	
X. Parecer do Conselho do Centro ao qual o requerente faz parte. ***(Anexo X).***	

Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

(Anexo II)

JUSTIFICATIVA PARA O AFASTAMENTO

(Obrigatório)

EU, JOSÉ FERDINANDY SILVA CHAGAS, portador do CPF nº 014.863.283-18, RG nº 2003030026003, matrícula siape nº 3608635, professor da área de computação dos cursos de Bacharelado em Ciência e Tecnologia (BC&T), Bacharelado em Tecnologia da Informação (BTI) e Bacharelado Engenharia da Computação do *Campus* de Pau dos Ferros da Universidade Federal Rural do Semiárido – UFERSA, venho solicitar a renovação do afastamento integral das minhas atividades no período de 21/05/2021 à 01/03/2022 para continuação do curso de Doutorado em Ciência da Computação pela Universidade Federal de Campina Grande – UFCG/PB.

No doutorado estou atuando em pesquisas na área de engenharia de software inteligente, no laboratório Embedded da UFCG/PB, faço parte do grupo ISE – Intelligent Software Engineering (http://isegroup.org). O que demanda tempo e dedicação, uma experiência que está agregando muito para minha formação como pesquisador.

Em 2021 irei desenvolver as atividades necessárias para conclusão do doutorado que envolvem experimentos computadorizados e entrevistas com praticantes da indústria. A intervenção será realizada no VIRTUS (https://www.virtus.ufcg.edu.br), Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação – um órgão suplementar da Universidade Federal de Campina Grande (UFCG) vinculado ao Centro de Engenharia Elétrica e Informática (CEEI).

Por fim, fui classificado no *ranking* para solicitação de afastamento, segundo o Edital de Qualificação Docente Condicionado à Contratação de Substituto – Ano 2018. Assim, o *Campus* de Pau dos Ferros não terá suas atividades de ensino e extensão comprometidas na minha ausência.

Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

Data: 25 de março de 2021

JOSE FERDINANDY SILVA CHAGAS:01486328318

Assinado de forma digital por JOSE FERDINANDY SILVA CHAGAS:01486328318
Dados: 2021.03.25 12:08:24 -03'00'

Assinatura do requerente
(Obrigatória)

Dúvidas: ***RESOLUÇÃO CONSUNI/UFERSA Nº 003/2018, de 25 de junho de 2018.***

Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

(Anexo III)

RELATÓRIO DE ATIVIDADES ACADÊMICAS

Programa de Pós-graduação em Ciência da Computação (PPGCC/UFCG)
Aluno: José Ferdinandy Silva Chagas

Semestre 2020.1:

- Cumprimento de disciplinas:
  - INF062 - PROJETO DE PESQUISA REUSO DE CONHECIMENTO NO SUPORTE À TOMADA DE DECISÃO II)

- Publicação e apresentação do artigo "*On the Reuse of Knowledge to Develop Intelligent Software Engineering Solutions*" na 32ª Conferência Internacional em Engenharia de Software e Engenharia de Conhecimento (http://ksiresearch.org/seke/seke20.html).

Semestre 2020.2:

- Cumprimento de disciplinas:
  - INF062 - PROJETO DE PESQUISA REUSO DE CONHECIMENTO NO SUPORTE À TOMADA DE DECISÃO III)
  - CC007 – QUALIFICAÇÃO DE DOUTORADO

Data: 22 de março de 2021

José Ferdinandy Silva Chagas

José Ferdinandy Silva Chagas
SIAP 3608635

Hyggo Oliveira de Almeida

Prof. Dr. Hyggo Oliveira de Almeida
Orientador – PPGCC/UFCG

Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

(Anexo IV)

RELATÓRIO DE AVALIAÇÃO DE DESEMPENHO

Declaro que o aluno **José Ferdinandy SilvaChagas**, matriculado no Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Campina Grande (UFCG/PB), sob o número 011801580-5D, realizou suas atividades, conforme descrito no relatório de atividades acadêmicas nos semestres 2020.1 e 2021.2.

Data: 22 de março de 2021

Prof. Dr. Hyggo Oliveira de Almeida
Orientador – PPGCC/UFCG

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Uma Abordagem para Priorização de Dívida Técnica em Desenvolvimento Ágil de Software

José Ferdinandy Silva Chagas

Proposta de Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Hyggo Oliveira de Almeida
(Orientador)

Campina Grande, Paraíba, Brasil

Resumo

O conceito de Dívida Técnica (DT) foi introduzido por Ward Cunningham em 1992, fazendo referência às decisões sobre o código-fonte que podem trazer benefícios e acelerar entregas no curto prazo, mas podem impactar negativamente, em longo prazo, a qualidade interna do produto, custos e cronogramas. No contexto de Desenvolvimento Ágil de Software, a Dívida Técnica é potencializada pela pouca ênfase em documentação e um maior foco em entregas rápidas. Apesar do crescente interesse na área, faltam estratégias e ferramentas adequadas para apoiar atividades de gerenciamento de Dívida Técnica, tais como suporte à priorização. A priorização é a atividade no gerenciamento de DT que define qual item de DT será "pago" primeiro e quais itens podem prosseguir para os próximos estágios do desenvolvimento. Algumas soluções propostas na literatura definem critérios para a priorização da DT baseados em métricas relacionadas ao código e direcionadas por estratégias de melhor custo-benefício. Entretanto, essas soluções muitas vezes ignoram fatores externos como, por exemplo, o impacto da experiência do desenvolvedor na aquisição e acúmulo de DT. Neste trabalho, propõe-se uma abordagem para auxiliar na tomada de decisão sobre a priorização de DT através de reuso de conhecimento, baseada em dados históricos. A solução proposta inclui um modelo de processo adaptado do Scrum que considera atividades de priorização da DT. Além disso, é proposto um sistema de recomendação de priorização de DT baseado em dados históricos coletados por ferramentas de análise estática de código e ferramentas de gerenciamento de projetos. O modelo de processo adaptado do Scrum e o modelo de sistema de recomendação já foram construídos. Nas próximas etapas do trabalho será desenvolvida uma base de dados históricos de projetos com as métricas necessárias para a aplicação e validação do modelo, bem como a construção e validação de uma ferramenta de apoio que implementa a solução proposta.

Abstract

Ward Cuninham introduced the Technical Debt (TD) concept in 1992, which refers to decisions on source code that can bring benefits and speed up deliveries in the short term but can generate future problems or even make it impossible to continue projects in the long run. The problems generated harm the internal quality of the product, costs, and deadlines. In the agile context, the Technical Debt is enhanced by the low emphasis on documents and a greater focus on quick delivery. Despite the growing interest in the area, there is a lack of tools or approaches to support some management activities and comprehensive solutions that consider management as a whole. The lack of support resources makes it difficult for teams and agile project managers to work, leading to manually performing part of the management. In large projects, performing the management manually is not feasible. In agile projects, it creates additional steps that hinder the process, and consequently, an obstacle to applying the TD concept in the software industry. To optimize resource allocation that TD items should be prioritized. Prioritization is an activity in management that defines which TD item will be paid first and which items can proceed to the next development stages. The solutions use criteria for prioritizing TD based on code-related metrics aimed at the best cost-benefit ratio. However, these solutions often ignore external factors, such as the developer's experience on the acquisition and accumulation of TD. In an attempt to alleviate these problems and reduce the distance between the techniques and the practitioners of the industry, we propose an approach to assist in decision-making on the prioritization and management of TD through the reuse of knowledge based on historical data. The proposed solution involves a process model adapted from Scrum, considering TD prioritization activities and a recommendation system based on historical data collected by static code analysis tools and project management tools. We already built the process model adapted from Scrum and the formal model of the recommendation. In the next stages of the work, we will build a historical project database with the metrics necessary to validate the model and the construction and validation of a support tool that implements the solution.

Conteúdo

Lista de Símbolos

DAS - *Desenvolvimento Ágil de Software*

DoD - *Definition of Done*

DT - *Dívida Técnica*

US - *User Stories*

kNN - *k-Nearest Neighbors Algorithm*

MOEA - *Multi-objective Evolutionary Algorithm*

OE - *Objetivo Específico*

PO - *Product Owner*

ROI - *Return over Investment*

RQ - *Research Question* (Questão de Pesquisa)

SR - *Sistema de Recomendação*

TAM - *Technology Acceptance Model*

TEDIOUS - *Technical Debt Identification System*

UFCG - *Universidade Federal de Campina Grande*

VIRTUS - *Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

O Manifesto para Desenvolvimento Ágil de Software, publicado em 2001, declara um conjunto de quatro valores e doze princípios que apoiam e constituem a essência do desenvolvimento ágil de software [23]. O manifesto declara que, em qualquer situação em que uma escolha deve ser feita, a prioridade é dada para software funcionando e pessoas, em detrimento de documentação extensiva e processos [11].

Desenvolvimento Ágil de Software (DAS) é uma área da Engenharia de Software que envolve metodologias, métodos, técnicas e processos que seguem os princípios e valores do Manifesto Ágil. Ao contrário de processos considerados tradicionais, no DAS as fases são iterativas e incrementais, com foco na redução da sobrecarga processual, alta colaboração entre as partes interessadas e adaptação a mudanças nos requisitos [34]. Com a redução de custos, melhor produtividade, qualidade e satisfação, o DAS obteve grande disseminação na indústria de desenvolvimento de software nas duas últimas décadas [67].

Em DAS há uma forte ênfase no autogerenciamento da equipe de desenvolvimento. Como as mudanças são bem-vindas, a equipe deve sempre revisar sua estrutura, relacionamento e organização. Em cada iteração, a equipe deve manter as práticas que servem as suas necessidades e alterar aquelas que foram obstáculos no seu caminho. Cada iteração deve incluir planejamento, análise de requisitos, projeto, codificação e teste, pois a ênfase é sempre na entrega de software funcionando.

Estas características benéficas para a produtividade no DAS também podem levar à redu-

ção na atenção a aspectos importantes do desenvolvimento de software, tais como boas práticas de arquitetura, projeto e programação, qualidade e cobertura de testes, documentação mínima necessária à evolução, dentre outras. Tais características podem gerar consequências negativas dando origem a Dívidas Técnicas.

O conceito de Dívida Técnica (DT) foi introduzido por Ward Cunningham em 1992 [19] como um meio para comunicar os desafios que surgem das consequências de um desenvolvimento de software descuidado. Desde então, profissionais e pesquisadores têm evoluído o conceito para abranger diferentes aspectos do desenvolvimento de software que envolvem desde problemas arquiteturais e de projeto até problemas de documentação e pessoas. Cunningham definiu o conceito como uma analogia ao débito financeiro quando ao contrair uma dívida podemos obter vantagens no curto prazo, mas podemos enfrentar problemas com "juros" maiores no futuro caso a dívida não seja paga.

A DT tem se tornado um conceito popular no DAS, relacionando suas implicações econômicas e as características específicas do DAS que o torna propenso a incorrer em DT. Como exemplo das consequências negativas da DT em DAS, companhias que aplicam DAS podem enfrentar um aumento na duração das entregas, aumentando os custos, e falhar ao tentar manter a qualidade. Além disso, em alguns casos, as implicações podem levar à perda de mercado por dificultar o relacionamento com as partes interessadas [8]. Entretanto, quando devidamente gerenciada e controlada, a DT pode ser utilizada de forma estratégica para ganhar oportunidades de negócio [41; 58].

Várias pesquisas têm sido realizadas para entender as perspectivas e implicações da DT em DAS. Porém, as conclusões dos pesquisadores sugerem que há uma lacuna sobre o entendimento do conceito de DT e seu gerenciamento. É importante entender a DT nas perspectivas teórica e prática para o avanço do estado da arte [40]. Devido à falta de solidificação do conceito, praticantes e pesquisadores têm proposto classificações de DT para auxiliar o entendimento e o seu gerenciamento. Steve McConnell propôs uma taxonomia para classificar DT em "intencional" e "não intencional" [55]. Já Martin Fowler [31] classifica a DT como "prudente" e "imprudente", afirmando que ambas podem ser intencionais ou não intencionais.

O gerenciamento de DT envolve um conjunto de atividades e ferramentas que visam prevenir ou reduzir a DT no processo de desenvolvimento. As principais atividades são:

identificar, medir, monitorar, priorizar, documentar, comunicar, pagar e prevenir a DT [77].

Por exemplo, quando não há regras de gerenciamento de DT bem definidas os desenvolvedores podem fazer escolhas que podem contribuir para o acúmulo de DT no projeto. Consequentemente, isso prejudica a produtividade da equipe, a qualidade do produto, aumentam os custos do projeto, e prejudica os relacionamentos de negócio. Compreender DT em DAS é importante para tomar decisões adequadas no tempo certo [10].

Por outro lado, pagar toda a dívida técnica presente no sistema é inviável e desnecessário. Para otimizar a alocação de recursos os desenvolvedores devem priorizar os itens de DT. A priorização é a atividade no gerenciamento de dívida técnica que define qual item de DT será pago primeiro e quais itens podem prosseguir para os próximos estágios do processo de desenvolvimento [6]. Um obstáculo encontrado na priorização é o aumento da complexidade da priorização da dívida técnica a cada novo item identificado. Além disso, é necessário definir quais critérios devem ser utilizados para incluir ou excluir um item da lista de pagamento da dívida.

Várias pesquisas já foram realizadas com foco em dívida técnica de código, considerando violações de código como critério para priorização [30; 17; 6]. Alguns trabalhos consideraram a baixa produtividade dos desenvolvedores, como problemas de arquitetura causando retrabalho [80; 79], e outros consideram como o desenvolver trabalha [50; 73]. Entretanto, poucas pesquisas foram realizadas considerando violações de código e o impacto do fator humano na priorização da dívida técnica.

Apesar de existirem diversos trabalhos na área de priorização e gerenciamento de dívida técnica, pesquisadores reconhecidos na área como Li *et al.* [45] apontam que faltam pesquisas que apresentem de forma detalhada modelos, processos e ferramentas para realizar as atividades de gerenciamento da Dívida Técnica.

É neste contexto de estratégias para abordar a Dívida Técnica no Desenvolvimento Ágil de Software que se insere este trabalho. Mais especificamente, busca-se auxiliar no processo de priorização de dívida técnica considerando o fator humano no processo.

1.2 Problemática

A literatura aponta para diferentes razões para explicar por que DAS é propenso a acumular DT. Um exemplo é a característica da pouca ênfase em práticas de documentação para priorizar a entrega de software funcionando [33]. Outro exemplo é a falta de atenção à arquitetura do software [69]. Um estudo realizou um mapeamento de pesquisas sobre implantação contínua e apontou que, como consequência da relação entre a rápida implantação do software e baixo rigor no desenvolvimento, teste e práticas para garantir a qualidade, as organizações tendem a incorrer DT ao longo do tempo [66]. Assim, equipes ágeis precisam ser cuidadosas com DT, suas implicações econômicas, estratégias para pagamento da DT e outras implicações relacionadas ao DAS.

Existem várias lacunas no gerenciamento de DT que precisam ser preenchidas. As limitações como a falta de ferramentas ou estratégias adequadas para apoiar algumas atividades e a falta de soluções compreensíveis que consideram o gerenciamento como um todo dificultam a tarefa das equipes e dos gerentes de projeto. Este problema leva à necessidade de executar parte do gerenciamento manualmente ou personalizar ferramentas e estratégias. Porém, em grandes projetos com uma quantidade elevada de itens de dívida técnica esta tarefa pode ser inviável. A dificuldade para solucionar estas questões pode ser o obstáculo para a aplicação prática do conceito de DT na indústria de software [63].

Os desenvolvedores possuem um papel fundamental no sucesso do projeto. Estudos apontam que os fatores humanos, tais como estrutura organizacional, experiência e o foco em produzir mais funcionalidades impactam diretamente a qualidade do software [13; 24; 57; 62]. Além disso, problemas que prejudicam a qualidade do software causam um impacto negativo na manutenibilidade do sistema e na satisfação do usuário [13; 53]. O aumento inevitável na demanda por software com flexibilidade para mudanças tem criado desafios para o gerenciamento da qualidade, prevenir defeitos e para a manutenção do código.

Além disso, a Engenharia de Software é uma das áreas no mercado com maior ingresso de desenvolvedores juniores. A distorção entre o número de desenvolvedores juniores e seniores nas equipes ágeis está crescendo rápido. Esse fator aumenta o estresse significativamente no processo de capacitação das equipes. Nessas condições, ferramentas que fornecem *feedback* sobre a qualidade interna do software através da análise do código-fonte são importantes

para que os desenvolvedores possam aprender e melhorar suas habilidades de codificação com mínima consulta aos colegas mais experientes. Algumas investigações empíricas indicam que desenvolvedores juniores são menos familiares com o conceito de DT do que seus colegas mais experientes [47].

Neste contexto, enuncia-se então o problema abordado neste trabalho: como auxiliar os gerentes de projeto na tomada de decisão sobre a priorização de dívida técnica de código, considerando o débito acumulado e a inexperiência dos desenvolvedores, de maneira integrada ao processo de desenvolvimento?

1.3 Objetivos

O objetivo geral neste trabalho é propor um modelo para auxiliar no processo de tomada de decisão sobre a priorização de dívida técnica no Desenvolvimento Ágil de Software. Mais especificamente, propõe-se uma abordagem de recomendação integrada ao processo de desenvolvimento para incluir explicitamente o gerenciamento de dívida técnica e a priorização de dívida técnica utilizando dados de projetos em andamento e dados históricos de projetos.

A abordagem de recomendação utiliza dados históricos de projetos para identificar projetos com tecnologias similares para recomendar a priorização da dívida técnica de código considerando o débito acumulado em itens de dívida técnica na base histórica e a experiência do desenvolvedor responsável. A recomendação é disponibilizada ao gerente de projetos para auxiliar na tomada de decisão sobre o pagamento da dívida técnica em uma dada iteração.

A abordagem é integrada ao arcabouço ágil Scrum, com adaptações para o gerenciamento da dívida técnica. Scrum foi escolhido porque é um dos métodos ágeis mais utilizados na indústria. A flexibilidade, popularidade e simplicidade do arcabouço permitem sua aplicação em diversos domínios [4].

Para validação do trabalho, será realizado um estudo experimental com dados de projetos de software reais desenvolvidos no Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação (VIRTUS) da Universidade Federal de Campina Grande (UFCG).

Com base no exposto anteriormente, os seguintes objetivos específicos são definidos:

- OE1 – Investigar quais os desafios enfrentados por profissionais na priorização de DT;
- OE2 – Propor um modelo para gerenciamento de dívida técnica integrado ao arcabouço Scrum;
- OE3 – Modelar um sistema de recomendação e priorização de tarefas para pagamento de dívida técnica;
- OE4 – Construir uma base de dados históricos de projetos de software para utilizar como base para o sistema de recomendação;
- OE5 – Avaliar a solução proposta por um estudo experimental utilizando os dados coletados por ferramentas de análise estática de código e ferramentas de gerenciamento de projetos.

1.4 Metodologia

A metodologia utilizada nesse trabalho é baseada no método científico de i) definição de questões de pesquisa e hipóteses; ii) proposição de soluções; e iii) avaliação experimental dos resultados buscando responder às questões de pesquisa e confirmar/refutar as hipóteses. A seguir são apresentadas as questões de pesquisa, as hipóteses e, por fim, as atividades definidas para a execução do trabalho, destacando aquelas que já foram realizadas até o momento.

1.4.1 Questões de Pesquisa

Para cada um dos objetivos específicos apresentados anteriormente foram definidas questões de pesquisa que embasaram o processo de abordagem, proposição de soluções e validação. Estas questões de pesquisa são apresentadas a seguir.

Objetivo Específico (OE1)

Para o Objetivo Específico (OE1) há questões de investigação inicial da pesquisa:

- RQ01 - Qual o nível de entendimento dos profissionais sobre o conceito de dívida técnica e seu gerenciamento?

- RQ02 – Como os profissionais realizam o gerenciamento e priorização da DT?
- RQ03 - Qual a relação entre o acúmulo de dívida técnica com a experiência do desenvolvedor?

Objetivo Específico (OE2)

Para o Objetivo Específico (OE2), há uma questão para direcionamento da solução:

- RQ04 – Como fornecer suporte às equipes ágeis no gerenciamento de dívida técnica no processo ágil?

Objetivo Específico (OE3)

Para o Objetivo Específico (OE3) há questões específicas da solução:

- RQ05 – Quais fatores podem ser utilizados para priorização da DT de projeto e código?
- RQ06 – Como pode ser realizada a priorização da DT utilizando dados históricos?

Objetivo Específico (OE4)

Para o Objetivo Específico (OE4) tem-se uma questão para direcionamento do reuso de conhecimento a ser utilizado na solução:

- RQ07 - Quais dados históricos são necessários para construção de uma Base de Dados de DT, para viabilizar o reuso de conhecimento para priorização de novos itens?

Objetivo Específico (OE5)

Para o Objetivo Específico (OE5) há uma questão para direcionamento da solução:

- RQ08 - Qual a utilidade prática da solução proposta?

1.4.2 Hipóteses

Com base nas questões de pesquisa, as hipóteses definidas para este trabalho são as seguintes:

H1. A experiência do desenvolvedor impacta diretamente no acúmulo da dívida técnica em nível de código.

A prova é por análise estatística do acúmulo de dívida técnica pelo desenvolvedor.

H2. É possível reduzir o acúmulo da dívida técnica considerando um gerenciamento explícito dentro do processo ágil utilizando uma lista priorizada de dívida técnica.

A prova é por simulação com os dados do tempo de conclusão das tarefas de pagamento de dívida técnica em projetos reais.

1.4.3 Atividades

A seguir são listadas as atividades associadas para cada questão de pesquisa definida.

1. Realizar uma revisão na literatura sobre o gerenciamento e priorização de dívida técnica (RQ1,RQ2, RQ3);
2. Aplicar um *survey* para avaliar o nível de entendimento e aplicação do conceito de dívida técnica e seu gerenciamento entre os praticantes da indústria (RQ1,RQ2, RQ3);
3. Desenvolver um modelo de gerenciamento de dívida técnica alinhado com o processo definido pelo Scrum (RQ4);
4. Identificar quais critérios podem ser utilizados para a priorização de DT de projeto e de código (RQ5);
5. Definir um modelo para priorização de dívida técnica utilizando dados históricos (RQ6);
6. Identificar quais dados históricos podem ser utilizados para a construção de uma Base de Dados de DT buscando viabilizar o reuso de conhecimento associado à dívida técnica (RQ7);
7. Realizar a validação do modelo de processo adaptado do Scrum (R4);
8. Realizar a validação do sistema de recomendação da priorização (R6);

9. Desenvolver ferramenta de apoio que implementa a abordagem proposta para auxiliar os gerentes e desenvolvedores no processo ágil (RQ8).

10. Realizar estudos de caso para avaliar a utilidade e facilidade de uso da abordagem (RQ8).

Até o momento, as atividades 1, 2, 3, 4 e 5 foram concluídas. Para obter uma visão macro sobre o entendimento dos desenvolvedores sobre dívida técnica e gerenciamento de dívida técnica, foi realizada uma revisão na literatura e aplicado um *survey* com 66 questões sobre DT aplicado a desenvolvedores com uma grande diversidade de perfis em várias partes do país. Com o objetivo de definir um processo para o gerenciamento da dívida técnica de forma integrada com o processo ágil foi proposto um modelo de processo adaptado do Scrum, e um modelo formal da recomendação baseada em dados históricos. As demais atividades serão realizadas conforme cronograma apresentado no último capítulo do documento.

1.5 Relevância

O trabalho se insere na área de Engenharia de Software, com relevância observada nas perspectivas científica, industrial e institucional.

Do ponto de vista científico o trabalho busca contribuir para a base de conhecimento em dívida técnica apresentando uma nova abordagem baseada em técnicas inteligentes para discussão. Segundo a literatura, ainda existem diversas lacunas da dívida técnica na Engenharia de Software, faltando ferramentas, modelos e abordagens bem definidas. Este trabalho tem potencial de avanço do estado da arte na área.

Na perspectiva industrial o trabalho busca reduzir a distância entre os problemas e soluções discutidos na comunidade científica e os problemas enfrentados no dia a dia das organizações. Além disso, o trabalho colabora na área de qualidade de software, que é considerado um ponto estratégico do processo de desenvolvimento. O melhor gerenciamento da dívida técnica traz impacto positivo na gestão de qualidade, de riscos e de custos em projetos de software. Com o objetivo alcançado espera-se que a equipe seja capaz de gerenciar adequadamente a dívida técnica e que desenvolvedores juniores incorram menos em dívida técnica em projetos futuros.

Por fim, institucionalmente, este trabalho contribui para o avanço nas pesquisas do grupo ISE(*Intelligent Software Engineering*) da UFCG, que investiga a aplicação de técnicas inteligentes para a melhoria da produtividade na prática de Engenharia de Software.

1.6 Estrutura do Documento

O documento está organizado da seguinte forma: no Capítulo 2 apresentam-se conceitos da fundamentação teórica; no Capítulo 3, discute-se o estado da arte no gerenciamento de dívida técnica; no Capítulo 4 apresenta-se a discussão sobre os estudos iniciais realizados; no Capítulo 5 apresenta-se o modelo de processo adaptado do Scrum proposto no trabalho; o Capítulo 6 descreve-se o modelo do sistema de recomendação; por fim, no Capítulo 7, descreve-se o estado atual da pesquisa e o cronograma de atividades para conclusão.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos envolvidos na elaboração do trabalho de pesquisa. São abordadas as definições, tipos e atividades do gerenciamento da dívida técnica, priorização, e automação no processo de gerenciamento. Além disso, são apresentados conceitos sobre sistemas de recomendação que estão relacionados com a solução proposta.

2.1 Dívida Técnica

O conceito de Dívida Técnica foi introduzido por Ward Cunninghan em 1992 como uma metáfora associada a problemas encontrados durante o processo de desenvolvimento de software que são ocasionados pela execução inadequada de tarefas. Segundo Cunningham, é comum a equipe incorrer em dívida técnica, mas o problema surge quando a dívida não é paga e, então, o tempo gasto com código inadequado conta como juros [19].

Inicialmente o conceito era associado ao escopo de código, mas com os anos o conceito passou a abranger as outras etapas do desenvolvimento de software. Como Avgeriou *et al.* [9] definem no seu trabalho, a dívida técnica como um conjunto de blocos de projeto ou implementações que são oportunos no curto prazo, mas podem tornar mudanças mais custosas ou impossíveis no futuro. Ainda segundo Avgeriou *et al.* [9], os problemas causados pelo acúmulo de dívida técnica impactam principalmente na manutenibilidade e capacidade de evolução do sistema.

Na literatura já foram identificados vários tipos de DT. Entre os tipos mais citados estão as DTs de: requisitos, arquitetura, projeto, código, teste, *build*, documentação, versiona-

mento, social e defeitos. Os itens de DT podem ser classificados nesses tipos de DT e podem estar presentes em diversos tipos de artefatos.

Na Tabela 2.1 são apresentados os principais tipos listados por Li e Rios [45; 63].

Tipo	**Descrição**
Requisitos	Representa a diferença entre a implementação atual do sistema e a especificação dos requisitos.
Arquitetura	Representa problemas associados a decisões de projeto que comprometem a qualidade interna do sistema, como confiabilidade, manutenibilidade ou segurança.
Projeto	Representa problemas ocasionados por "atalhos" especificados no projeto detalhado do sistema.
Código	Representa código mal escrito que viola as boas práticas de programação.
Teste	Representa problemas ocasionados por escolhas que comprometem a etapa de testes, como por exemplo falta de cobertura, ou falta de testes de unidade.
Build	Problemas no processo de compilação que tornam a compilação mais complexa.
Documentação	Problemas ocasionados pela falta de documentação de qualquer etapa do desenvolvimento de software que pode levar, por exemplo, a uma interpretação equivocada de requisitos e dificultar a manutenção por falta de documentação de código.
Infraestrutura	Refere-se a configurações inadequadas de tecnologias, processos e ferramentas associadas ao projeto em desenvolvimento.
Versionamento	Problemas ocasionados por decisões inadequadas no versionamento do código, como, por exemplo, a criação de versões desnecessariamente.
Defeito	Refere-se às falhas e *bugs* encontrados no sistema de software.
Social	Problemas com pessoal da equipe de desenvolvimento que podem ocasionar o atraso do projeto ou prejudicar a qualidade.

Tabela 2.1: Tipos de dívida técnica.

Ainda que sejam tipos listados em vários trabalhos, não há um consenso entre os pesqui-

sadores sobre todos eles, assim como sobre o próprio conceito de DT. No presente trabalho o escopo é restrito à DT de projeto, código e social. Exemplos de DT de código e projeto são códigos duplicados, classes inutilizadas, métodos não utilizados e métodos demasiadamente longos.

Para evitar o acúmulo da DT é necessário um gerenciamento adequado. O gerenciamento de DT envolve um conjunto de atividades e ferramentas que visam prevenir ou reduzir a DT durante o processo de desenvolvimento. Segundo Li *et al.* [77], as principais atividades do gerenciamento de DT são: identificar, medir, monitorar, priorizar, documentar, comunicar, pagar e prevenir a DT. Na Tabela 2.2 são listadas as atividades do processo de gerenciamento com suas respectivas descrições.

Atividade	**Descrição**
Identificação	Atividades relacionadas à identificação de itens de dívida técnica gerados por ações intencionais ou não.
Medição	Atividades de quantificação e estimativa da dívida técnica em um software. Pode envolver a estimativa de itens individuais ou a dívida técnica geral associada ao projeto.
Priorização	Atividades que geram uma lista ordenada dos itens de dívida técnica que serão pagos seguindo regras predefinidas.
Prevenção	Atividades que buscam evitar a aquisição de dívida técnica não-intencional. Podem envolver atividades como treinamento da equipe, por exemplo.
Monitoramento	Atividades que avaliam periodicamente os itens de dívida técnica não pagos, avaliando os custos e os benefícios ao longo do tempo.
Pagamento	Atividades que reduzem a dívida técnica resolvendo os itens de dívida técnica identificados. Podem envolver atividades de reengenharia ou refatoramento de código, por exemplo.
Documentação	Atividades de representação e registro da dívida técnica associada ao projeto para comunicação com os *stakeholders*.
Comunicação	Atividades que tornam a dívida técnica visível para os *stakeholders* para facilitar as discussões.

Tabela 2.2: Atividades do gerenciamento de dívida técnica.

Em grande parte das pesquisas encontradas na literatura são apresentados modelos, *frameworks* e ferramentas de suporte às atividades específicas de gerenciamento e para determinados tipos de dívida técnica. Poucos trabalhos abordam o processo completo de gerenciamento. Rios *et al.* [63] categorizaram diversas ferramentas para as atividades do gerenciamento de dívida técnica. Entre as ferramentas mais citadas está o SonarQube e FindBugs. Estas ferramentas são conhecidas e utilizadas por praticantes na indústria.

No contexto do DAS as pesquisas apontam que os projetos ágeis estão mais propensos a apresentar DT. A justificativa é a ênfase na entrega rápida de funcionalidades com menor preocupação com aspectos da qualidade interna do sistema, como boas práticas de programação e cobertura de testes. Isso leva ao surgimento da DT, comprometendo prazos, custos e a qualidade do produto final [12].

Assim, as pesquisas focadas em gerenciamento de DT em DAS procuram reduzir o esforço e, consequentemente, os custos de manutenção e os prazos de entrega. Entretanto, a maioria dos estudos empíricos no contexto do gerenciamento de DT em DAS apresentam apenas lições aprendidas. Segundo Behutiye *et al.* [12], isso ocorre pela insuficiência de ferramentas de suporte, modelos, *frameworks* e *guidelines*.

2.2 Priorização de Dívida Técnica

A DT sempre está presente nos projetos de software, o que prejudica o processo de desenvolvimento e a qualidade do produto. É questionável e desnecessário pagar toda a dívida técnica. O problema surge em como classificar quais dívidas devem ser pagas no curto prazo e quais dívidas podem prosseguir para serem pagas em etapas posteriores. Segundo Alfayez *et al.* [6], as estratégias e fatores estão principalmente relacionados com valor, custo e restrição de recursos.

No contexto de dívida técnica, a priorização está geralmente associada com os conceitos de “principal” e “juros”. Rios *et al.* [63] definem tais conceitos como:

- Principal: é o esforço/custo que será necessário para eliminar a dívida. Por exemplo, o esforço para refatorar o código ou definir novos casos de teste.

- Juros: é o risco associado ao não pagamento no curto-prazo da dívida técnica. O risco

está no aumento do esforço para atividades futuras ou na queda da produtividade da equipe, por exemplo.

Há um dilema na priorização da DT para os profissionais responsáveis por tomar as decisões do projeto. É necessário definir se a priorização será escolhida entre itens de dívida técnica ou se uma dívida técnica poderá ser priorizada em relação a uma funcionalidade. Considerando esses fatores, as principais estratégias para a priorização são baseadas em: qualidade interna do software; produtividade; correção do software; análise de custo-benefício; ou combinação dessas estratégias.

Para cada estratégia são utilizados fatores específicos que podem variar de acordo com o contexto. Segundo Lenarduzzi *et al.* [42], os principais fatores utilizados em trabalhos da literatura podem ser classificados de acordo com a Tabela 2.3.

As estratégias de gerenciamento de dívida técnica estão diretamente relacionadas com os conceitos de dívida principal e juros. São os conceitos que representam o custo/esforço e o impacto da dívida técnica ao longo do tempo, respectivamente. Esses aspectos mudam de acordo com o contexto do projeto e o interesse do cliente e da equipe de desenvolvedores.

2.3 Automação no Gerenciamento de Dívida Técnica

O processo de desenvolvimento de software gera vários artefatos compostos por dados que podem ser utilizados para medição da dívida técnica. A medição da DT requer uma grande quantidade de dados, mas também requer muito esforço para extrair e tratar estes dados. Assim, são necessárias ferramentas de apoio que possam realizar esse processo de forma automática sem sobrecarregar os desenvolvedores e permitindo o monitoramento contínuo da DT [35].

Várias iniciativas já foram realizadas para criação de ferramentas de apoio ao gerenciamento da dívida técnica. Rios *et al.* [63] apontam que existem diversas ferramentas que apoiam tecnologias específicas e etapas específicas do processo de gerenciamento de DT. Algumas dessas ferramentas são listadas na Tabela 2.4.

Apesar de haver uma variedade de ferramentas de suporte ao tratamento de dados sobre o processo de desenvolvimento, na área de pesquisa em DT as ferramentas ainda não possuem

Categoria	**Fatores**
Negócio	Vantagem competitiva, prazo de entrega, atratividade para o mercado, penalidades, uso de recursos, valor de negócios, ROI.
Cliente	Satisfação, valor para o cliente, expectativas do cliente, efeito para o cliente.
Evolução	Tempo do impacto na evolução, risco de impacto na evolução, impacto em outras funcionalidades, impacto em funcionalidades futuras.
Manutenção	Modificabilidade, número de *bugs*, custo de manutenção.
Qualidade do Sistema	Robustez, desempenho, segurança, escalabilidade.
Dívida de Qualidade	Número de *issues* ou sua coocorrência.
Produtividade	Número de desenvolvedores trabalhando na DT, horas de desenvolvimento desperdiçadas, esforço de codificação.
Projeto	Disponibilidade de recursos, tamanho e complexidade do projeto, adiamento de *bugs*.
Social	Moral do desenvolvedor, Dívida social, Impacto positivo da DT, Cultura da equipe.
Outros	Propagação do impacto no sistema, número de usuários afetados, histórico do sistema, custos futuros, etc.
Não especificado	Risco, probabilidade de juros, juros, severidade, e critério personalizado.

Tabela 2.3: Fatores utilizados para priorização de DT.

maturidade adequada. A maior parcela das pesquisas apresentam propostas de novas abordagens que automatizam parcialmente o processo recorrendo a ferramentas prototipadas apenas para dar suporte ao estudo. Este fato dificulta a aplicação das tecnologias desenvolvidas na indústria [35].

Ferramenta	**Descrição**
SonaQube	Ferramenta de análise estática de código que realiza avaliações contínuas sobre a qualidade do código baseada no método SQALE. A ferramenta detecta: *bugs*, *code smells* e vulnerabilidades.
JCaliper	Automaticamente extrai o número, tipo e sequência de atividades de refatoração que são necessárias para obter um design otimizado.
PyDriller	*Framework* Pyton que permite a extração de dados de repositórios Git. Extrai dados como: a mensagem de *commit*; o número de desenvolvedores; modificações; *diffs*; e o código-fonte de um *commit*.
Ptidej	Ferramenta para detecção de *code smells* e *anti-patterns* em código Java.
Refactoring Miner	Ferramenta *open-source* que recebe uma lista de *commits* e retorna uma lista de operações de refatoramento realizadas entre *commits* consecutivos.

Tabela 2.4: Ferramentas para automação no gerenciamento de DT.

2.4 Sistemas de Recomendação

Os sistemas de recomendação podem ser definidos como sistemas que guiam usuários para itens de seu interesse, ou gera itens do seu interesse, dentro de uma grande variedade de opções [14; 38]. Existem vários exemplos de aplicações práticas de sistemas de recomendação em sistemas comerciais, como nas plataformas da Amazon e Netflix. O sucesso destas aplicações contribuiu para a aplicação de sistemas de recomendação em novos domínios, tais como na Engenharia de Software [27].

2.4.1 Abordagens

As principais abordagens utilizadas por sistemas de recomendação são: filtragem baseada em conteúdo; filtragem colaborativa; recomendação baseada em conhecimento; e recomendação por grupos [27].

Filtragem Baseada em Conteúdo

A filtragem baseada em conteúdo considera os itens que o usuário demonstrou interesse no passado e com base nessa informação o sistema tenta identificar novas recomendações.

Exemplos de recomendações desse tipo são encontradas em sistemas de *e-commerce* onde itens são recomendados a partir de compras que o usuário realizou no passado [59].

Filtragem Colaborativa

A filtragem colaborativa é utilizada para recomendar itens como música e filmes [37; 39; 46]. Esta abordagem considera as preferências de usuários com interesses similares e parte da ideia de que o interesse em um item pode ser influenciado pela opinião dos usuários com os mesmos interesses. Por exemplo, se um usuário A comprar um filme similar ao usuário B, o sistema irá recomendar para A os filmes que B comprou e que ainda não foram adquiridos por A. A principal diferença para a filtragem por conteúdo é a necessidade de mais informações sobre o interesse dos outros usuários e menos informações sobre o item [27].

Recomendação Baseada em Conhecimento

A recomendação baseada em conhecimento é uma abordagem baseada em regas e restrições. Esta abordagem é utilizada em contextos sem muitas informações atualizadas e disponíveis sobre o interesse dos usuários. Por exemplo, itens como casas, apartamentos e carros que não são comprados com tanta frequência. O principal desafio dessa abordagem está na aquisição do conhecimento [26].

Recomendação por Grupo

A recomendação por grupo é uma abordagem que considera recomendações direcionadas para um grupo de usuários. Então, diferente das outras abordagens, para a recomendação é considerada a preferência do grupo e não a preferência individual do usuário. Por exemplo, filmes para um grupo de amigos, ou requisitos de software que devem ser implementados [52].

2.5 Sistemas de Recomendação na Engenharia de Software

Na Engenharia de Software, os sistemas de recomendação podem oferecer suporte evitando a sobrecarga de informações geradas no processo de desenvolvimento [65]. Os SRs podem

auxiliar no processo de desenvolvimento, por exemplo, na recomendação de requisitos [56], recomendação de código [18; 54], ou recomendação metodológica [15; 60].

Sistemas de recomendação podem auxiliar engenheiros de dados e conhecimento no tratamento da complexidade e tamanho das estruturas de conhecimento [16]. Eles podem auxiliar indicando ações de correção e refatoramento para bases de conhecimento mal construídas [29].

Os sistemas de recomendação podem auxiliar no desenvolvimento e manutenção de bases de conhecimento, por exemplo, recomendando caminhos para navegação na base [27].

Em relação às mudanças nas bases de conhecimento por demanda do domínio, os sistemas de recomendação podem auxiliar recomendando testes e ações de correção nas bases otimizando as mudanças necessárias para manter a consistência [28].

2.6 Scrum

O Scrum é um *framework* para auxiliar as equipes de desenvolvimento de software a gerar valor com soluções adaptáveis para problemas complexos. Ele é fundamentado na ideia de que o conhecimento vem da experiência e as decisões devem ser tomadas com base em fatos [68].

O *framework* aplica uma abordagem incremental com iterações e combina quatro eventos formais de inspeção e adaptação, dentro de um evento maior chamado *sprint*. O trabalho é desenvolvido por profissionais, aos quais são atribuídos os seguintes papéis: *Product Owner*, *Scrum Master* e os Desenvolvedores [68]. Na Figura 2.1 apresenta-se uma visão geral do Scrum. A seguir são listadas as atribuições de cada papel, os eventos e artefatos do Scrum.

2.6.1 Papéis no Scrum

Product Owner

É o responsável por otimizar o valor do produto desenvolvido. Mais especificamente, ele é responsável por realizar o gerenciamento do chamado *Backlog* do Produto, o conjunto de funcionalidades que devem ser desenvolvidas para entregar valor ao cliente. No gerenciamento, o PO deve estabelecer e comunicar o objetivo do produto, definir e comunicar os

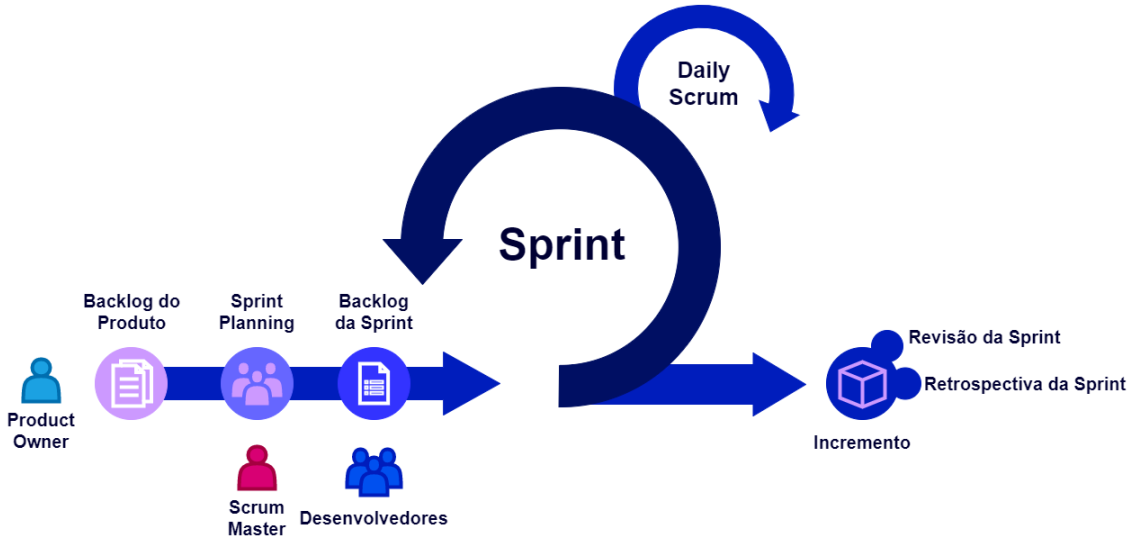

Figura 2.1: Visão geral do processo baseado no Scrum.

itens do Backlog do Produto, indicar quais itens devem ser desenvolvidos na *sprint*.

Scrum Master

O *Scrum Master* é o responsável por garantir que o fluxo do Scrum seja compreendido e seguido pelos outros membros da equipe. Ele direciona o auto-gerenciamento e a multifuncionalidade da equipe. Além disso, ele auxilia a equipe a manter o foco em entregas de valor ao cliente e remove os obstáculos do progresso da equipe. Também assegura a produtividade dentro de um espaço de tempo. No nível organizacional, o Scrum Master auxilia no planejamento e divulgação do Scrum nas empresas.

Em relação ao PO, o *Scrum Master* auxilia tecnicamente na definição dos objetivos do produto e gerenciamento do Backlog do Produto. Ele auxilia a equipe no entendimento dos itens do *backlog* e facilita a colaboração com *stakeholders* quando necessário.

Desenvolvedores

Os desenvolvedores são os responsáveis por implementar as funcionalidades baseadas nos itens do Backlog do Produto definido pelo PO e que foram incluídos no Backlog da Sprint durante a reunião de *sprint planning* para serem entregues no próximo incremento. As ha-

bilidades necessárias aos desenvolvedores variam de um projeto para o outro de acordo com o domínio da aplicação. Eles são responsáveis por definir o Backlog da Sprint, garantir a qualidade e adaptar seus planos todos os dias para atender o objetivo da *sprint*.

2.6.2 Eventos do Scrum

A *Sprint*

A *sprint* é o evento principal do Scrum e inclui todos os outros eventos. Durante este evento é realizado todo o trabalho para atingir a meta do produto. A *sprint* permite uma previsibilidade assegurando o monitoramento e as adaptações necessárias para atingir a meta do produto. Elas tendem a ser realizadas em períodos curtos de um mês ou menos, para reduzir a complexidade, limitar os riscos e evitar o aumento de custos e esforço.

Sprint Planning

A *Sprint Planning* é o evento em que são definidas as entregas que serão incluídas no incremento da *sprint*. A equipe seleciona os itens do Backlog do Produto e refina esses itens em tarefas menores que então são atribuídas, estimadas e catalogadas no Backlog da Sprint. As tarefas definidas são mapeadas para atingir a meta da *sprint*. Este evento é crucial para a *sprint* e neste encontro são discutidos principalmente o valor que será entregue e o trabalho que deverá ser feito.

Daily Scrum

O *Daily Scrum* é um encontro de curto-período realizado uma vez por dia para verificar o progresso das atividades que foram estabelecidas no Backlog da Sprint. Os desenvolvedores se reúnem para discutir o andamento das atividades, identificar impedimentos, agilizar a tomada de decisão quando necessário, ou replanejar as atividades com o objetivo de atingir a meta da *sprint*.

Revisão da Sprint

A revisão da *sprint* é um encontro da equipe Scrum com os *stakeholders* com o objetivo de discutir o trabalho que foi realizado e as entregas que ficaram prontas dentro do planejado

para a *sprint*. Nesse ponto os participantes avaliam as mudanças necessárias e definem os próximos passos.

Retrospectiva da Sprint

Este evento tem o objetivo de melhorar a qualidade e a eficiência do trabalho realizado pela equipe Scrum. Nesse encontro são discutidas as principais dificuldades, que podem variar de acordo com o domínio do projeto. São discutidos os eventos que ocorreram durante a *sprint*, o que foi benéfico, ou o que prejudicou o progresso da *sprint*, e o *Definition of Done*.

2.6.3 Artefatos do Scrum

Os artefatos gerados no fluxo do Scrum representam o trabalho realizado. Eles são projetados para representar as informações principais do projeto de forma clara e transparente. O objetivo é que todos que fazem parte do projeto possuam o mesmo entendimento sobre o progresso do projeto e seus objetivos: o Backlog do Produto representa as metas do produto, o Backlog da Sprint as metas da *sprint*; e o incremento representa o *Definition of Done*.

2.7 Considerações Finais do Capítulo

Os conceitos e definições sobre Dívida Técnica ainda não possuem maturidade suficiente. Há diversas discussões conflituosas na literatura sobre a própria definição de DT, sobre os tipos de DT e a aplicabilidade de suas estratégias. Existem linhas de pesquisa que consideram DTs apenas relacionadas ao código, enquanto outras já consideram DTs ocasionadas por infraestrutura física e de pessoal. Outro ponto de discussão é a migração que a temática de DT ocasionou de pesquisadores que já trabalhavam em áreas relacionadas e que mudaram a abordagem reposicionando os trabalhos no contexto de Dívida Técnica. Isso ocorre, por exemplo, com pesquisadores da área de Testes e de Refatoramento. Por fim, como citado anteriormente as abordagens propostas não possuem maturidade adequada e ainda faltam soluções automatizadas que possam contribuir com o gerenciamento e priorização da DT.

Um subtópico também discutido nos trabalhos em DT é a chamada Dívida Técnica autoadmitida (*self-admitted*) [71], que ocorre quando os próprios desenvolvedores antecipam a identificação da DT utilizando comentários no código. Os desenvolvedores sinalizam os

pontos do código que precisam de revisão posteriormente. Este tópico ficou fora do escopo no nosso trabalho por tratar-se de um tipo muito subjetivo de DT, apesar de estar diretamente relacionada com o código, mas que dificulta o reuso de dados históricos e o tratamento automatizado.

Capítulo 3

Trabalhos Relacionados

A seguir são listadas pesquisas encontradas na literatura relacionadas com o contexto do presente trabalho. As pesquisas envolvem coleta de dados por meio de ferramentas automáticas, gerenciamento e priorização de dívida técnica, bem como sistemas de recomendação para dívida técnica.

3.1 Priorização da Dívida Técnica

Diversas abordagens já foram propostas para a priorização da dívida técnica. A maior parcela das propostas definem objetivos e critérios específicos para realizar a priorização. Além disso, muitos trabalhos restringem o escopo da priorização em determinados tipos de dívida técnica. O critério para escolha do tipo de dívida técnica e fatores que serão utilizados para a priorização varia de acordo com a disponibilidade de dados, impacto na avaliação de custo-benefício, entre outras questões. A seguir são apresentados trabalhos que discutem propostas de abordagens para priorização de dívida técnica de código, arquitetura e projeto [42].

Zazworka *et al.* [79] apresentam uma proposta baseada em análise de custo-benefício para a priorização da DT com o objetivo de reduzir o acúmulo de dívida técnica de projeto causada pela criação de *God Classes*. Esse item de DT ocorre quando o desenvolvedor cria uma classe com muitas responsabilidades. A abordagem proposta é baseada em análise de métricas e mineração de repositórios.

Arcelli Fontana, Ferme e Spinelli [30] apresentam uma proposta que prioriza o refatoramento de *code smells* considerando o impacto em métricas de qualidade com o objetivo

de identificar os itens de DT que podem causar maior dano. Esta abordagem considera o nível de correção do software como estratégia para a priorização. Os autores restringiram o escopo do trabalho em três tipos comuns de *code smells*: *Data Class*, *God Class* e código duplicado.

Akbarinasaii *et al.* [3] apresentam também uma abordagem para priorização de DT de defeito, considerando a severidade e o tempo estimado para correção. As DTs de defeito são falhas que não são corrigidas na *release* atual e são adiadas para as *releases* seguintes. Na abordagem proposta os autores identificam os "juros" da DT de defeito.

Codabux *et al.* [17] construíram um modelo de uma rede bayesiana para estimar a probabilidade de cada item de DT. A abordagem dá flexibilidade para as organizações decidirem qual será o tipo de DT foco da priorização. Além disso, a abordagem utiliza ferramentas para extração de métricas para a construção do modelo preditivo (Rede Bayesiana). Com base no modelo a equipe pode priorizar os itens de DT mais problemáticos e mais arriscados.

Poucos trabalhos apresentam uma combinação de abordagens diferentes. Um exemplo desses trabalhos é apresentado por Alfayez e Bohem [6], onde os autores apresentam uma proposta de priorização automatizada baseada em busca usando um algoritmo multiobjetivo chamado MOEA. Este algoritmo tem o objetivo de priorizar os itens de DT para otimizar o pagamento da DT respeitando restrições de custo específicas. A abordagem não é direcionada para nenhum tipo específico de DT, mas a pesquisa apresenta dois experimentos para avaliação da abordagem utilizando dados extraídos pelo SonarQube.

Dos artigos analisados, o trabalho mais próximo ao desenvolvido na presente pesquisa foi desenvolvido por Vidal *et al.* [74]. Os autores propuseram a ferramenta JSpirit, que prioriza DT de código considerando diferentes critérios. A ferramenta realiza um cálculo de ranqueamento para uma lista de *code smells* identificados. Então, classifica os itens de acordo com a sua importância. A importância é definida por critérios como: relevância do *code smell*, histórico do sistema e outras métricas do software.

Uma publicação relevante de Tornhill [73] apresenta uma ferramenta para visualização e análise preditiva para priorização de DT chamada CodeScene, que considera entre outros fatores a forma como os desenvolvedores trabalham com o código. A ferramenta realiza uma análise de complexidade baseada em métricas e na indentação do código para então definir uma priorização final com o suporte de um especialista.

Um trabalho técnico que serviu de base para o estudo apresentado aqui foi o *Technical Debt Dataset* [44], em que pesquisadores utilizaram ferramentas de análise estática e de mineração de repositórios (SonarQube, Jira, Ptidej) para coletar e catalogar dados de dívida técnica de 33 projetos da *Apache Software Foundation*. O trabalho apresenta o potencial das ferramentas de coleta automática e várias oportunidades para pesquisadores e praticantes da indústria com a utilização dessa base de dados.

3.2 Sistemas de Recomendação para Dívida Técnica

Apesar de vários trabalhos apresentarem propostas de abordagens para priorização com o objetivo de recomendar qual item de DT deve ser pago primeiro, foram encontrados poucos trabalhos na literatura que apresentam propostas explícitas de sistemas de recomendação para priorização de dívida técnica. Alguns trabalhos restringem a recomendação para tipos específicos de *code smells*, recomendam estratégias de refatoração e recomendação de código.

Daniela *et al.* [72] propõem um sistema que recomenda a refatoração de código duplicado e métodos longos em Java. Segundo os autores, apesar das ferramentas de análise estática serem frequentemente usadas, elas não levam a melhoria real da qualidade. Faltam aos desenvolvedores mecanismos estruturados de priorização para iniciar o processo de melhoria da qualidade.

Hanzhang *et al.* [76] propõem combinar o uso da engenharia de software baseada em busca com séries temporais para recomendar boas estratégias de refatoração com o objetivo de gerenciar a dívida técnica. Os pesquisadores utilizaram um algoritmo multiobjetivo para gerar recomendações de refatoração que maximizam a correção de problemas de qualidade importantes e minimizam o esforço.

Shirin *et al.* [2] apresentam uma categorização de defeitos em "regular" e defeitos "propenso a dívida". A abordagem proposta compara os dois tipos de defeitos para determinar a dívida principal, os juros atuais e a probabilidade de juros da dívida de defeito ao longo do tempo. Além disso, os pesquisadores propõem uma abordagem de aprendizagem por reforço para agendar quais dívidas de defeito precisam ser pagas e quando elas precisam ser pagas.

Fiorella *et al.* [78] propõem um sistema de recomendação chamado TEDIOUS (*Techni-*

cal debt identification system), para auxiliar os desenvolvedores durante a escrita do código recomendando quando admitir a dívida e quando melhorar a qualidade do código. As recomendações geradas por uma abordagem de aprendizado de máquina são baseadas em dados históricos de métricas de códigos de projetos anteriores. Os pesquisadores afirmam que a abordagem influenciou positivamente a legibilidade, tamanho e métricas de complexidade, tão bem quanto ferramentas de análise estática de código.

3.3 Considerações Finais do Capítulo

Assim como várias pesquisas em priorização de DT, no presente trabalho restringimos o escopo à DT de código e de projeto. Esta restrição foi considerada no intuito de aproveitar as vantagens e oportunidades viabilizadas pelos dados gerados pelas ferramentas de gerenciamento de projetos e análise estática de código. Por sua vez, os dados gerados podem viabilizar a automação de tarefas do gerenciamento de DT como a atividade de priorização.

O presente trabalho se diferencia das demais pesquisas citadas neste capítulo por direcionar seus esforços para o contexto de DT no Desenvolvimento Ágil de Software, onde há uma alta frequência de DT. Além disso, o gerenciamento adequado pode agilizar o processo, garantindo a qualidade e gerar oportunidades de negócio. Outro ponto de destaque é a proposta do modelo baseado no arcabouço Scrum, e de uma ferramenta de apoio automatizada que, diferente de outras propostas, evita a criação de passos adicionais distantes do dia a dia das equipes de desenvolvimento. Essa característica da solução poderá contribuir para viabilizar a aplicação das estratégias de DT na indústria.

Capítulo 4

Estudos Iniciais

Neste capítulo são apresentadas uma revisão da literatura e um *survey* aplicado com praticantes da indústria. O objetivo desse estudo inicial foi identificar quais os principais desafios enfrentados na priorização e gerenciamento da dívida técnica, e qual o impacto da experiência do desenvolvedor na DT (Objetivo Específico 1). As seções estão separadas em: revisão da literatura, a aplicação do *survey* e as considerações finais do capítulo.

4.1 Revisão da Literatura

Com o crescente interesse no tema de Dívida Técnica na Engenharia de Software foram realizadas várias publicações sobre o tema, bem como foram realizadas revisões sistemáticas da literatura nos últimos cinco anos. Para o propósito deste trabalho não foi realizada uma nova revisão sistemática por haver um número suficiente de publicações de trabalhos secundários e terciários sobre dívida técnica. Entre os trabalhos que mais se destacam estão: o mapeamento sistemático realizado por Li *et al.* [45]; a revisão sistemática da dívida técnica no contexto de desenvolvimento ágil de Behutiye *et al.* [12]; a revisão sistemática em priorização da dívida técnica de Lenarduzzi *et al.* [42]; e o estudo terciário de Rios *et al.* [63] sobre os tipos de DT, estratégias de gerenciamento e tendências de pesquisa em DT.

No mapeamento sistemático realizado por Li *et al.* [45], os pesquisadores listaram os principais desafios encontrados durante o levantamento. Os desafios que dificultam o gerenciamento da dívida técnica e a aplicação de estratégias e conceitos da DT na indústria. Os principais desafios são:

- Desafios no gerenciamento da DT induzida por *stakeholders*, como a delimitação de prazos curtos, e a DT não-intencional adquirida por desenvolvedores;
- A dificuldade em traduzir para os *stakeholders* atributos da qualidade interna do produto para valor de negócio;
- Desafio na mensuração da DT para permitir um monitoramento adequado e estimar adequadamente seu custo;
- Desafio da prevenção da DT no contexto de desenvolvimento ágil devido à dificuldade de balanceamento entre reduzir o número de *bugs* e, ao mesmo tempo, melhorar a qualidade interna.

No contexto de desenvolvimento ágil, Behutiye *et al.* [12] apresentam uma revisão sistemática onde os autores apontam que o desenvolvimento ágil é mais propenso a adquirir e acumular dívida técnica por consequência do maior foco em entregas rápidas. Os autores afirmam que apesar de haver muitas publicações no contexto ágil, a maioria das pesquisas não apresenta propostas de modelos, *frameworks*, teorias, *guidelines* ou ferramentas que explicitamente sejam focadas em DT no contexto ágil.

Em relação à priorização da DT, Lenarduzzi *et al.* [42] apresentam um levantamento sobre as principais estratégias, critérios e ferramentas utilizadas para a priorização de DT. Segundo os autores as pesquisas em priorização são direcionadas para tipos específicos de DT, principalmente DT de projeto, código e arquitetura. Apesar de existirem outros tipos de DT, poucas categorias além das citadas possuem ferramentas de apoio que permitam a extração de dados ou monitoramento automatizados. Entre os fatores considerados para a priorização estão principalmente análises de custo-benefício e custos de manutenção.

Em uma visão geral, Rios *et al.* [63] apontam que as ferramentas de análise estática de código, que coletam indicadores, estão sendo cada vez mais utilizadas pela comunidade científica para analisar como a DT pode afetar os projetos de software. Rios também afirma que os tipos de DT relacionados ao código causam efeitos que são observados mais rapidamente por equipes de desenvolvimento.

Outra pesquisa realizada por Rios *et al.* [64] foi a aplicação de um *survey* para avaliar as causas e efeitos da DT. Os pesquisadores coletaram 107 respostas e identificaram 78 causas

e 66 efeitos da DT. Segundo a pesquisa, as principais causas da DT são: prazos curtos; planejamento inapropriado; falta de conhecimento; falta de processo bem definido; falta de boas práticas; gerenciamento ineficiente de projeto; falta de profissionais qualificados; falta de experiência; documentação incompleta; e falta de comprometimento da equipe. O ranqueamento das causas é apresentado na Figura 4.1.

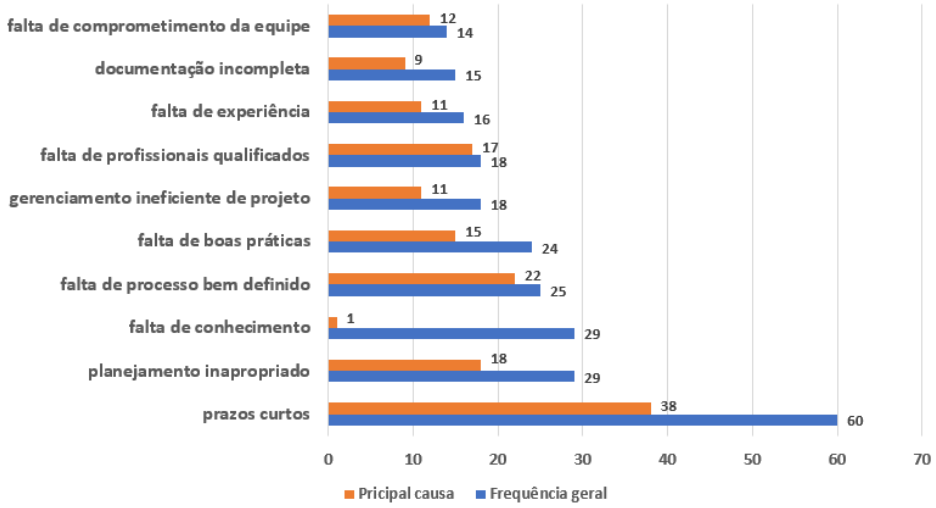

Figura 4.1: Principais causas da DT apontadas pelos praticantes (Adaptado [65]).

A pesquisa apontou que os principais efeitos causados pela DT são: baixa qualidade; atraso na entrega; baixa manutenibilidade; retrabalho; perdas financeiras; desmotivação da equipe; insatisfação dos *stakeholders*; documentação inadequada; baixo desempenho; e código ruim. O ranqueamento dos efeitos é apresentado na Figura 4.2.

Para cada tipo de DT foi contabilizado o número de efeitos associados. Os tipos de DT com maior número de efeitos são apresentados na Figura 4.3: dívida de projeto; dívida de documentação; dívida de código; dívida arquitetural; dívida de requisitos e dívida de testes.

Os trabalhos destacaram que, apesar de haver várias pesquisas sobre o tema, ainda há uma lacuna sobre a aplicação prática das soluções na indústria por falta de maturidade das soluções propostas, que algumas vezes criam passos adicionais no processo de desenvolvimento, o que inviabiliza sua aplicação.

Sobre a relação entre a experiência dos desenvolvedores e o aumento da dívida técnica no projeto, três trabalhos se destacaram. No primeiro, publicado em 2017, Theodoros *et al.* [7] realizaram um estudo com 60 desenvolvedores envolvidos em quatro projetos *open-source*.

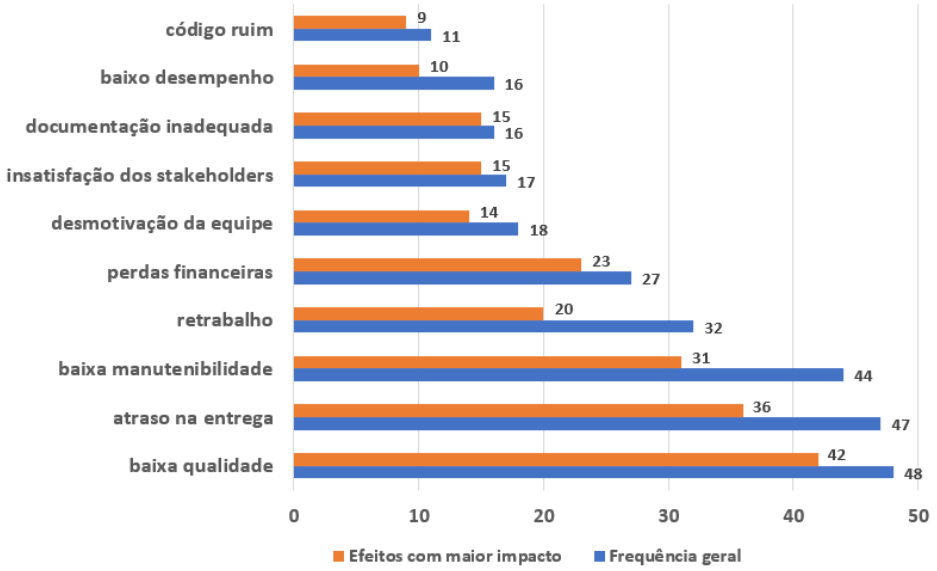

Figura 4.2: Principais efeitos da DT indicados pelos praticantes (Adaptado [65]).

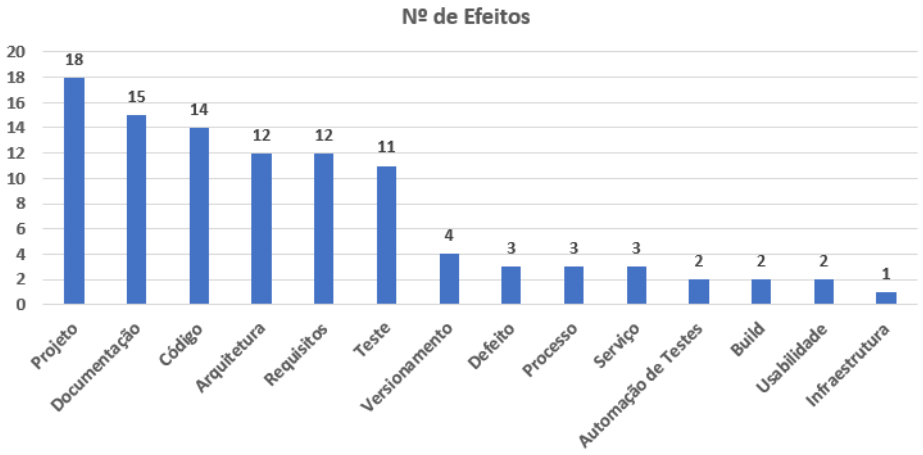

Figura 4.3: Número de efeitos associados aos tipos de DT (Adaptado [65]).

O objetivo foi investigar como a DT estava distribuída entre os desenvolvedores, os tipos de violações que foram causados por cada desenvolvedor, e qual a relação entre a maturidade do desenvolvedor e a tendência de acumular DT. Os pesquisadores não conseguiram obter evidências suficientes sobre a relação entre maturidade e acúmulo de DT. Entretanto, eles identificaram que 80% dos desenvolvedores que mais adicionaram DT estiveram ativos em menos de 33% do projeto.

No segundo trabalho, em 2018, Alfayez *et al.* [5] realizaram um estudo para investigar como diferentes desenvolvedores com diferentes características, como a experiência, estão

relacionados com a dívida técnica. Os pesquisadores identificaram que a responsabilidade pelo aumento ou redução da dívida técnica não é homogênea. Que a maturidade do desenvolvedor tem uma relação negativa com o aumento da DT, quanto maior a maturidade do desenvolvedor, menos DT será acumulada.

Por fim, Lenarduzzi *et al.* [43] realizaram em 2020 um estudo com 185 desenvolvedores juniores. Os pesquisadores identificaram que os desenvolvedores juniores tendem a adicionar mais *code smells* que outros tipos de DT. Também constataram que desenvolvedores juniores tendem a refatorar mais *code smells* que outros tipos de DT. Além disso, na pesquisa afirma-se que desenvolvedores juniores tratam igualmente todas as DTs, independentemente da severidade ou tipo. Por fim, os pesquisadores descobriram que os desenvolvedores juniores não gastam mais que 50% do tempo de correção dos itens de DT estimado por ferramentas de análise estática de código. Em alguns casos, o tempo real para correção foi 20 vezes menor que o estimado.

Para complementar a investigação sobre desafios da priorização da dívida técnica foi realizada a aplicação de um *survey* com profissionais da área. O objetivo foi obter uma perspectiva prática da aplicação de estratégias e da realização de atividades do gerenciamento e priorização da dívida técnica.

4.2 Aplicação do *Survey*

Além de revisões sistemáticas na literatura, existem trabalhos de pesquisa com aplicação de *surveys* na indústria, mas que não apresentavam a perspectiva dos profissionais no direcionamento necessário para o contexto do trabalho. As seções a seguir apresentam as observações com a aplicação do *survey*.

Entre os trabalhos publicados na literatura com resultados de *surveys* aplicados no Brasil, destaca-se o trabalho de Rios *et al.* [64] para melhor compreender a DT. Entretanto, para o propósito deste trabalho, viu-se a necessidade de informações complementares direcionadas para o contexto ágil e a priorização da dívida técnica. Assim, em parceria com outros pesquisadores interessados na temática de dívida técnica foi realizada a aplicação online de um *survey* com profissionais da indústria. O *survey* aplicado consiste em uma lista com 65 questões, onde 55 questões são de múltipla escolha e 10 questões são abertas. O questionário foi

respondido por 99 participantes que colaboraram de forma voluntária durante o ano de 2020. Destes 99 participantes, 14 se consideram iniciantes ou novatos. A seguir são apresentados os resultados relevantes ao tema de priorização da DT.

4.2.1 Perfil dos Respondentes

Os participantes do *survey* são profissionais em sua maioria com grande experiência prática no desenvolvimento de software. A pesquisa contou com a participação 2,1% de novatos, de 12,5% de profissionais iniciantes, 27,1% de profissionais qualificados, 26% proficientes, e 32,3% de especialistas.

Sobre experiência profissional, a pesquisa apontou que 22,9% possuem menos de 3 anos de experiência, enquanto 14,6% possui entre 3 e 5 anos, 25% entre 6 e 10 anos, 17,7% entre 11 e 15 anos, 14,6% entre 16 e 20 anos, e 5,2% com mais de 21 anos de experiência.

Dos papeis nas equipes de desenvolvimento, a maioria dos respondentes afirmou participar principalmente como desenvolvedor (65,6%), engenheiro de software (40,6%), líder de equipe (29,2%), e arquiteto de software (27,1%). Os resultados apresentam respostas duplicadas, pois os respondentes podem estar responsáveis por duas ou mais funções simultaneamente.

Quanto ao tamanho das equipes onde trabalham, os respondentes indicaram participar de equipes com até 5 membros (37,5%), outros em equipes entre 6 e 10 membros (32,3%), e (30,2%) participam de equipes com 11 ou mais membros.

Em relação à metodologia utilizada, muitas das equipes seguem metodologias ágeis (59,4%), enquanto outras utilizam metodologias híbridas (31,3%), e 9,3% utilizam a metodologia tradicional.

Em resumo, em sua maioria são desenvolvedores com boa experiência na área, trabalhando em equipes pequenas, seguindo metodologias ágeis. Um perfil adequado ao estudo proposto nesta pesquisa.

4.2.2 A Definição e o Gerenciamento de Dívida Técnica

A primeira etapa do *survey* aborda o conceito de Dívida Técnica. O questionário apresenta uma breve definição e questiona aos participantes qual é a proximidade da definição apre-

sentada com o entendimento que o indivíduo possui sobre o tema. Nesta questão, 86,5% responderam que a definição está próxima ou muito próxima do seu entendimento, enquanto 13,5% não tinha conhecimento prévio sobre DT ou a definição está muito distante do seu entendimento.

Quanto à identificação da DT a pesquisa aponta que em 70% dos casos a identificação é realizada de modo informal, apenas 12,5% realiza a identificação formal, e 17,5% não considera explicitamente a identificação de DT no processo de desenvolvimento. Por outro lado, 72,7% afirmaram que a aplicação de estratégias de identificação da DT no projeto é opcional, enquanto 27,3% afirma ser obrigatório. As principais ferramentas utilizadas na identificação da DT são apresentadas na Figura 4.4.

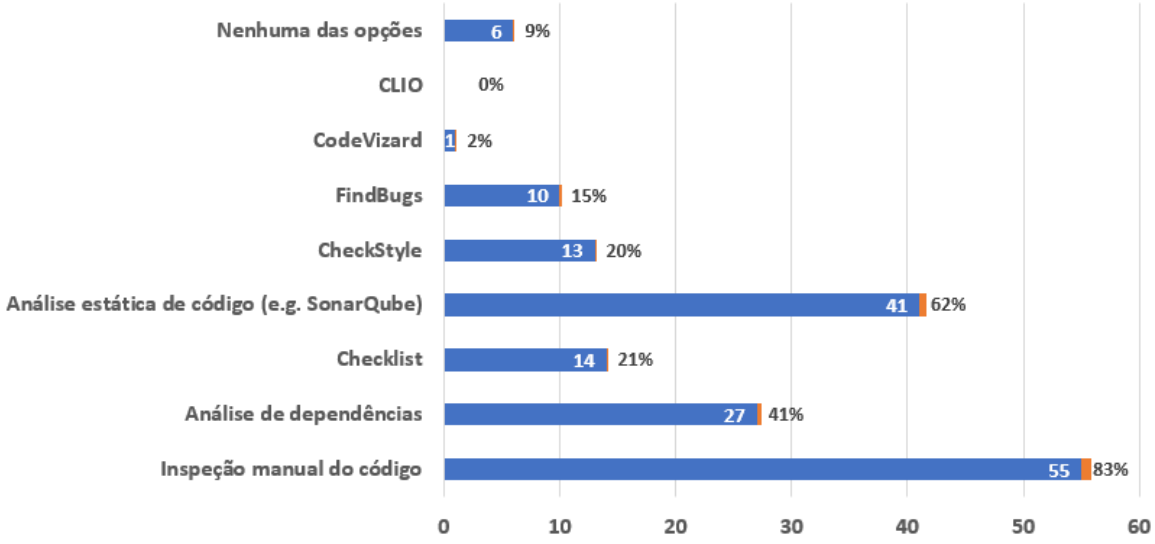

Figura 4.4: Principais técnicas utilizadas para identificação da DT.

Segundo a pesquisa, geralmente a dívida técnica não é documentada (53,8%), 32,5% realiza a documentação informalmente, e apenas 13,8% realiza documentação formal da DT. Esta resposta reflete também sobre a medição da DT. Entre os respondentes, 66,3% afirmam que a medição da DT não é realizada, enquanto 23,8% indicam uma medição informal, e apenas 10% declaram realizar a medição da DT formalmente.

Quanto ao pagamento da dívida técnica, a maioria dos respondentes (68,8%) afirmou realizar o pagamento sem seguir um processo prescrito, 16,2% afirmaram não pagar a DT e 15% declararam realizar o pagamento seguindo um processo formal prescrito. Em relação ao planejamento do pagamento, 68,7% afirmaram que o pagamento é realizado sem plane-

jamento e de acordo com a necessidade, enquanto 20,9% afirma pagar quando não é mais possível evitar, e apenas 10,4% afirma realizar o planejamento do pagamento da DT. Na Figura 4.5 apresentam-se as principais práticas utilizadas para pagamento da DT.

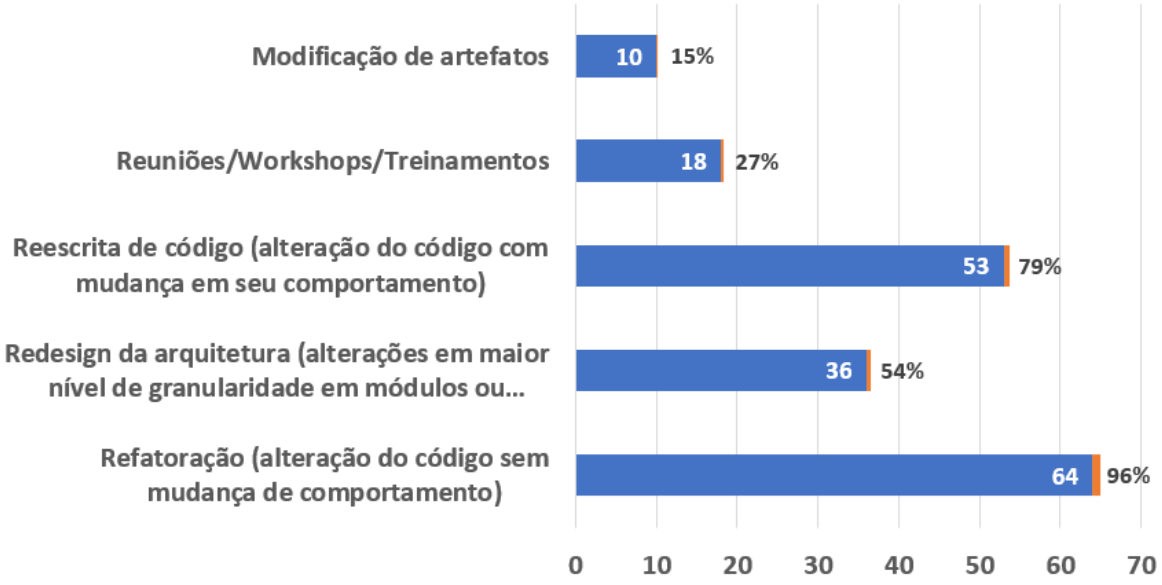

Figura 4.5: Principais técnicas utilizadas para pagamento da DT.

Assim, a percepção geral sobre o gerenciamento da DT na pesquisa é de que a maioria dos profissionais não possui a prática de realizar tais atividades, bem como as organizações não incluem as práticas de gerenciamento de DT na cultura organizacional. Quando as atividades são realizadas, na maioria dos casos, não seguem um processo formal prescrito e muitas vezes não é documentada.

4.2.3 A Priorização da Dívida Técnica

Em relação à atividade de priorização de DT, as respostam também refletem a falta de gerenciamento de DT. Entre os respondentes, 42,5% afirmam não realizar a priorização da DT, enquanto 43,7% afirmam realizar a priorização informalmente, e apenas 13,8% afirma realizar a priorização seguindo um processo formal. Na Figura 4.6 apresenta-se como a DT é priorizada.

Por fim, quanto aos critérios utilizados para a priorização da dívida técnica, os respondentes consideram os fatores representados na Figura 4.7. Os principais critérios considerados

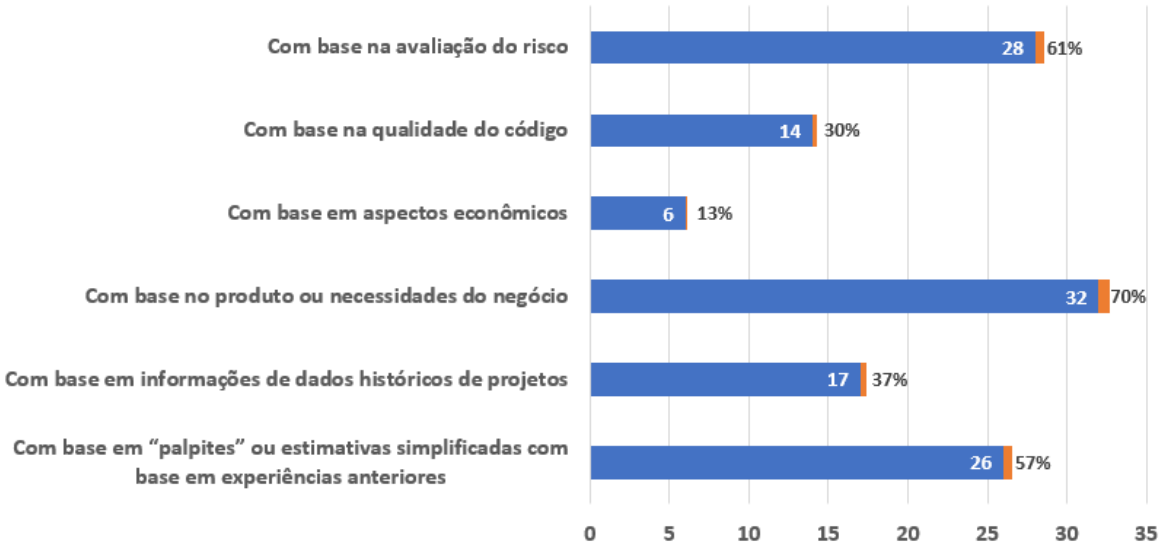

Figura 4.6: Abordagens utilizadas para priorização da DT.

estão alinhados com o entendimento encontrado na literatura que aponta uma tendência da priorização buscar o melhor custo-benefício.

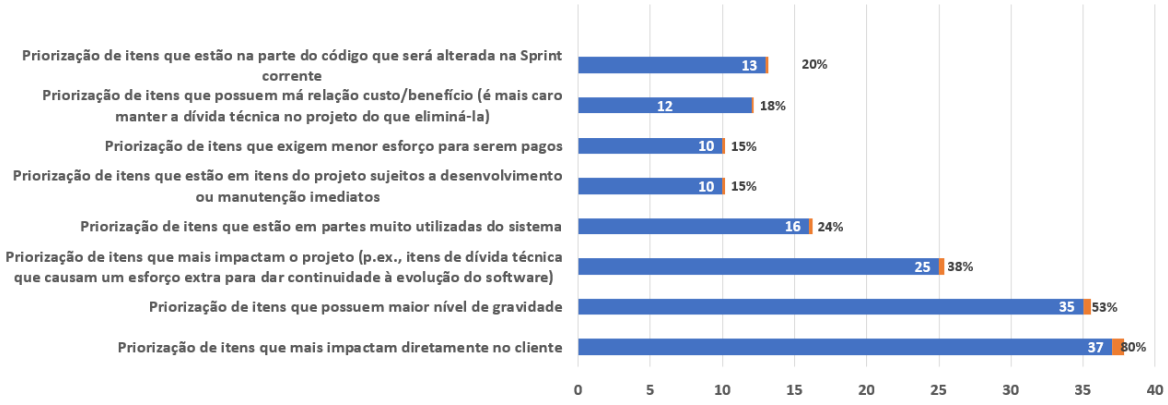

Figura 4.7: Principais critérios utilizados para priorização.

A partir dos estudos realizados foi possível obter uma visão geral sobre o entendimento e percepção da academia e dos praticantes da indústria sobre os problemas e desafios do gerenciamento da Dívida Técnica no contexto ágil. Em resumo, os desafios considerados para a proposta deste trabalho são:

- O foco em entregas rápidas aumenta a probabilidade de incorrer em DT no contexto de desenvolvimento ágil;

- Os profissionais precisam de ferramentas de apoio para realizar o gerenciamento e priorização de forma automatizada.

4.3 Considerações finais do Capítulo

Como foi apresentado nas seções anteriores, a problemática abordada na pesquisa foi identificada por uma revisão da literatura e aplicação de um *survey* com profissionais da indústria de desenvolvimento de software no país. A revisão foi realizada para identificar os principais desafios no gerenciamento e priorização da dívida técnica e a relação entre a maturidade do desenvolvedor e o acúmulo de DT. As conclusões obtidas com o estudo são:

- Os desenvolvedores possuem um bom entendimento sobre o conceito de DT e suas principais causas e efeitos (RQ1);
- Apesar de reconhecer a importância, os desenvolvedores encontram dificuldades em realizar as atividades do gerenciamento da DT. As atividades, no geral, quando são realizadas, são executadas sem um processo bem definido (RQ2);
- O acúmulo de DT no projeto é inversamente proporcional à maturidade do desenvolvedor (RQ3).

Capítulo 5

Processo baseado em Scrum com priorização de Dívida Técnica

Neste capítulo é apresentado um modelo de processo ágil adaptado do Scrum que inclui a priorização de Dívida Técnica (Objetivo Específico 2). A adaptação envolve novas responsabilidades aos papeis do Scrum, novas atividades no ciclo da *sprint* e um novo artefato chamado Lista Priorizada da DT.

Inicialmente, apresenta-se o fluxo do processo e suas etapas, destacando a adaptação ao processo padrão do Scrum. Em seguida, são apresentados os papeis, os eventos e artefatos do processo que inclui a priorização de Dívida Técnica.

5.1 O Fluxo do Processo e suas Etapas

O modelo proposto para suporte à priorização de DT no contexto de desenvolvimento ágil de software foi concebido de modo a impactar minimamente no processo. Assim, foi proposto um modelo adaptado do arcabouço Scrum como um guia para o gerenciamento formal da DT. O modelo é apresentado na Figura 5.1.

A seguir são apresentadas as etapas propostas no modelo. Para cada etapa são apresentadas as ações realizadas pela Equipe Scrum e são apresentados ao final de cada subseção os artefatos de entrada e saída envolvidos.

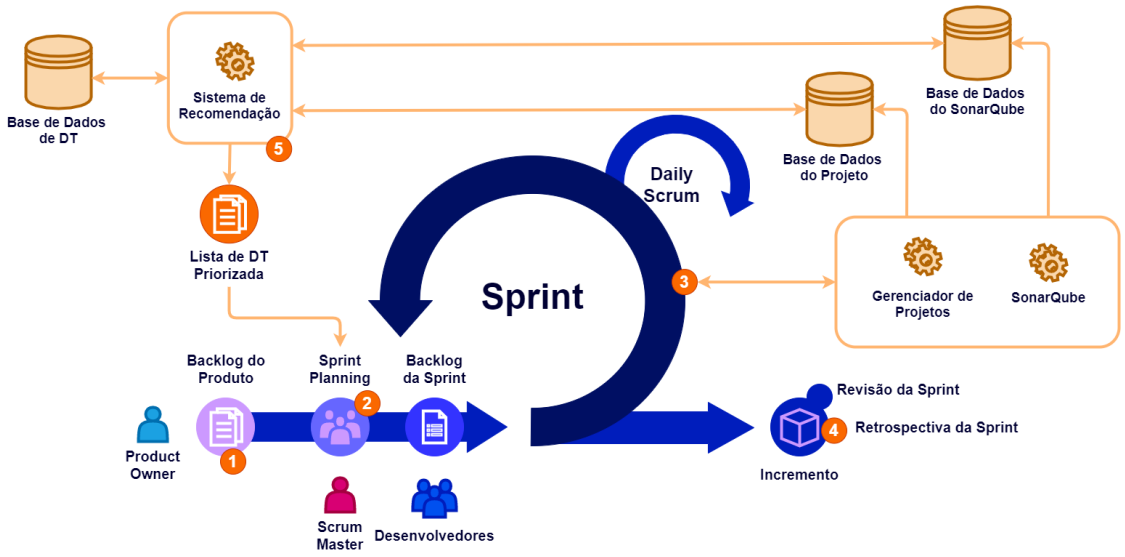

Figura 5.1: Modelo de processo adaptado do Scrum para DT.

5.1.1 Etapa 1

Na etapa inicial do projeto, o Product Owner (PO), em colaboração com os *stakeholders*, define um conjunto inicial de requisitos que devem compor os primeiros incrementos do projeto. Os requisitos e funcionalidades definidos nesta etapa são registrados no Backlog do Produto que guiará o desenvolvimento do projeto quanto às suas entregas e metas para o produto alvo. Este artefato está sujeito a modificações periódicas para atender as demandas dos clientes, usuários ou de negócios. Na Tabela 5.1 apresenta-se um exemplo de Backlog do Produto. Além de definir quais requisitos devem compor o produto, o PO ainda define a prioridade para cada requisito indicando a ordem na qual devem ser desenvolvidos.

Em relação à qualidade dos incrementos, na etapa inicial do projeto, o PO define em conjunto com a equipe Scrum os requisitos de qualidade do produto que devem ser atendidos. Estes requisitos são registrados no *Definition of Done*, um artefato utilizado como referência para o gerenciamento da qualidade do produto que será verificada nas reuniões de Retrospectiva da Sprint.

Assim, para o gerenciamento da DT, nesta primeira etapa a equipe inclui no *Denifition of Done* qual o nível tolerável de acúmulo de DT será aceito no projeto. O acúmulo da DT, no contexto deste trabalho, representa uma estimativa do esforço necessário para, refatorar o código, corrigir *bugs* ou adicionar novas funcionalidades. O nível tolerável é definido

ID	User Stories	Prioridade
1	O cliente deve conseguir comprar um livro escolhido por vez	4
2	O cliente deve conseguir pagar sua compra via boleto bancário	5
3	O cliente deve conseguir pagar sua compra via cartão de crédito	7
4	O cliente deve conseguir pesquisar livros por autor	1
5	O cliente deve conseguir pesquisar livros por título	2
6	O cliente deve conseguir pesquisar livros por editora	3
7	O cliente deve conseguir rastrear sua encomenda	6

Tabela 5.1: Exemplo de Backlog do Produto.

como a DT acumulada, representada em horas, que será aceita pela equipe. A definição pode ser realizada considerando-se uma porcentagem tolerável do tempo da *sprint*, ou outros critérios escolhidos pela equipe. Na Tabela 5.2 apresenta-se um exemplo de *Definition of Done* para um projeto e em destaque o item que representa o nível de DT que será tolerado. Considerando uma *sprint* de duas semanas (80 horas), a DT tolerada será de no máximo 50% da *sprint* ou 40 horas.

ID	Descrição
1	Testes unitários validados
2	Teste funcionais aprovados
3	Critérios de aceitação atendidos
4	Testes não funcionais atendidos
5	Product Owner aceita a história
6	**Dívida técnica acumulada abaixo de 40 horas**

Tabela 5.2: Exemplo de *Definition of Done* incluindo a tolerância para a DT.

Entrada (Etapa 1): Necessidades dos *stakeholders*, Critérios de Qualidade.

Saída (Etapa 1): Backlog do Produto, *Definition of Done*.

5.1.2 Etapa 2

Na reunião de *Sprint Planning*, a Equipe Scrum define quais funcionalidades devem ser implementadas durante a *sprint* com base nas demandas, prazos e prioridades definidas pelo PO no Backlog do Produto. Os itens são listados em breves descrições que precisam ser refinadas para um nível de tarefas que serão realizadas para atender ao requisito. Além disso, para cada tarefa é definida uma estimativa do esforço que será necessário para sua execução. Esse refinamento é realizado dentro da *Sprint Planning*. Uma forma de representação dos itens do Backlog do Produto são as chamadas *User Stories* (USs). Na Tabela 5.3 apresenta-se uma lista de USs refinadas ao nível de tarefas que precisam ser executadas na *sprint*. As USs refinadas em tarefas com suas especificações constituem o Backlog da Sprint, que é o principal artefato resultante deste evento Scrum.

ID	**US**	**Tarefa**	**Linguagem**	**Framework**
1	US01	Criar campos Login e Senha	HTML/CSS	Angular
2	US01	Criar rota de cadastro	JavaScript	Node
3	US01	Persistir no banco os dados de acesso	JavaScript	Node
5	US01	Criar rota de cadastro	JavaScript	Node
6	US01	Encriptar senha do novo usuário	JavaScript	Node
7	US01	Gerar senha padrão	JavaScript	Node
8	US01	Criar middlewares para validação do token	JavaScript	Node
9	US02	Criar tela alteração	HTML/CSS	Angular
10	US02	Criar rota alteração	HTML/CSS	Node

Tabela 5.3: Lista de USs refinadas em tarefas.

No momento da escolha dos itens que serão implementados no incremento, a Equipe Scrum precisa reservar parte da *sprint* para o pagamento da DT, que pode ser, por exemplo, 25% da *sprint*. O tempo reservado é definido com base nos prazos e o nível tolerável de DT especificado no *Definition of Done*. Caso a DT tenha ultrapassado o limite estipulado, o tempo reservado deve ser redefinido para atender ao critério previamente estabelecido. Estas informações estarão disponíveis no Relatório da DT, que é um artefato gerado após a avaliação da DT pela Equipe Scrum após a reunião de Retrospectiva da Sprint.

Uma vez definido o espaço na *sprint* para pagamento da DT, a Equipe Scrum analisa a

Lista Priorizada da DT, que é o artefato gerado pelo sistema de recomendação da priorização proposto neste trabalho (Capítulo 6). A Lista Priorizada contém a descrição da dívida, o arquivo do código-fonte onde a dívida foi identificada, o autor da dívida e o débito associado. Então, com base na priorização e tempo para pagamento da DT, a equipe seleciona quais itens devem ser incluídos no Backlog da Sprint para pagamento. Na Tabela 5.4 apresenta-se um exemplo de Backlog da Sprint com destaque para as atividades de pagamento da DT.

ID	**US**	**Tarefa**	**Linguagem**	**Framework**
1	US01	Criar campos Login e Senha	HTML/CSS	Angular
2	US01	Criar rota de cadastro	JavaScript	Node
3	US01	Persistir no banco os dados de acesso	JavaScript	Node
5	US01	Criar rota de cadastro	JavaScript	Node
6	US01	Encriptar senha do novo usuário	JavaScript	Node
7	US01	Gerar senha padrão	JavaScript	Node
8	US01	Criar middlewares para validação do token	JavaScript	Node
9	US02	Criar tela alteração	HTML/CSS	Angular
10	US02	Criar rota alteração	HTML/CSS	Node
11	**DT01**	**Refatorar Código Duplicado**	**JavaScript**	**Node**
12	**DT02**	**Refatorar Método Longo**	**JavaScript**	**Node**
13	**DT03**	**Refatorar God Class**	**JavaScript**	**Node**

Tabela 5.4: Backlog da Sprint com destaque das tarefas para pagamento da DT.

Entrada (Etapa 2): Backlog do Produto, Definition of Done, Lista Priorizada da DT, Relatório da DT.

Saída (Etapa 2): Backlog da Sprint.

5.1.3 Etapa 3

Após a *Sprint Planning*, quando são definidas as funcionalidades que serão implementadas, os desenvolvedores iniciam de fato a execução das tarefas especificadas no Backlog da Sprint. As tarefas são reavaliadas durante cada *sprint* dentro das reuniões diárias chamadas *Daily Scrum*. Reuniões de acompanhamento para avaliar o que foi feito, o que falta ser feito

e possíveis obstáculos identificados. Durante a execução das tarefas, os desenvolvedores realizam o envio do código-fonte desenvolvido para o repositório de código do projeto e complementam os dados das tarefas no Backlog da Sprint com o tempo real utilizado para execução da tarefa. Neste ponto do processo, o sistema de gerenciamento do projeto armazenará os dados atualizados do projeto e são identificados automaticamente os novos itens de DT no código-fonte adicionado ao repositório (usando uma ferramenta como o SonarQube). Na Tabela 5.5 apresenta-se um exemplo com a lista de itens de DT identificados pelo SonarQube. Os dados gerados nesta etapa serão utilizados como fonte de informações para a recomendação da priorização dos itens de DT. O Backlog do Produto e o *Definition of Done* são utilizados como referência para avaliar as metas do produto e de qualidade do projeto. Ao final da *sprint* é gerado um novo incremento que atende a meta do produto e deve estar dentro dos critérios estabelecidos no *Defniiton of Done*.

ID	ITEM DT	AUTOR	CÓDIGO	SEV.	DEBITO
d_1	Código Duplicado	dev01	rep\pproject01\crud\main.js	MAJOR	40
d_2	Método Longo	dev02	rep\pproject01\crud\cad.js	MAJOR	50
d_3	Código Duplicado	dev01	rep\pproject01\crud\rem.js	CRITICAL	120
d_4	God Class	dev03	rep\pproject01\crud\main.js	MINOR	40
d_5	Código Duplicado	dev01	rep\pproject01\crud\buscar.js	MAJOR	30

Tabela 5.5: Exemplo de itens identificados pelo SonarQube.

Entrada (Etapa 3): Backlog do Produto, Backlog da Sprint, *Definition of Done*.
Saída (Etapa 3): Itens de DT identificados, dados atualizados das tarefas do projeto, Backlog da Sprint atualizado, Incremento.

5.1.4 Etapa 4

Na Retrospectiva da Sprint é avaliada a qualidade e eficiência do trabalho realizado. Nesta etapa são avaliados os itens de DT e discutidos os pontos relacionados à execução das atividades de pagamento da DT e o acúmulo de DT na *sprint*. São registrados os desafios no pagamento da DT, os principais autores dos novos itens de DT e discutidas as principais causas da aquisição de nova DT. O Backlog do Produto, Backlog da Sprint e o *Definition*

of Done são utilizados como referências para a discussão, mas não é realizada nenhuma alteração nesses artefatos. Ao final da discussão são encaminhadas as ações necessárias para o controle da DT para a próxima *sprint* que devem constar no Relatório da DT, que é um artefato que apresenta uma visão da DT geral do projeto, com novos itens, dívida paga, bem como o acúmulo geral. Na Figura 5.2 apresenta-se um exemplo de Relatório de DT.

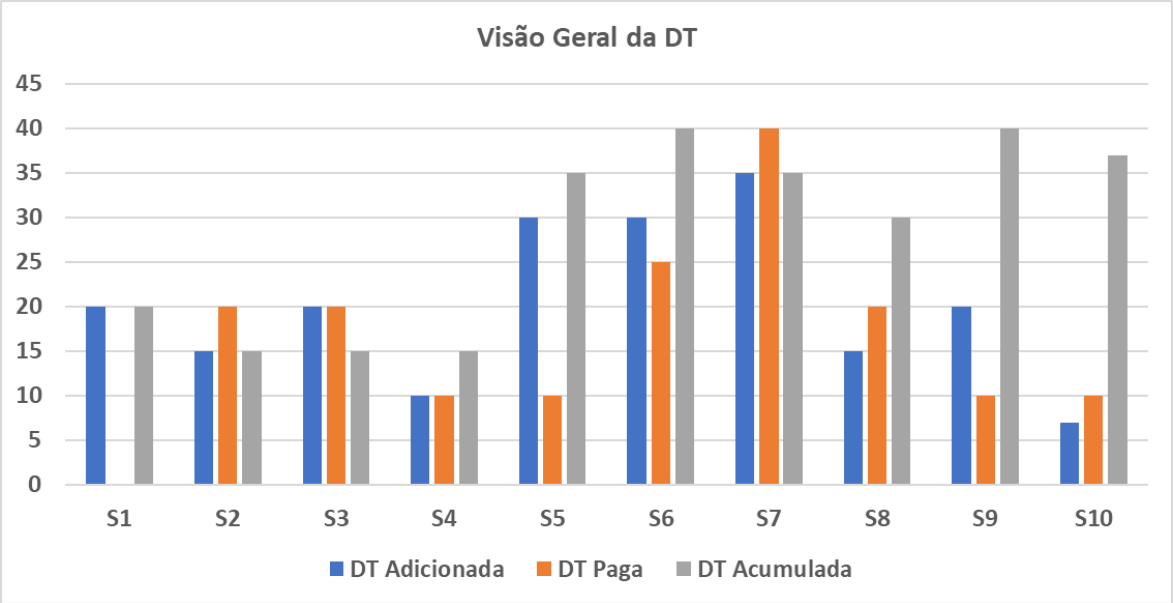

Figura 5.2: Exemplo de Visão Geral da DT incluída no Relatório da DT.

Entrada (Etapa 4): Backlog do Produto, Backlog da Sprint, *Definition of Done*, Itens de DT identificados.
Saída (Etapa 4): Relatório da DT.

5.1.5 Etapa 5

A partir dos dados gerados pela ferramenta de gerenciamento de projetos e dos itens de DT (identificados pelo SonarQube), o Sistema de Recomendação gera uma lista priorizada contendo uma classificação atualizada dos itens de DT identificados no projeto. A recomendação considera os dados históricos sobre o esforço real empenhado no pagamento de itens de DT em projetos similares bem como a experiência do desenvolvedor que incorreu na dívida e que pagará a dívida. A proposta da recomendação é fazer o reuso do conhecimento sobre o pagamento da DT em projetos similares registrados e armazenados na base histórica de projetos para auxiliar na tomada de decisão sobre o pagamento futuro de itens de DT.

A dívida priorizada representa um item que, pela recomendação baseada em projetos anteriores, requer um esforço mais significativo por parte do desenvolvedor inexperiente, apresenta um risco maior (e.g. pela classificação do SonarQube), e que historicamente possui mais episódios de acúmulo durante o progresso do projeto. A Lista Priorizada da DT gerada pelo sistema de recomendação será utilizada para avaliação da DT na próxima *sprint*. Na Tabela 5.6 apresenta-se um exemplo de lista priorizada. Os fatores considerados para a priorização precisam ser reavaliados a cada nova *sprint*. Assim, dependendo do valor dos indicadores, um novo item de DT pode ser priorizado em detrimento de itens que já constaram na Lista Priorizada da DT em *sprints* passadas.

ID	ITEM DT	EXP.	PLAT.	LING.	FRAM.	SEV.	DEBITO
d_3	Código Duplicado	Júnior	WEB	JavaScript	Node.js	Critical	120
d_1	Código Duplicado	Júnior	WEB	JavaScript	React	Major	40
d_5	Código Duplicado	Júnior	WEB	JavaScript	React	Major	30
d_2	Método Longo	Pleno	WEB	JavaScript	Node.js	Major	50
d_4	God Class	Sênior	WEB	JavaScript	React	Minor	40

Tabela 5.6: Exemplo de tabela com os itens de DT priorizados pelo Sistema de Recomendação.

Entrada (Etapa 5): Dados do gerenciamento do projeto, Dados de identificação de DTs.
Saída (Etapa 5): Lista Priorizada da DT.

5.2 Papeis

5.2.1 *Product Owner*

No gerenciamento da DT, o PO é responsável por definir previamente, em conjunto com a equipe, qual é o nível de DT aceitável no projeto. Esse parâmetro deverá ser utilizado como referência durante todo o desenvolvimento do projeto.

5.2.2 *Scrum Master*

Nas atividades do gerenciamento de DT o *Scrum Maste*r é o responsável por definir as atividades de pagamento da DT durante as reuniões de *sprint planning*. Além disso, também é responsável por realizar o monitoramento da DT para garantir que o projeto esteja nos parâmetros aceitáveis de acúmulo de DT.

5.2.3 Desenvolvedores

No contexto de DT, os desenvolvedores são os principais responsáveis pelo acúmulo de DT no projeto, seja intencional ou não-intencional. Entretanto, eles também são os principais responsáveis pelo pagamento da DT, principalmente daquelas relacionadas ao código, testes, projeto e arquitetura.

5.3 Eventos

5.3.1 *Sprint*

No contexto da proposta, durante a *sprint* também são realizadas as atividades de identificação, monitoramento, priorização, e pagamento de DT com o apoio das ferramentas de gerenciamento de projetos e as ferramentas de análise estática de código.

5.3.2 *Sprint Planning*

Neste ponto do fluxo do Scrum, pela nossa proposta, também serão discutidos os itens de DT que devem ser pagos dentro da *sprint*. O pagamento da dívida é descrito como uma tarefa de refatoramento que dever ser atribuída, estimada e incluída no Backlog da Sprint. Os itens de DT estarão registrados e priorizados em um artefato denominado Lista Priorizada da DT gerada pelo sistema de recomendação proposto com base nos itens de DT identificados nas *sprints* anteriores (e.g. pelo SonarQube). Após incluído no Backlog da Sprint, o pagamento da dívida será tratado como as outras atividades da *sprint*.

5.3.3 Retrospectiva da Sprint

A Retrospectiva da Sprint é um ponto ideal no fluxo do Scrum para avaliar e discutir o acúmulo da DT. A equipe deverá discutir o acúmulo analisando os relatórios das ferramentas de análise estática de código e a lista de novos itens identificados. Caso o acúmulo de DT tenha ultrapassado os parâmetros preestabelecidos será necessário incluir mais atividades de pagamento de dívida na próxima *sprint* para que o índice de acúmulo volte aos parâmetros toleráveis.

5.4 Artefatos

5.4.1 Lista Priorizada da DT

Lista priorizada dos itens de DT que foram identificados em *sprints* anteriores. É um artefato gerado automaticamente pelo sistema de recomendação apresentado no Capítulo 6. A partir deste artefato a equipe Scrum definirá na Sprint Planning quais itens serão adicionados ao Backlog da Sprint para pagamento.

5.4.2 Relatório da DT

Um artefato gerado após a Retrospectiva da Sprint que contem uma visão geral sobre a dívida adicionada e a dívida acumulada, bem como uma visão da dívida por desenvolvedor. Além disso, o relatório apresenta uma lista de principais causas da DT e os principais desafios que levam à equipe incorrer em DT, ou dificultam o pagamento da DT.

5.5 Considerações Finais do Capítulo

Os estudos no capítulo anterior identificaram as dificuldades dos desenvolvedores em realizar as atividades do gerenciamento de DT. Com o objetivo de auxiliar na inclusão de tais atividades no dia a dia das equipes propomos um modelo de processo adaptado do Scrum incluindo o gerenciamento de DT. A ideia é incluir as atividades no fluxo que os desenvolvedores já realizam diariamente para facilitar a sua adaptação. O Scrum foi escolhido com base no *14th Annual State of Agile Report* [22] que entrevistou 1121 profissionais e aponta

que o Scrum é o *framework* ágil utilizado por 75% dos respondentes. No próximo capítulo é discutido o sistema de recomendação que gera a Lista de DT Priorizada.

Capítulo 6

Modelo de Sistema de Recomendação para a Priorização de DT

Neste capítulo apresenta-se o modelo de sistema de recomendação para auxiliar na tomada de decisão sobre a priorização da DT (Objetivo Específico 3). A solução reutiliza o conhecimento registrado em dados históricos de projetos anteriores. Especificamente, a solução avalia o débito, a severidade, e o acúmulo de DT em projetos anteriores para recomendar a priorização de itens de DT no projeto atual.

6.1 O Problema de Recomendação para Priorização

O desenvolvimento de software envolve uma grande quantidade de informações de diferentes tipos. Muitas vezes é difícil para o desenvolvedor encontrar a informação necessária para alcançar seus objetivos no desenvolvimento, sem interromper seu fluxo de trabalho. Como o processo envolve muitas informações, há uma sobrecarga que dificulta a tomada de decisão em diferentes atividades como: escolha de tecnologias, definição de requisitos não-funcionais, testes e gerenciamento da DT. Neste contexto, para facilitar o processo de desenvolvimento, os sistemas de recomendação estão sendo aplicados na Engenharia de Software [72; 78]. No contexto deste trabalho a recomendação é direcionada para a priorização da DT.

O objetivo específico da abordagem é auxiliar na tomada de decisão sobre a inclusão de atividades de refatoramento no *backlog* das *sprints*. As ferramentas de apoio ao desenvol-

vimento de software fornecem métricas do projeto, como tecnologias envolvidas e itens de dívida técnica identificados. Entretanto, a decisão muitas vezes é subjetiva, assim a proposta utiliza dados históricos para avaliar métricas de itens de DT identificados no passado para recomendar a priorização. A referência para a recuperação dos dados são os itens de DT identificados no projeto atual. Então, a partir destas informações o sistema gera uma lista priorizada de itens de DT identificados no projeto atual, baseada em uma recomendação de predição de nota (rating prediction [61]).

Mais formalmente, seja $D = \{d_1, d_2, d_3, ..., d_n\}$ o conjunto de todos os itens de DT do projeto atual, e seja $I = \{i_1, i_2, i_3, ..., i_m\}$ o conjunto de todos os itens de DT registrados na base histórica de projetos de uma organização e o item d_a o alvo da recomendação, onde $d_a \notin I$. Temos que cada item i_k possui um conjunto de métricas associadas $M_k \subset M$ e uma similaridade $sim(d_a, i_k) \in (0, 1)$. Assim, nossa proposta consiste em encontrar um subconjunto de itens que possuam maior similaridade com d_a e calcular com base no subconjunto de métricas $M_k = \{m_1, m_2, m_3, ..., m_t\}$ associadas a cada item i_k um $score$ para uma função utilidade $u(d_a, i_k)$ que avalia a prioridade do item d_a.

Para o desenvolvimento adequado de um sistema de recomendação é necessário definir algumas especificações como: o alvo da recomendação; o problema que o recomendador resolve; a resposta oferecida; e o valor da sugestão. Assim, podemos definir o sistema proposto nos seguintes pontos:

- O alvo da recomendação é a equipe Scrum que precisa controlar a dívida técnica;
- O problema a ser resolvido é a priorização da dívida técnica;
- A solução oferecida é uma sugestão formada por uma lista de itens de DT priorizados;
- A recomendação busca auxiliar a tomada de decisão da equipe quanto a priorização de DT.

6.2 Extração de Dados

Para funcionamento do sistema de recomendação são utilizados dados extraídos de ferramentas de análise estática e ferramentas de gerenciamento de projetos, como: vulnerabilidades

no código, cobertura de testes, débito associado, severidade do débito, tecnologias, equipe, entre outras. As ferramentas de análise estática são ferramentas como SonarQube, SonarGraph, Klockwork, utilizadas para identificação de DT a partir de métricas relacionadas ao código-fonte. As ferramentas de gerenciamento de projetos são utilizadas para controle do projeto e registram informações úteis sobre o contexto e progresso do projeto.

Para exemplificar, no trabalho realizado por Lenarduzzi *et al.* [44] foi construída uma base com dados de DT de 33 projetos *open-source* Java da *Apache Software Foundation*, coletados pelo SonarQube. A ferramenta disponibiliza os seguintes dados: data da criação; data de fechamento; status geral; status da resolução; caminho do código-fonte com o problema; nível de severidade; débito associado; autor do *commit*. Este trabalho serviu de referência para identificação de quais informações podem ser coletadas a partir do SonarQube e outras ferramentas. A seguir são apresentados dois exemplos de *code smells* identificados pelo SonarQube: *long method*; *lazy class*.

Exemplo 01 - *Long Method*

É um problema que ocorre quando um método é escrito usando um número grande de linhas de forma desnecessária, prejudicando a leitura e, consequentemente, a manutenção do código. As Figuras 6.1 e 6.2 apresentam os dados coletados para a DT.

long_method.creationDate <chr>	long_method.closeDate <chr>	long_method.status <chr>	long_method.resolution <chr>
2003-09-04T23:28:19Z	2003-09-27T15:45:02Z	CLOSED	FIXED
2003-09-04T23:28:19Z	2010-02-18T08:46:51Z	CLOSED	FIXED
2004-02-27T07:57:51Z	2010-03-19T16:55:14Z	CLOSED	FIXED
2004-02-27T07:57:51Z	2010-09-24T11:28:45Z	CLOSED	FIXED
2004-02-27T07:57:51Z	2010-10-17T11:40:21Z	CLOSED	FIXED
2010-03-19T16:55:14Z	2010-03-27T14:14:45Z	CLOSED	FIXED
2010-03-27T14:14:45Z	2010-10-16T15:43:32Z	CLOSED	FIXED
2010-10-15T15:32:40Z		OPEN	null
2010-10-15T19:20:55Z		OPEN	null

Figura 6.1: Exemplo 01 - datas de criação, fechamento, status, e resolução.

long_method.component <chr>	long_method.severity <chr>	long_method.debt <chr>	long_method.author <chr>
org.apache.deamon:src/java/org/apache/commons/daemon/support/DaemonLoader.java	MAJOR	1h30min	yoavs@apache.org
org.apache.deamon:src/java/org/apache/commons/daemon/DaemonPermission.java	MAJOR	1h30min	yoavs@apache.org
org.apache.deamon:src/java/org/apache/commons/daemon/support/DaemonLoader.java	MAJOR	1h30min	jfclere@apache.org
org.apache.deamon:src/java/org/apache/commons/daemon/DaemonPermission.java	MAJOR	1h30min	jfclere@apache.org
org.apache.deamon:src/java/org/apache/commons/daemon/support/DaemonLoader.java	MAJOR	1h30min	jfclere@apache.org
org.apache.daemon:src/samples/ProcrunService.java	MAJOR	1h30min	sebb@apache.org
org.apache.daemon:src/samples/ProcrunService.java	MAJOR	1h30min	sebb@apache.org
org.apache.daemon:src/main/java/org/apache/commons/daemon/support/DaemonConfiguration.java	MAJOR	1h30min	mturk@apache.org
org.apache.daemon:src/main/java/org/apache/commons/daemon/support/DaemonWrapper.java	MAJOR	1h30min	mturk@apache.org

Figura 6.2: Exemplo 01 - caminho do código-fonte, severidade, débito, autor

Exemplo 02 - *Lazy Class*

É um item de dívida técnica que representa classes que possuem poucas responsabilidades. Essas classes aumentam o código, de forma desnecessária, dificultando a manutenção. As Figuras 6.3 e 6.4 apresentam os dados coletados para *Lazy Class*.

lazy_class.creationDate <chr>	lazy_class.closeDate <chr>	lazy_class.status <chr>	lazy_class.resolution <chr>
2003-09-04T23:28:19Z	2004-11-05T12:08:10Z	CLOSED	FIXED
2003-09-04T23:28:19Z	2004-11-05T12:08:10Z	CLOSED	FIXED
2004-02-27T07:57:51Z	2010-10-16T14:58:39Z	CLOSED	FIXED

Figura 6.3: Exemplo 02 - datas de criação, fechamento, status, e resolução.

lazy_class.resolution <chr>	lazy_class.component <chr>	lazy_class.severity <chr>	lazy_class.debt <chr>	lazy_class.author <chr>
FIXED	org.apache:daemon:src/samples/ServiceDaemonReadThread.java	MAJOR	1h30min	yoavs@apache.org
FIXED	org.apache:deamon:src/native/nt/procrun/java/Test.java	MAJOR	1h30min	yoavs@apache.org
FIXED	org.apache:daemon:src/samples/ServiceDaemonReadThread.java	MAJOR	1h30min	jfclere@apache.org

Figura 6.4: Exemplo 02 - caminho do código-fonte, severidade, débito, autor

O acúmulo da dívida técnica pode ser avaliado realizando uma comparação entre o número de itens de DT identificados e o número de itens resolvidos ao longo do tempo. Na Figura 6.5, apresenta-se o gráfico da DT acumulada do projeto *Apache Commons*.

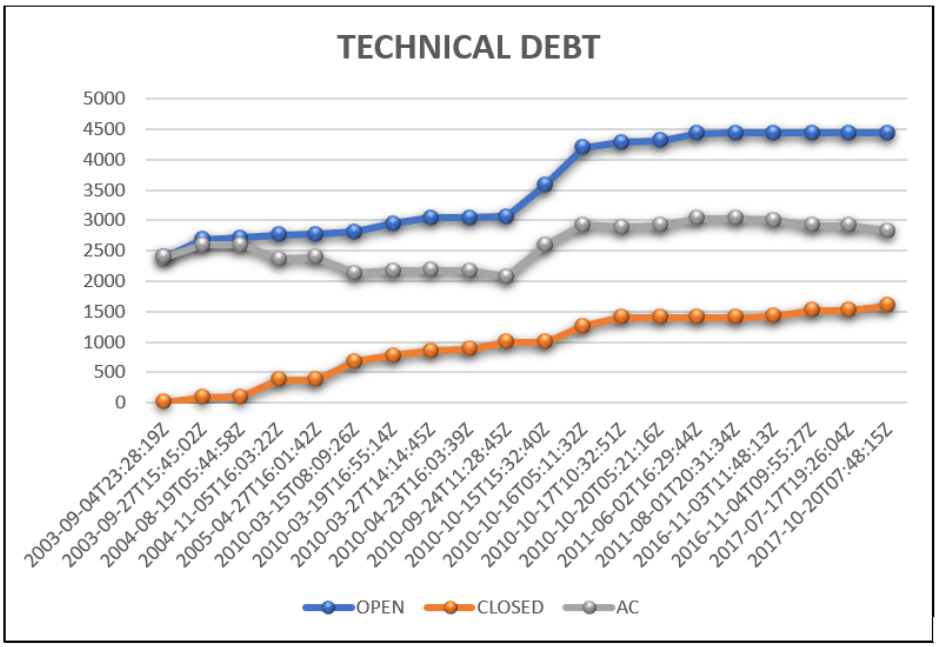

Figura 6.5: Acúmulo da DT ao longo do tempo.

Para avaliar a aquisição e resolução de DT por desenvolvedor é possível comparar quantos itens foram adicionados e quantos foram solucionados ao longo do tempo. Na Figura 6.6 apresenta-se a proporção de itens identificados nos *commits* de cada desenvolvedor e os itens de DT que foram solucionados.

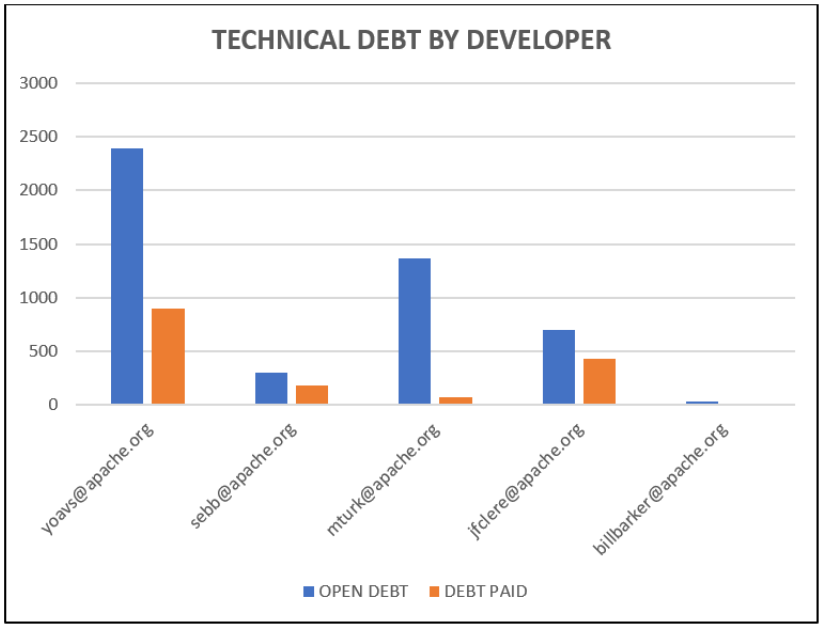

Figura 6.6: Relação de DT adquirida e paga por desenvolvedor.

Os Critérios para Priorização

Os critérios para priorização de DT identificados na literatura, como em Lenarduzzi *et al.* [42], são classificados em categorias como: negócio [51; 77], cliente [49; 25], manutenção [80; 74], qualidade [20; 70], etc. Nas pesquisas, cada critério é direcionado para priorização de tipos específicos de DT como DT de projeto, DT de código, e DT de requisitos. Entretanto, trabalhos secundários [42; 45; 6; 63; 12], e a experiência dos profissionais discutidos nos resultados de surveys [64; 48], indicam que as equipes tendem a direcionar os esforços para análises de custo-benefício e manutenibilidade. Adicionalmente, consideramos as métricas disponibilizadas por ferramentas de análise estática de código, como a estimativa do débito e a severidade da DT. A outra fonte de dados utilizada são dados disponibilizados por ferramentas de gerenciamento de projetos. Dados como as tecnologias envolvidas, domínio da aplicação, e dados sobre a equipe. A seguir são listados os critérios

considerados para realizar a priorização:

- Débito – a estimativa do esforço associado à correção de um item de dívida técnica. Este esforço é representado em horas;
- Severidade – classificação do possível impacto que o débito pode gerar ao projeto;
- Tempo de conclusão – a dívida acumulada do projeto por itens de DT identificados;
- Experiência do Desenvolvedor – nível de experiência do desenvolvedor: Júnior, Pleno, Sênior.

6.3 O Recomendador

O sistema de recomendação proposto utiliza os dados históricos para avaliar e recomendar a priorização de itens de DT identificados no projeto atual baseado em projetos anteriores semelhantes. As métricas selecionadas para avaliar a priorização do item de DT são: débito, severidade, tempo de conclusão, e a experiência do desenvolvedor. A abordagem considera que o desenvolvedor que paga a dívida é o mesmo que incorreu na dívida.

Em sistemas de recomendação esta abordagem é chamada de Filtragem Colaborativa baseada em Memória [21]. Nesta abordagem o algoritmo busca em um conjunto de dados os itens com maior similaridade com o perfil do item alvo. O método utilizado para a recomendação é o *rating prediction*, um tipo de recomendação que tenta prever a avaliação do usuário para aqueles itens que não foram avaliados ainda por ele. As predições são computadas a partir de *feedbacks* explícitos de usuários, ou podem ser obtidos por avaliações de itens no passado [61]. As caraterísticas dos itens para o cálculo da similaridade são apresentadas na Tabela 6.1. Esta representação dos perfis é realizada após um pré-processamento dos dados coletados a partir de ferramentas de análise estática de código, e da ferramenta de gerenciamento de projeto.

Para realizar a identificação dos itens similares será realizada uma pré-filtragem por cada tipo de DT identificado em *sprints* anteriores. Também será utilizado método KNN (*k-Nearest-Neighbours*) [32], que retorna os k vizinhos mais próximos de um elemento de entrada. Para cada item de DT existe uma lista de métricas associadas catalogadas como mostrado na Tabela 6.2.

ID	ITEM DT	PLAT.	LING.	FRAM.	SEV.	DEBITO
i_1	Código Duplicado	WEB	PHP	Laravel	Major	50
i_2	Código Duplicado	DESKTOP	JAVA	Spring	Critical	120
i_3	Código Duplicado	WEB	PHP	Laravel	Minor	40
i_4	Código Duplicado	MOBILE	JavaScript	React	Major	30

Tabela 6.1: Exemplo de perfis coletados de projetos catalogados.

ID	Item DT	Débito	Nº de linhas	Severidade	Conclusão	Experiência
i_1	Código Duplicado	50	70	Major	30	Pleno
i_2	God Class	120	345	Critical	60	Sênior
i_3	Método Longo	40	52	Minor	20	Júnior
i_4	Código Duplicado	30	28	Major	30	Júnior

Tabela 6.2: Exemplo de métricas coletadas para cada item de DT na base histórica.

Para analisar a similaridade dos itens de DT e encontrar os vizinhos, o sistema gera um vetor de características para cada item. As características são extraídas do item de DT alvo. A matriz com os vetores de características permite calcular a distância e assim a similaridade entre os itens. A representação em vetores de características é exemplificada na Tabela 6.3.

ID	PLAT.	LING.	FRAM.	SEV.	DEBITO
d_1	1	1	1	1	1
i_1	1	0	0	0	0
i_2	0	0	0	0	0
i_3	1	0	0	0	0
i_4	0	1	1	1	0

Tabela 6.3: Exemplo de matriz utilizada para o cálculo da similaridade.

A partir dos vetores de características é calculada a distância utilizando como métrica a distância de Hamming (Equação 6.1). A distância de Hamming [75] é uma métrica utilizada para comparar dois vetores binários com comprimento fixo. Esta métrica de distância é utilizada para medir a distância entre variáveis categóricas. Se p e q são vetores binários consistindo de zeros e uns. A distância Hamming é o número de posições onde o padrão de

bits é diferente. Por exemplo, sejam duas *strings* 1100 e 1001, para $1100 \oplus 1001 = 0101$, distância de Hamming d(1100,1001)=2. Em seu trabalho Janett *et al.* [75] realizaram um comparativo entre métricas de distância para funções de proximidade de vizinhos (*Nearest Neighbors*). O trabalho indica que a distância de Hamming é mais adequada para cenários onde o tamanho do vetor é fixo e os vetores são binários. As outras funções de distância como a Euclideana e a de Manhattan devem ser utilizadas em situações com vetores n-dimensionais não fixos [75]. Com a distância entre os vetores de características é calculada a similaridade com a Equação 6.2.

$$HAM(d_a, i_k) = \sum_{t=1}^{n} |d_a - i_k| \tag{6.1}$$

$$sim(d_a, i_k) = 1 - \frac{HAM(d_a, i_k)}{m} \tag{6.2}$$

Para avaliar o $score$ de um determinado item DT histórico é utilizada a Equação 6.3, onde $debta_i$ é o débito acumulado de i_k, esf_i o esforço associado o item i_k na base histórica, sev_{i_k} a severidade do item na base histórica, e exd_{d_a} a experiência do desenvolvedor que incorreu no item de DT atual. Para o cálculo serão considerados os valores 1, 2 e 3 para representar as categorias de desenvolvedores júniores, pleno e sênior, respectivamente. Como também serão considerados os valores 1, 2, 3 e 4 para representar a severidade do débito minor, major, critical e blocker, respectivamente. Calculado o $score$ e a similaridade é calculada a utilidade da recomendação definida pela Equação 6.4 que representa a média do score dos itens, ponderada pela similaridade.

$$score(d_a, i_k) = \frac{debta_{i_k} * esf_{i_k} * sev_{i_k}}{exd_{d_a}} \tag{6.3}$$

$$u_{d_a,i_k} = \frac{1}{m} * \sum_{i_k \in I} sim(d_a, i_k) \times score(d_a, i_k) \tag{6.4}$$

O sistema recomenda uma avaliação que é utilizada para priorizar os itens de DT identificados no projeto atual. Formalizando, seja I o conjunto de todos os itens de dívida técnica que podem ser recomendados. Seja $u_{d_a,p}$ a função de utilidade de um item de DT d_a para o projeto atual P. Então, $u : D \times P \rightarrow L$, onde L é o conjunto ordenado de itens. Para cada item $d_a \in D$, o sistema recomenda a avaliação L_a que maximize a utilidade do item

d_a. Mais formalmente, o problema pode ser representado pelas Equações 6.5 adaptado de Adomavicius *et al.* [1].

$$\forall d_a \in D, u_{d_a,p} = argmax\ u(d_a, i) \tag{6.5}$$

A priorização será reavaliada após cada *sprint*. No início de cada *sprint*, durante a reunião de *sprint planning* o sistema de recomendação será acionado e deverá realizar uma nova avaliação dos itens identificados. Assim, a cada nova *sprint* a Lista Priorizada da DT será atualizada.

6.4 Exemplo de uso

Considere um projeto A desenvolvido para plataforma *web* em JavaScript utilizando os *frameworks* React e Node.js. Então considere que após a execução da primeira *sprint* o SonarQube identificou um conjunto de DTs $D = \{d_1, d_2, d_3, d_4, d_5\}$. Os itens são listados na Tabela 6.4.

ID	ITEM DT	EXP.	PLAT.	LING.	FRAM.	SEV.	DEBITO
d_1	Código Duplicado	Júnior	WEB	JavaScript	React	Major	40
d_2	Método Longo	Pleno	WEB	JavaScript	Node.js	Major	50
d_3	Código Duplicado	Júnior	WEB	JavaScript	Node.js	Critical	120
d_4	God Class	Sênior	WEB	JavaScript	React	Minor	40
d_5	Código Duplicado	Júnior	WEB	JavaScript	React	Major	30

Tabela 6.4: Itens de DT identificados.

O processo de recomendação possui duas etapas principais: a identificação de itens de DT similares na base de dados; o cálculo de *score* do item d_a para cada item i_k.

Então para cada dívida d_a identificada no projeto, utilizando o algoritmo kNN, o sistema de recomendação deve buscar nos dados históricos informações sobre projetos similares com a mesma dívida. Assim, considere para o item d_1 do nosso exemplo que os dados históricos retornados após a filtragem por categoria da DT (ou seja, "Código Duplicado") são listados na Tabela 6.5.

ID	ITEM DT	PLAT.	LING.	FRAM.	SEV.	DEBITO
i_1	Código Duplicado	WEB	PHP	Laravel	Major	50
i_2	Código Duplicado	DESKTOP	JAVA	Spring	Critical	80
i_3	Código Duplicado	WEB	PHP	Laravel	Minor	60
i_4	Código Duplicado	MOBILE	JavaScript	React	Major	30

Tabela 6.5: Itens de DT recuperados da base de dados.

Após a pré-filtragem dos dados históricos de projetos similares, é então calculada a distância entre a DT histórica retornada e d_1 considerando a representação vetorial com dados binários da Tabela 6.6. Para cada item i_k é calculada a distância de Hamming para a DT alvo d_1. Que é definida pela Equação 6.1. Fazendo o cálculo para i_1 temos $11111 \oplus 10000 = 01111$, assim a distância de Hamming $HAM(d_a, i_1) = 4$. O cálculo da distância para cada item i_k é apresentado na Tabela 6.7

ID	PLAT.	LING.	FRAM.	SEV.	DEBITO
d_1	1	1	1	1	1
i_1	1	0	0	0	0
i_2	0	0	0	0	0
i_3	1	0	0	0	0
i_4	0	1	1	1	0

Tabela 6.6: Representação vetorial dos dados coletados.

.

ID	PLAT.	LING.	FRAM.	SEV.	DEBITO	DIST.
d_1	1	1	1	1	1	
i_1	1	0	0	0	0	4
i_2	0	0	0	0	0	5
i_3	1	0	0	0	0	4
i_4	0	1	1	1	0	2

Tabela 6.7: Representação vetorial com o cálculo da distância Hamming para cada item i_k.

Após o cálculo da distância para cada item i_k é possível calcular a similaridade

$sim(d_a, i_k)$. Considerando o item i_1, o cálculo da similaridade é apresentado na Equação 6.6. Os dados com o cálculo da similaridade de cada item são apresentados na Tabela 6.8.

$$sim(d_1, i_1) = 1 - \frac{4}{5} = 0,2 \tag{6.6}$$

ID	PLAT.	LING.	FRAM.	SEV.	DEBITO	DIST.	SIM.
d_1	1	1	1	1	1		
i_1	1	0	0	0	0	4	0,2
i_2	0	0	0	0	0	5	0
i_3	1	0	0	0	0	4	0,2
i_4	0	1	1	1	0	2	0,6

Tabela 6.8: Dados dos itens com o cálculo da similaridade.

Para cada item similar i_k é calculado o score referente a DT alvo d_a. Para o nosso exemplo o cálculo da função $score(d_1, i_1)$ é definido pela Equação 6.7. A Tabela 6.9 apresenta o cálculo do $score$ para cada item i_k do nosso exemplo.

$$score(d_1, i_1) = \frac{12 * 30 * 2}{1} = 720 \tag{6.7}$$

ID	SEV.	ESF.	ACUM.	SCORE
i_1	2	30	12	720
i_2	3	60	5	900
i_3	1	20	15	300
i_4	2	30	22	1320

Tabela 6.9: Cálculo do $score$ para cada item.

Por fim, a utilidade geral é calculada sobre a média, ponderada pela similaridade, da utilidade $u(d_a, i_k)$. Este procedimento será realizado para cada item d_a do projeto A. O cálculo da utilidade d_1 é apresentado na Equação 6.8. A Tabela 6.10 lista todos os itens d_a com sua respectiva pontuação de utilidade.

$$u(d_1, i_k) = \frac{(0,2 * 720) + (0 * 900) + (0,2 * 300) + (0,6 * 1320)}{4} = 249 \tag{6.8}$$

ID	ITEM DT	EXP.	PLAT.	LING.	FRAM.	SEV.	DEBITO	UTIL.
d_1	Código Duplicado	Júnior	WEB	JavaScript	React	Major	40	249
d_2	Método Longo	Pleno	WEB	JavaScript	Node.js	Major	50	144
d_3	Código Duplicado	Júnior	WEB	JavaScript	Node.js	Critical	120	347
d_4	God Class	Sênior	WEB	JavaScript	React	Minor	40	89
d_5	Código Duplicado	Júnior	WEB	JavaScript	React	Major	30	211

Tabela 6.10: Lista de itens de DT identificados com suas respectivas pontuações.

ID	ITEM DT	EXP.	PLAT.	LING.	FRAM.	SEV.	DEBITO	UTIL.
d_3	Código Duplicado	Júnior	WEB	JavaScript	Node.js	Critical	120	347
d_1	Código Duplicado	Júnior	WEB	JavaScript	React	Major	40	249
d_5	Código Duplicado	Júnior	WEB	JavaScript	React	Major	30	211
d_2	Método Longo	Pleno	WEB	JavaScript	Node.js	Major	50	144
d_4	God Class	Sênior	WEB	JavaScript	React	Minor	40	89

Tabela 6.11: Tabela com os itens de DT priorizados com base no score.

Ao final do processo o sistema de recomendação gera a lista de itens de DT priorizada a partir da utilidade. A lista ordenada do conjunto D de itens de DT do projeto A é apresentada na Tabela 6.11

6.5 Considerações Finais do Capítulo

A solução proposta envolve um modelo de processo baseado no arcabouço do Scrum e um sistema de recomendação que usa os dados coletados por ferramentas de gerenciamento de projetos e ferramentas de análise estática de código. No próximo capítulo são apresentados os direcionamentos da pesquisa para os próximos passos e o cronograma das atividades que ainda devem ser executadas.

Capítulo 7

Estado Atual e Cronograma de Conclusão

Neste capítulo é apresentado o estado atual da pesquisa e o planejamento das atividades previstas para a conclusão. Em resumo as próximas atividades visam a validação dos modelos, implementação e avaliação da solução proposta.

7.1 Estado Atual da Pesquisa

Segundo o que foi estabelecido nos objetivos específicos da pesquisa, o trabalho cumpriu com os objetivos específicos OE1, OE2, OE3:

- OE1 – Investigar quais os desafios enfrentados por profissionais na priorização de DT;
- OE2 – Propor um modelo para gerenciamento de dívida técnica integrado ao arcabouço Scrum;
- OE3 – Modelar um sistema de recomendação e priorização de tarefas para pagamento de dívida técnica;

Para as questões RQ01,RQ02, RQ03, relacionadas aos desafios enfrentados (OE1), através de revisão da literatura e resultados de *surveys* apresentados no Capítulo 4, encontramos que:

- Os desenvolvedores possuem um bom entendimento sobre o conceito de DT e suas principais causas e efeitos (RQ1);

- Apesar de reconhecer a importância, os desenvolvedores encontram dificuldades em realizar as atividades do gerenciamento da DT. As atividades, no geral, quando são realizadas, são executadas sem um processo bem definido (RQ2);

- O acúmulo de DT no projeto é inversamente proporcional à maturidade do desenvolvedor (RQ3).

Sobre a questão de pesquisa RQ04, propor um modelo para o gerenciamento da dívida técnica no contexto ágil (OE2), foi proposto no Capítulo 5 um modelo de processo adaptado do Scrum contemplando as atividades e responsabilidades necessárias para o gerenciamento da dívida técnica.

No Capítulo 6 foi apresentada a proposta do modelo de recomendação de priorização de dívida técnica (RQ06)(OE4), onde listamos que os critérios selecionados para a priorização foram (RQ05):

- Débito – a estimativa do esforço associado à correção de um item de dívida técnica. Este esforço é representado em horas.

- Severidade – o impacto que um determinado tipo de *code smell* pode causar ao projeto. A severidade pode ser menor (MINOR), maior (MAJOR), crítica (CRITICAL) e bloqueante (BLOCKER);

- Tempo de conclusão – a dívida acumulada do projeto por itens de DT identificados;

- Experiência do Desenvolvedor – nível de experiência do desenvolvedor.

Sobre as hipóteses levantadas na problemática do trabalho:

- H1. A experiência do desenvolvedor impacta diretamente no acúmulo da dívida técnica em nível de código.

- H2. É possível reduzir o acúmulo da dívida técnica considerando um gerenciamento explícito dentro do processo ágil utilizando uma lista priorizada de dívida técnica.

A literatura apresenta estudos com indícios de que a H1 é verdadeira. Porém, ainda realizaremos experimentos e avaliações para apresentar uma conclusão. Ainda são necessários testes e experimentos que estão previstos nas atividades futuras para conclusão da pesquisa e validação das hipóteses H1 e H2.

7.2 Próximos Passos

As próximas etapas do trabalho envolvem a construção de uma base de dados históricos de projetos no contexto ágil para viabilizar a validação, implementação e avaliação da solução proposta.

Objetivo Específico (OE4)

Para o Objetivo Específico (OE4) tem-se uma questão para direcionamento do reuso de conhecimento a ser utilizado na solução:

- RQ07 - Quais dados históricos são necessários para construção de uma Base de Dados de DT, para viabilizar o reuso de conhecimento para priorização de novos itens?

A construção da base permitirá a validação do trabalho e a pesquisa será direcionada para a implementação das ferramentas para apoiar os desenvolvedores aplicando a solução proposta.

Objetivo Específico (OE5)

Para o Objetivo Específico (OE5) há uma questão para direcionamento da solução:

- RQ08 - Qual a utilidade prática da solução proposta?

Para validação da recomendação será realizado um estudo de caso, além de uma validação apoiada pela opinião de especialistas, comparando a recomendação gerada pelo sistema e a escolhas feitas por profissionais para exemplos de itens DT identificados. Enquanto a avaliação da ferramenta de apoio será realizada com aplicação do modelo TAM [36]. Um mecanismo para avaliar a aceitação de tecnologias dentro de um contexto específico na perspectiva do usuário.

Cronograma para conclusão

Considerando as atividades listadas na Tabela 7.1, a Tabela 7.2 apresenta o cronograma de conclusão com defesa da Tese prevista para 20 de Fevereiro de 2022.

Atividades	**Descrição**
A1	Construção de uma base dados em DT
A2	Validação do modelo de processo
A3	Validação do sistema de recomendação.
A4	Implementação da ferramenta de apoio ao modelo de priorização
A5	Avaliação da ferramenta de apoio
A6	Escrever artigo
A7	Escrever documento da Tese
A8	Defesa da Tese

Tabela 7.1: Lista de atividades para 2021

Atividades	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**
Fevereiro/2021	X							
Março/2021	X	X				X		
Abril/2021			X	X		X		
Maio/2021				X	X	X		
Junho/2021					X	X		
Julho/2021					X	X	X	
Agosto/2021						X	X	
Setembro/2021						X	X	
Outubro/2021							X	
Novembro/2021							X	
Dezembro/2021							X	
Janeiro/2022							X	
Fevereiro/2022								X

Tabela 7.2: Cronograma de atividades para 2021

As atividades de validação dos modelos estão concentradas no primeiro semestre para reservar uma janela de tempo para publicação de artigos e escrita da Tese.

Bibliografia

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[2] Shirin Akbarinasaji. Toward measuring defect debt and developing a recommender system for their prioritization. In *Proceedings of the 13th International Doctoral Symposium on Empirical Software Engineering*, pages 15–20, 2015.

[3] Shirin Akbarinasaji, Ayse Bener, and Adam Neal. A heuristic for estimating the impact of lingering defects: can debt analogy be used as a metric? In *2017 IEEE/ACM 8th Workshop on Emerging Trends in Software Metrics (WETSoM)*, pages 36–42. IEEE, 2017.

[4] Samar Al-Saqqa, Samer Sawalha, and Hiba AbdelNabi. Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, 14(11), 2020.

[5] Reem Alfayez, Pooyan Behnamghader, Kamonphop Srisopha, and Barry Boehm. An exploratory study on the influence of developers in technical debt. In *Proceedings of the 2018 International Conference on Technical Debt*, pages 1–10, 2018.

[6] Reem Alfayez and Barry Boehm. Technical debt prioritization: A search-based approach. In *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, pages 434–445. IEEE, 2019.

[7] Theodoros Amanatidis, Alexander Chatzigeorgiou, Apostolos Ampatzoglou, and Ioannis Stamelos. Who is producing more technical debt? a personalized assessment of td principal. In *Proceedings of the XP2017 Scientific Workshops*, pages 1–8, 2017.

[8] Vard Antinyan, Miroslaw Staron, Wilhelm Meding, Per Österström, Erik Wikstrom, Johan Wranker, Anders Henriksson, and Jörgen Hansson. Identifying risky areas of software code in agile/lean software development: An industrial experience report. In *2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, pages 154–163. IEEE, 2014.

[9] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman. Managing technical debt in software engineering (dagstuhl seminar 16162). In *Dagstuhl Reports*, volume 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[10] Raja Bavani. 10 principles for success in distributed agile delivery. *Cutter IT J. LLC*, 26(11):30, 2013.

[11] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001.

[12] Woubshet Nema Behutiye, Pilar Rodríguez, Markku Oivo, and Ayşe Tosun. Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Information and Software Technology*, 82:139–158, 2017.

[13] Barry W Boehm, John R Brown, and Mlity Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering*, pages 592–605, 1976.

[14] Robin Burke. Knowledge-based recommender systems. *Encyclopedia of library and information systems*, 69(Supplement 32):175–186, 2000.

[15] Robin Burke and Maryam Ramezani. Matching recommendation technologies and domains. In *Recommender systems handbook*, pages 367–386. Springer, 2011.

[16] Gloria Chatzopoulou, Magdalini Eirinaki, and Neoklis Polyzotis. Query recommendations for interactive database exploration. In *International Conference on Scientific and Statistical Database Management*, pages 3–18. Springer, 2009.

[17] Zadia Codabux and Byron J Williams. Technical debt prioritization using predictive analytics. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pages 704–706. IEEE, 2016.

[18] Davor Cubranic, Gail C Murphy, Janice Singer, and Kellogg S Booth. Hipikat: A project memory for software development. *IEEE Transactions on Software Engineering*, 31(6):446–465, 2005.

[19] Ward Cunningham. The wycash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30, 1992.

[20] Bill Curtis, Jay Sappidi, and Alexandra Szynkarski. Estimating the principal of an application's technical debt. *IEEE software*, 29(6):34–42, 2012.

[21] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.

[22] Digital.ai. 14th annual state of agile report, May 2020.

[23] John Erickson, Kalle Lyytinen, and Keng Siau. Agile modeling, agile software development, and extreme programming: the state of research. *Journal of Database Management (JDM)*, 16(4):88–100, 2005.

[24] Jon Eyolfson, Lin Tan, and Patrick Lam. Correlations between bugginess and time-based commit characteristics. *Empirical Software Engineering*, 19(4):1009–1039, 2014.

[25] Davide Falessi and Alexander Voegele. Validating and prioritizing quality rules for managing technical debt: An industrial case study. In *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, pages 41–48. IEEE, 2015.

[26] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Zanker. An integrated environment for the development of knowledge-based recommender applications. *International Journal of Electronic Commerce*, 11(2):11–34, 2006.

[27] Alexander Felfernig, Michael Jeran, Gerald Ninaus, Florian Reinfrank, and Stefan Reiterer. Toward the next generation of recommender systems: applications and research challenges. In *Multimedia services in intelligent environments*, pages 81–98. Springer, 2013.

[28] Alexander Felfernig, Monika Schubert, and Christoph Zehentner. An efficient diagnosis algorithm for inconsistent constraint sets. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AI EDAM*, 26(1):53, 2012.

[29] Alexander Felfernig, Christoph Zehentner, and Paul Blazek. Corediag: Eliminating redundancy in constraint sets. *DX*, pages 219–224, 2011.

[30] Francesca Arcelli Fontana, Vincenzo Ferme, and Stefano Spinelli. Investigating the impact of code smells debt on quality code evaluation. In *2012 Third International Workshop on Managing Technical Debt (MTD)*, pages 15–22. IEEE, 2012.

[31] Martin Fowler. Technical debt quadrant. *Martin Fowler*, pages 14–0, 2009.

[32] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In *OTM Confederated International Conferences"On the Move to Meaningful Internet Systems*

, pages 986–996. Springer, 2003.

[33] Yuepu Guo, Rodrigo Oliveira Spínola, and Carolyn Seaman. Exploring the costs of technical debt management–a case study. *Empirical Software Engineering*, 21(1):159–182, 2016.

[34] Kiran Jammalamadaka and V Rama Krishna. Agile software development and challenges. *International Journal of Research in Engineering and Technology*, 2(08):125–129, 2013.

[35] Ilya Khomyakov, Zufar Makhmutov, Ruzilya Mirgalimova, and Alberto Sillitti. An analysis of automated technical debt measurement. In *International Conference on Enterprise Information Systems*, pages 250–273. Springer, 2019.

[36] William R King and Jun He. A meta-analysis of the technology acceptance model. *Information & management*, 43(6):740–755, 2006.

[37] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[38] Joseph A Konstan and John Riedl. Recommender systems: from algorithms to user experience. *User modeling and user-adapted interaction*, 22(1-2):101–123, 2012.

[39] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[40] Philippe Kruchten, Robert L Nord, Ipek Ozkaya, and Davide Falessi. Technical debt: towards a crisper definition report on the 4th international workshop on managing technical debt. *ACM SIGSOFT Software Engineering Notes*, 38(5):51–54, 2013.

[41] Jo Ann Lane, Supannika Koolmanojwong, and Barry Boehm. 4.6. 3 affordable systems: Balancing the capability, schedule, flexibility, and technical debt tradespace. In *INCOSE International Symposium*, volume 23, pages 1385–1399. Wiley Online Library, 2013.

[42] Valentina Lenarduzzi, Terese Besker, Davide Taibi, Antonio Martini, and Francesca Arcelli Fontana. A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools. *Journal of Systems and Software*, 171:110827, 2020.

[43] Valentina Lenarduzzi, Vladimir Mandić, Andrej Katin, and Davide Taibi. How long do junior developers take to remove technical debt items? In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–6, 2020.

[44] Valentina Lenarduzzi, Nyyti Saarimäki, and Davide Taibi. The technical debt dataset. In *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 2–11, 2019.

[45] Zengyang Li, Paris Avgeriou, and Peng Liang. A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101:193–220, 2015.

[46] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[47] Vladimir Mandić, Nebojša Taušan, and Robert Ramač. The prevalence of the technical debt concept in serbian it industry: Results of a national-wide survey. In *Proceedings of the 3rd International Conference on Technical Debt*, pages 77–86, 2020.

[48] Antonio Martini, Terese Besker, and Jan Bosch. Technical debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations. *Science of Computer Programming*, 163:42–61, 2018.

[49] Antonio Martini and Jan Bosch. Towards prioritizing architecture technical debt: information needs of architects and product owners. In *2015 41St euromicro conference on software engineering and advanced applications*, pages 422–429. IEEE, 2015.

[50] Antonio Martini and Jan Bosch. Revealing social debt with the caffea framework: An antidote to architectural debt. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, pages 179–181. IEEE, 2017.

[51] Antonio Martini, Jan Bosch, and Michel Chaudron. Investigating architectural technical debt accumulation and refactoring over time: A multiple-case study. *Information and Software Technology*, 67:237–253, 2015.

[52] Judith Masthoff. Group recommender systems: Combining individual models. In *Recommender systems handbook*, pages 677–702. Springer, 2011.

[53] Bruce R Maxim and Marouane Kessentini. An introduction to modern software quality assurance. In *Software Quality Assurance*, pages 19–46. Elsevier, 2016.

[54] Frank Mccarey, Mel Ó Cinnéide, and Nicholas Kushmerick. Rascal: A recommender agent for agile reuse. *Artificial Intelligence Review*, 24(3-4):253–276, 2005.

[55] S McConnell. Technical debt-10x software development| construx, 1 november 2007, 2014.

[56] Bamshad Mobasher and Jane Cleland-Huang. Recommender systems in requirements engineering. *AI magazine*, 32(3):81–89, 2011.

[57] Nachiappan Nagappan, Brendan Murphy, and Victor Basili. The influence of organizational structure on software quality. In *2008 ACM/IEEE 30th International Conference on Software Engineering*, pages 521–530. IEEE, 2008.

[58] Robert L Nord, Ipek Ozkaya, Philippe Kruchten, and Marco Gonzalez-Rojas. In search of a metric for managing architectural technical debt. In *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, pages 91–100. IEEE, 2012.

[59] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

[60] Bernhard Peischl, Markus Zanker, Mihai Nica, and Wolfgang Schmid. Constraint-based recommendation for software project effort estimation. *Journal of Emerging Technologies in Web Intelligence*, 2(4):282–290, 2010.

[61] Štefan Pero and Tomáš Horváth. Opinion-driven matrix factorization for rating prediction. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 1–13. Springer, 2013.

[62] Martin Pinzger, Nachiappan Nagappan, and Brendan Murphy. Can developer-module networks predict failures? In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 2–12, 2008.

[63] Nicolli Rios, Manoel Gomes de Mendonça Neto, and Rodrigo Oliveira Spínola. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology*, 102:117–145, 2018.

[64] Nicolli Rios, Rodrigo Oliveira Spínola, Manoel Mendonça, and Carolyn Seaman. The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from brazil. *Empirical Software Engineering*, pages 1–72, 2020.

[65] Martin Robillard, Robert Walker, and Thomas Zimmermann. Recommendation systems for software engineering. *IEEE software*, 27(4):80–86, 2009.

[66] Pilar Rodríguez, Alireza Haghighatkhah, Lucy Ellen Lwakatare, Susanna Teppola, Tanja Suomalainen, Juho Eskeli, Teemu Karvonen, Pasi Kuvaja, June M Verner, and Markku Oivo. Continuous deployment of software intensive products and services: A systematic mapping study. *Journal of Systems and Software*, 123:263–291, 2017.

[67] Pilar Rodríguez, Mika Mäntylä, Markku Oivo, Lucy Ellen Lwakatare, Pertti Seppänen, and Pasi Kuvaja. Advances in using agile and lean processes for software development. In *Advances in Computers*, volume 113, pages 135–224. Elsevier, 2019.

[68] Ken Schwaber and Jeff Sutherland. The scrum guide-the definitive guide to scrum: The rules of the game. *SCRUM. org, Jul-2013*, 2013.

[69] Robert Schwanke, Lu Xiao, and Yuanfang Cai. Measuring architecture quality by structure plus history analysis. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 891–900. IEEE, 2013.

[70] Tushar Sharma, Girish Suryanarayana, and Ganesh Samarthyam. Challenges to and solutions for refactoring adoption: An industrial perspective. *IEEE Software*, 32(6):44–51, 2015.

[71] Giancarlo Sierra, Emad Shihab, and Yasutaka Kamei. A survey of self-admitted technical debt. *Journal of Systems and Software*, 152:70–82, 2019.

[72] Daniela Steidl and Sebastian Eder. Prioritizing maintainability defects based on refactoring recommendations. In *Proceedings of the 22nd International Conference on Program Comprehension*, pages 168–176, 2014.

[73] Adam Tornhill. Prioritize technical debt in large-scale systems using codescene. In *Proceedings of the 2018 International Conference on Technical Debt*, pages 59–60, 2018.

[74] Santiago Vidal, Hernan Vazquez, J Andres Diaz-Pace, Claudia Marcos, Alessandro Garcia, and Willian Oizumi. Jspirit: a flexible tool for the analysis of code smells. In *2015 34th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–6. IEEE, 2015.

[75] Janett Walters-Williams and Yan Li. Comparative study of distance functions for nearest neighbors. In *Advanced techniques in computing sciences and software engineering*, pages 79–84. Springer, 2010.

[76] Hanzhang Wang, Marouane Kessentini, William Grosky, and Haythem Meddeb. On the use of time series and search based software engineering for refactoring recommendation. In *Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems*, pages 35–42, 2015.

[77] Jesse Yli-Huumo, Andrey Maglyas, and Kari Smolander. How do software development teams manage technical debt?–an empirical study. *Journal of Systems and Software*, 120:195–218, 2016.

[78] Fiorella Zampetti, Cedric Noiseux, Giuliano Antoniol, Foutse Khomh, and Massimiliano Di Penta. Recommending when design technical debt should be self-admitted. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 216–226. IEEE, 2017.

[79] Nico Zazworka, Carolyn Seaman, and Forrest Shull. Prioritizing design debt investment opportunities. In *Proceedings of the 2nd Workshop on Managing Technical Debt*, pages 39–42, 2011.

[80] Nico Zazworka, Michele A Shaw, Forrest Shull, and Carolyn Seaman. Investigating the impact of design debt on software quality. In *Proceedings of the 2nd Workshop on Managing Technical Debt*, pages 17–23, 2011.

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO CIENCIAS DA COMPUTACAO
Rua Aprígio Veloso, 882, - Bairro Universitário, Campina Grande/PB, CEP 58429-900

DECLARAÇÃO

Processo nº 23096.015007/2021-42

Declaramos, para fins de comprovação que, JOSÉ FERDINANDY SILVA CHAGAS, matrícula 0118015803-5D, é aluno(a) regularmente matriculado(a) no curso de DOUTORADO do Programa de Pós-Graduação Stricto Sensu em Ciência da Computação da Universidade Federal de Campina Grande - UFCG, na área de CIÊNCIA DA COMPUTAÇÃO, sob a orientação de HYGGO OLIVEIRA DE ALMEIDA

Prof. Everton Leandro Galdino Alves
Coordenador da Pós-Graduação em Ciência da Computação

Documento assinado eletronicamente por **EVERTON LEANDRO GALDINO ALVES**, **COORDENADOR (A)**, em 24/03/2021, às 10:51, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **1363237** e o código CRC **5B7F9959**.

Referência: Processo nº 23096.015007/2021-42 SEI nº 1363237

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO CIENCIAS DA COMPUTACAO
Rua Aprígio Veloso, 882, - Bairro Universitário, Campina Grande/PB, CEP 58429-900

DECLARAÇÃO

Processo nº 23096.015007/2021-42

Histórico Escolar Resumido

Nome: JOSÉ FERDINANDY SILVA CHAGAS
Matrícula: 0118015803-5D

CÓDIGO	DISCIPLINA	PERÍODO	CRED.	NOTA	CONCEITO	SITUAÇÃO
	EQUIVALÊNCIA AO TÍTULO DE MESTRE		22			APROVADO
INF106	FUNDAMENTOS DE PESQUISA EM CIÊNCIA DA COMPUTAÇÃO I	2018.1	4	8.4		APROVADO
INF107	FUNDAMENTOS DE PESQUISA EM CIÊNCIA DA COMPUTAÇÃO II	2018.1	4	5,2		REPROVADO
INF108	FUNDAMENTOS DE PESQUISA EM CIÊNCIA DA COMPUTAÇÃO III	2018.1	4	8		APROVADO
CC001	PROJETO DE PESQUISA(APOIO À DECISÃO EM PROJETOS ÁGEIS BASEADO EM ANÁLISE DE DADOS)	2018.2	2	10,0		APROVADO
69883	PROFICIÊNCIA EM LÍNGUA ESTRANGEIRA (INGLÊS)	2018.2	0			APROVADO

INF066	PROFICIÊNCIA EM LÍNGUA ESTRANGEIRA (ESPANHOL)	2018.2	0			APROVADO
CC004	ESTÁGIO DOCÊNCIA I (APROVEITAMENTO DE ESTUDOS)	2019.1	2	APROVADO		APROVADO
CC005	ESTÁGIO DOCÊNCIA II (APROVEITAMENTO DE ESTUDOS)	2019.1	2	APROVADO		APROVADO
INF062	PROJETO DE PESQUISA (ANÁLISE DE USER STORIES PARA SUPORTE À TOMADA DE DECISÃO)	2019.1	2	10,0		APROVADO
INF062	PROJETO DE PESQUISA (REUSO DE CONHECIMENTO NO SUPORTE À TOMADA DE DECISÃO)	2019.2	2	9,5		APROVADO
INF062	PROJETO DE PESQUISA (REUSO DE CONHECIMENTO NO SUPORTE À TOMADA DE DECISÃO II)	2020.1	2	9,0		APROVADO
CC001	PROJETO DE PESQUISA (REUSO DE CONHECIMENTO NO SUPORTE À TOMADA DE DECISÃO III)	2020.2	2	9,0		APROVADO
CC007	QUALIFICAÇÃO DE DOUTORADO	2020.2	0	APROVADO		APROVADO
INF077	TRABALHO DE TESE	2021.1	0			MATRICULADO

CRA: 8.25
Carga horário obtida: 720 horas.
Obs.: Cada crédito corresponde a 15 horas/aula.
CRÉDITOS INTEGRALIZADOS PARA O DOUTORADO: 44.

Prof. Everton Leandro Galdino Alves
Coordenador da Pós-Graduação em Ciência da Computação

Documento assinado eletronicamente por **EVERTON LEANDRO GALDINO ALVES**, **COORDENADOR (A)**, em 24/03/2021, às 10:52, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **1363246** e o código CRC **F65EF0BB**.

Referência: Processo nº 23096.015007/2021-42 SEI nº 1363246

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO - UFERSA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO - PROPPG
Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

(Obrigatório)

TERMO DE DECLARAÇÃO E COMPROMISSO

EU, JOSÉ FERDINANDY SILVA CHAGAS, portador do CPF nº 014.863.283-18 RG nº 2003030026003, matrícula siape nº 3608635, devidamente autorizado(a) pela Universidade Federal Rural do Semi-Árido – UFERSA para realizar o curso de Doutorado em Ciência da Computação, pelo presente e na melhor forma de direito, conforme a Lei nº 8.112/90, em seu Artigo 96-A, o Regimento Geral da UFERSA, em seu Artigo 338, e a RESOLUÇÃO CONSUNI/UFERSA Nº 003/2018, de 25 de junho de 2018, assumo o compromisso formal de permanecer, obrigatoriamente a serviço da UFERSA, por tempo integral e com dedicação exclusiva por um prazo igual ao do afastamento, a contar da conclusão do referido curso, sob pena de ressarcimento de todas as despesas, diretas ou indiretas em que a mesma tenha incorrido financiando aquele curso, tais como: salários, gratificações, passagens, diárias, ajudas de custo, bolsa de complementação salarial, bolsa de estudos, custos de matrícula, mensalidades e anuidades, enfim, qualquer dispêndio feito pela União, através da sua administração direta ou indireta, centralizada ou descentralizada, com o fim de custeio do curso em epígrafe.

Declaro estar ciente das Normas e Regulamentos do Curso.

Fica eleito o foro da Justiça Federal, Seção Judiciária do Rio Grande do Norte para dirimir todas as questões porventura decorrentes deste instrumento.

Pau dos Ferros (RN), 27 de março de 2021.

José Ferdinandy Silva Chagas
Assinatura
(Obrigatória)

Jéssica Regiane Silva de Oliveira
Nome da testemunha (Obrigatório)
CPF: 054.508.193-08

Nome da testemunha (Obrigatório)
CPF: 054.406.995-50

Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

(Anexo IX)

PARECER DA CHEFIA IMEDIATA

(Departamento Acadêmico de lotação do requerente)

(Obrigatório)

Pode utilizar documento oficial do setor (Departamento) em que o solicitante esteja vinculado dispensando este formulário.

Data: ___/___/____

_____

Assinatura do Chefe imediato

Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

(Anexo X)

PARECER DO CONSELHO DO CENTRO AO QUAL O REQUERENTE FAZ PARTE
(Obrigatório)

Pode utilizar documento oficial do CONSELHO DO CENTRO em que o solicitante esteja vinculado dispensando este formulário.

Data: ___/___/____

_____
Assinatura do presidente do Conselho de Centro

Av. Francisco Mota, 572 – C. Postal 137 – Bairro Pres. Costa e Silva – Mossoró – RN – CEP: 59.625-900 - Tel.: (84)3317-8296/8295 – E.mail: proppg@ufersa.edu.br

INFORMAÇÕES IMPORTANTES

A falta de qualquer um destes anexos irá indeferir seu pedido de renovação de afastamento.

A solicitação de renovação de afastamento do docente deverá ser **apreciada e aprovada**, sucessivamente, nas seguintes instâncias:

I - Assembleia do Departamento Acadêmico de lotação do requerente;

II - Conselho do Centro ao qual o requerente faz parte;

III - PROPPG;

IV - PROGEPE;

V - Comissão Permanente de Pessoal Docente (CPPD); e

VI - Conselho Superior competente.

Dúvidas? Laia a RESOLUÇÃO CONSUNI/UFERSA Nº 003/2018, de 25 de junho de 2018, publicada no site da PROPPG.

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
GABINETE

MEMORANDO ELETRÔNICO Nº 133/2021 - GR (11.03)
(Identificador: 202188035)

Nº do Protocolo: 23091.004977/2021-56

Mossoró-RN, 22 de Abril de 2021.

CENTRO MULTIDISCIPLINAR - PAU DOS FERROS

CC:
CAMPUS PAU DOS FERROS

PRÓ-REITORIA DE GESTÃO DE PESSOAS

DIVISÃO DE DESENVOLVIMENTO DE PESSOAL

Título: Destinação de Códigos de Vaga de Professor para o Centro Multidisciplinar de Pau dos Ferros

Senhor/a Diretor/ra de Centro,

01. Considerando a PORTARIA Nº 213, de 14 de abril de 2021, do Ministério da Educação, a qual remaneja códigos de vaga de Professor do Magistério Superior para a UFERSA, vimos comunicar que foram alocados **03 (três) códigos de vaga** para o **Centro Multidisciplinar de Pau dos Ferros**.

02. Dessa forma, solicitamos que **até o dia 05 de maio de 2021 (quarta-feira)** seja enviado memorando à Divisão de Desenvolvimento de Pessoas - DDP/PROGEPE com cópia para o Gabinete da Reitoria informando o perfil das vagas a serem ocupadas, a fim de sejam feitos os encaminhamentos cabíveis no âmbito da Pró-reitoria de Gestão de Pessoas para o preenchimento da vaga, conforme a legislação vigente.

03. Certos do atendimento desta demanda, colocamo-nos à disposição para informações que repute necessárias.

Atenciosamente,

(Autenticado em 22/04/2021 11:22)
CLAUDIA ALVES DE SOUSA MUNIZ
CHEFE DE GABINETE - TITULAR
Matrícula: 3680521

DISTRIBUIÇÃO DE CARGA HORÁRIA POR DOCENTE DO C

1º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Cálculo I	4	2	8	Profs da matemática
Análise e Expressão Textual	4	2	8	Profs de letras
Ambiente Energia e Sociedade	4	2	8	Vaga da Profª Gabriela Valones
Geometria Analítica	4	2	8	Profs da matemática
Algoritmos e Programação I	4	2	8	Profs de TI

2º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Cálculo II	4	2	8	Profs da matemática
Álgebra Linear	4	2	8	Profs da matemática
Mecânica Clássica	4	2	8	Profs de física
Laboratório de Mecânica Clássica	2	4	8	Profs de física
Química Geral	4	2	8	Profs de química
Laboratório de Química Geral	2	4	8	Profs de química

3º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Introdução às Funções de Várias Variáveis	4	2	8	Profs da matemática
Ondas e Termodinâmica	4	2	8	Profs de física
Laboratório de Ondas e Termodinâmica	2	2	4	Profs de física
Estatística	4	2	8	Vaga do Prof André Rocha
Fundamentos de Ciências dos Materiais	4	2	8	Profs de química
Economia	2	2	4	Lauro Cesar Bezerra Nogueira

4º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Expressão Gráfica	4	2	8	Profs Arquitetura
Eletricidade e Magnetismo	4	2	8	Profs de física
Sociologia	4	1	4	Glauber Barreto Luna
Mecânica Geral I	4	2	8	José Flávio Timóteo Junior

5º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Resistência dos Materiais I	4	1	4	Clawsio Rogério Cruz de Sousa
Filosofia da Ciência	4	2	8	Claudio de Souza Rocha
Administração e Empreendedorismo	4	1	4	Anderson Queiroz Lemos
Laboratório de Eletricidade e Magnetismo	2	2	4	Profs de física
Ética e Legislação	2	2	4	Katia Cilene da Silva Santos

6º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Fenômenos de Transporte	4	2	8	Ricardo Paulo Fonseca Melo
Projeto Auxiliado por Computador	4	1	4	Profs Arquitetura
Sistema de Gestão de Saúde e Segurança do Traabalho	4	1	4	Samara Martins Nascimento

URSO DE C&T

	Professor	Total final de créditos por professor
1	Profs Matemática*	40
2	Profs de letras	8
3	Vaga da Profª Gabriela Valones	8
4	Profs de TI	8
5	Profs de física	40
6	Profs de química	24
7	Vaga do Prof André Rocha	8
8	Lauro Cesar Bezerra Nogueira	4
9	Profs Arquitetura	12
10	Glauber Barreto Luna	4
11	José Flávio Timóteo Junior	8
12	Clawsio Rogério Cruz de Sousa	4
13	Claudio de Souza Rocha	8
14	Anderson Queiroz Lemos	4
15	Katia Cilene da Silva Santos	4
16	Ricardo Paulo Fonseca Melo	8
17	Samara Martins Nascimento	4

DISTRIBUIÇAO DE CARGA HORARIA POR DOCENTE DO CURSO ARQUITETURA E URBA

1º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Estética e História das Artes I	4	1	4	Anna Cristina de Andrade Ferreira
Análise e Expressão Textual	4	1	4	Profs de letras
Seminário de Introdução ao Curso	2	1	2	Profs Arquitetura
Ambiente, Energia e Sociedade	4	1	4	Profs ambiental
Introdução ao Desenho	4	1	4	Profs Arquitetura
Oficina de Plástica I	4	1	4	Francisco Rocha Vasconcelos Neto
Geometria Descritiva	4	1	4	Monique Lessa Vieira Olimpio
Matemática para Arquitetura	4	1	4	Profs de matemática

2º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Estética e História das Artes II	4	1	4	Gabriel Leopoldino Paulo de Medeiros
Filosofia da Ciência e Metodologia Científica	4	1	4	Claudio de Souza Rocha
Desenho de Arquitetura	4	1	4	Antonio Carlos Leite Barbosa
Oficina de Plástica II	4	1	4	Francisco Rocha Vasconcelos Neto
Oficina de Desenho	4	1	4	Francisco Rocha Vasconcelos Neto
Introdução à Ciência dos Materiais	2	1	2	Profs de química
Mecânica Clássica	4	1	4	Profs de física

3º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Sociologia	4	1	4	Glauber Barreto Luna
Teoria e História da Arq. do Urbanismo I	4	1	4	Gabriel Leopoldino Paulo de Medeiros
Introdução ao Projeto de Arquitetura	4	1	4	Rafaela Santana Balbi
Psicologia Ambiental	4	1	4	Trícia Caroline da Silva Santana
Material de Construção I	4	1	4	Adla Kellen Dionisio Sousa de Oliveira
Mecânica Geral I	4	1	4	José Flávio Timóteo Junior
Desenho Auxiliado por Computador I	4	1	4	Ellen Priscila Nunes de Souza

4º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Teoria e História da Arq. do Urbanismo II	4	1	4	Gabriel Leopoldino Paulo de Medeiros
Projeto de Arquitetura I	4	1	4	Daniel Paulo de Andrade Silva
Planejamento e Projeto da Paisagem I	4	1	4	Ellen Priscila Nunes de Souza
Planejamento e Projeto Urbano e Regional I	4	1	4	Trícia Caroline da Silva Santana
Materiais de Construção II	4	1	4	Adla Kellen Dionisio Sousa de Oliveira
Resistência dos Materiais I	4	1	4	Clawsio Rogerio Cruz de Souza
Desenho Auxiliado por Computador II	4	1	4	Bárbara Laís Felipe de Oliveira
Topografia	4	1	4	José Daniel Dantas Jales

5º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Teoria e História da Arquitetura e do Urbanismo III	4	1	4	Anna Cristina de Andrade Ferreira
Projeto de Arquitetura II	4	1	4	Rafaela Santana Balbi
Planejamento e Projeto da Paisagem II	4	1	4	Rafaela Santana Balbi
Planejamento e Projeto Urbano e Regional II	4	1	4	Tamms Maria da Conceição Morais Campos
Instalações I	4	1	4	Daniel Paulo de Andrade Silva
Estruturas I	4	1	4	Ellen Priscila Nunes de Souza
Conforto Ambiental I	4	1	4	Eduardo Raimundo Dias Nunes

6º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Projeto de Arquitetura III	4	1	4	Antonio Carlos Leite Barbosa
Planejamento e Projeto da Paisagem III	4	1	4	Trícia Caroline da Silva Santana
Planejamento e Projeto Urbano e Regional III	4	1	4	Tamms Maria da Conceição Morais Campos
Instalações II	4	1	4	Bárbara Laís Felipe de Oliveira
Estruturas II	4	1	4	Bárbara Laís Felipe de Oliveira
Conforto Ambiental II	4	1	4	Eduardo Raimundo Dias Nunes

7º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Projeto de Arquitetura IV	4	1	4	Francisco Rocha Vasconcelos Neto
Planejamento e Projeto Urbano e Regional IV	4	1	4	Tamms Maria da Conceição Morais Campos
Tecnologia das Edificações	4	1	4	Rafaelly Angelica Fonseca Bandeira
Conforto Ambiental III	4	1	4	Eduardo Raimundo Dias Nunes
Ergonomia do Ambiente Construído **(optativa)**	3	1	3	Rafaela Santana Balbi
Tópicos Especiais em Arquitetura e Urbanismo **(optativa)**	3	1	3	Antonio Carlos Leite Barbosa
Multimeios **(optativa)**	3	1	3	Daniel Paulo de Andrade Silva

8º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Prática Profissional	4	1	4	Monique Lessa Vieira Olimpio
Projeto de Arquitetura V	4	1	4	Monique Lessa Vieira Olimpio
Planejamento e Projeto Urbano e Regional V	4	1	4	Antonio Carlos Leite Barbosa
Orçamento, Planejamento e Controle de Obras	4	1	4	Rafaelly Angelica Fonseca Bandeira
Conservação e Técnicas Retrospectivas	4	1	4	Anna Cristina de Andrade Ferreira
Urbanismo (optativa para Eng. Ambiental)	2	1	2	Tamms Maria da Conceição Morais Campos
Urbanização e Plano Diretor Municipal **(optativa)**	3	1	3	Tamms Maria da Conceição Morais Campos

9º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Introduçao ao Trabalho de Conclusão de Curso	4	1	4	Daniel Paulo de Andrade Silva

10º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Trabalho de Conclusão de Curso	4			É definido conforme o projeto do discente

ANISMO

	Professor	Total final de créditos por professor
1	Anna Cristina de Andrade Ferreira	12
2	Antonio Carlos Leite Barbosa	15
3	Bárbara Laís Felipe de Oliveira	12
4	Daniel Paulo de Andrade Silva	15
5	Eduardo Raimundo Dias Nunes	12
6	Francisco Rocha Vasconcelos Neto	16
7	Gabriel Leopoldino Paulo de Medeiros	12
8	Monique Lessa Vieira Olimpio	12
9	Rafaela Santana Balbi	15
10	Tamms Maria da Conceição Morais Campos	15
11	Trícia Caroline da Silva Santana	12
12	Ellen Priscila Nunes de Souza	12
13	Profs Arquitetura	6
14	Profs letras	4
15	Profs ambiental	4
16	Profs de matemática	4
17	Claudio de Souza Rocha	4
18	Profs de química	2
19	Profs de física	4
20	Glauber Barreto Luna	4
21	José Flávio Timóteo Junior	4
22	Clawsio Rogerio Cruz de Souza	4

Componentes compartilhados com Eng civil e que já foram contabilizados na planilha de Eng Civil

Não contabilizada em ARQ, uma vez que para efeito deste cálculo geral, vamos considerar que ela retorne para os profs de ambiental

DISTRIBUIÇÃO DE CARGA HORÁRIA POR DOCENTE DO CURSO DE ENGEN
ESTRUTURA CURRICULAR ATUAL

1º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Ambiente, Energia e Sociedade	4	3	12	Vaga da Profª Gabriela Valones
6º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Química Ambiental	4	1	4	Kytéria Sabina Lopes Figueredo
7º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Ecologia	4	1	4	Janaína Côrtez de Oliveira
Química Orgânica	4	1	4	Kytéria Sabina Lopes Figueredo
Geoprocessamento	4	1	4	Alex Pinheiro Feitosa
8º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Poluição Ambiental	4	1	4	Joel Medeiros Bezerra
Estudo e Avaliação de Impactos	4	1	4	Joel Medeiros Bezerra
Fundamentos de Análise Química	4	1	4	Kytéria Sabina Lopes Figueredo
Microbiologia Ambiental	4	1	4	Janaína Côrtez de Oliveira
9º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Mitigação de Impactos Ambientais	4	1	4	Joseane Dunga da Costa
Gestão de Recursos Naturais e Zoneamento Ambiental	4	1	4	Joel Medeiros Bezerra
Sistemas Urbanos de Águas, Esgotos e Drenagem	4	1	4	Alex Pinheiro Feitosa
Sistemas de Gestão Ambiental	4	1	4	Jorge Luís de Oliveira Pinto Filho
Recursos Hídricos	4	1	4	Alex Pinheiro Feitosa
10º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Recuperação de Áreas Degradadas	4	1	4	Jorge Luís de Oliveira Pinto Filho
Gestão de Resíduos Sólidos	4	1	4	Vaga da Profª Gabriela Valones
Direito Ambiental	4	1	4	Jorge Luís de Oliveira Pinto Filho
Tratamento de Águas Residuárias	4	1	4	Joseane Dunga da Costa
OPTATIVAS				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Cultura e Ambiente	2	1	2	Janaína Côrtez de Oliveira
Salinização e Drenagem (optativa)	2	1	2	Joseane Dunga da Costa
Controle Ambiental na Indústria Têxtil (optativa)	2	1	2	Joseane Dunga da Costa
Optativa IV	2	1	2	**Sem professor**
Optativa V	2	1	2	**Sem professor**

DISTRIBUIÇÃO DE CARGA HORÁRIA POR DOCENTE DO CURSO DE ENGEN
ESTRUTURA CURRICULAR NOVA

1º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Ambiente, Energia E sociedade	4	3	12	Vaga da Profª Gabriela Valones
4º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Química Orgânica	4	1	4	Kytéria Sabina Lopes Figueredo
6º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Climatologia e Metereologia	2	1	2	**Sem professor**
Direito Ambiental	4	1	4	Jorge Luís de Oliveira Pinto Filho
Ecologia	4	1	4	Janaína Côrtez de Oliveira
7º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Geoprocessamento	4	1	4	Alex Pinheiro Feitosa
Química Ambiental	4	1	4	Kytéria Sabina Lopes Figueredo
Poluição Ambiental	4	1	4	Joel Medeiros Bezerra
8º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Microbiologia Ambiental	4	1	4	Janaína Côrtez de Oliveira
Laboratório de Microbiologia Ambiental	4	1	4	Janaína Côrtez de Oliveira
Estudo e Avaliação de Impactos Ambientais	4	1	4	Joel Medeiros Bezerra
Laboratório de Química Ambiental	4	1	4	Kytéria Sabina Lopes Figueredo
9º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Saúde Ambiental	4	1	4	Joseane Dunga da Costa
Gestão Ambiental Empresarial	4	1	4	Jorge Luís de Oliveira Pinto Filho
Gestão de Recursos Naturais	4	1	4	Joel Medeiros Bezerra
Tratamento de águas Residuárias	4	1	4	Alex Pinheiro Feitosa
Fundamentos da Ciência do Solo	4	1	4	Joseane Dunga da Costa
10º semestre				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Projeto de Engenharia Ambiental e Sanitária	4	1	4	Vaga da Profª Gabriela Valones
Drenagem e Esgotamento Sanitário	4	1	4	Alex Pinheiro Feitosa
Gerencimento e Manejo das Águas	4	1	4	Vaga da Profª Gabriela Valones
Gestão de Resíduos Sólidos	4	1	4	Vaga da Profª Gabriela Valones
Recuperação de Áreas Degradadas	4	1	4	Jorge Luís de Oliveira Pinto Filho
Técnicas de Controle Ambiental	4	1	4	Joseane Dunga da Costa
OPTATIVAS				
Componente	**Créditos**	**Número de turmas**	**Creditos x Turmas**	**Professor responsável**
Modelagem de Sistemas Ambientais	2	1	2	Vaga da Profª Gabriela Valones
Optativa II	2	1	2	**Sem professor**
Optativa III	2	1	2	**Sem professor**
Optativa IV	2	1	2	**Sem professor**

Cada discente do curso de Engenharia Ambiental e Sanitária terá que cumprir uma CH de 120 Horas de Optativas.

NHARIA AMBIENTAL E SANITÁRIA

	Professor	Total final de créditos por professor	
1	Alex Pinheiro Feitosa	12	Coordenador do curso
2	Joel Medeiros Bezerra	12	
3	Janaína Côrtez de Oliveira	10	
4	Jorge Luís de Oliveira Pinto Filho	12	
5	Joseane Dunga da Costa	12	Vice-coordenadora
6	Kytéria Sabina Lopes Figueredo	12	
7	Vaga da Profª Gabriela Valones	12	
	Sem professor	4	
	já contabilizada na planilha do C&T e ARQ		

NHARIA AMBIENTAL E SANITÁRIA

	Professor	Total final de créditos por professor	
1	Alex Pinheiro Feitosa	12	Coordenador do curso
2	Joel Medeiros Bezerra	12	
3	Janaína Côrtez de Oliveira	12	
4	Jorge Luís de Oliveira Pinto Filho	12	
5	Joseane Dunga da Costa	12	Vice-coordenadora
6	Kytéria Sabina Lopes Figueredo	12	
7	Vaga da Profª Gabriela Valones	14	
	Sem professor	8	
	já contabilizada na planilha do C&T e ARQ		

DISTRIBUIÇÃO DE CARGA HORÁRIA POR DOCENTE DO CURSO DE

5º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Resistência dos Materiais II	4	1	4	Paulo Henrique Araújo Bezerra
Topografia	4	1	4	Wesley de Oliveira Santos
Hidráulica	4	1	4	Wesley de Oliveira Santos
Geologia Aplicada à Engenharia	4	1	4	Marília Cavalcante Santiago

6º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Materiais de Construção I	4	1	4	Adla Kellen Dionisio Sousa de Oliveira
Mecânica das Estruturas I	4	1	4	Paulo Henrique Araújo Bezerra
Eletricidade Básica	4	1	4	Adelson Menezes Lima
Mecânica dos Solos I	4	1	4	José Daniel Dantas Jales

7º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Materiais de Construção II	4	1	4	Adla Kellen Dionisio Sousa de Oliveira
Saneamento	4	1	4	Alisson Gadelha de Medeiros
Mecânica das Estruturas II	4	1	4	Paulo Henrique Araújo Bezerra
Estradas	4	1	4	Marília Cavalcante Santiago
Instalações Hidrossanitárias	4	1	4	Alisson Gadelha de Medeiros
Mecânica dos Solos II	4	1	4	José Daniel Dantas Jales
Instalações Elétricas	4	1	4	Adelson Menezes Lima

8º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Tecnologia das Edificações	4	1	4	Rafaely Angélica Fonseca Bandeira
Sistemas de Abastecimento de Água	4	1	4	Alisson Gadelha de Medeiros
Estruturas de Aço	4	1	4	Matheus Fernandes de Araújo Silva
Estruturas de Concreto Armado I	4	1	4	Leonardo Henrique Borges de Oliveira
Hidrologia	4	1	4	Wesley de Oliveira Santos
Engenharia dos Transportes	4	1	4	Marília Cavalcante Santiago

9º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Orçamento, Planejamento e controle de Obras	4	1	4	Rafaely Angélica Fonseca Bandeira
Estruturas de Concreto Armado II	4	1	4	Leonardo Henrique Borges de Oliveira
Fundações e Estruturas de Contenção	4	1	4	Matheus Fernandes de Araújo Silva

10º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Fontes Alternativas de energia (optativa)	4	1	4	Adelson Menezes Lima
Gestão da Produção na Construção Civil (optativa)	4	1	4	Rafaely Angélica Fonseca Bandeira
Patologia e Reabilitação das Construções (optativa)	4	1	4	Adla Kellen Dionisio Sousa de Oliveira
Estruturas de Concreto Protendido (optativa)	4	1	4	Matheus Fernandes de Araújo Silva
Pontes (optativa)	4	1	4	Leonardo Henrique Borges de Oliveira

ENG CIVIL

	Professor	Total final de créditos por professor
1	Paulo Henrique Araújo Bezerra	12
2	Wesley de Oliveira Santos	12
3	Marília Cavalcante Santiago	12
4	Adla Kellen Dionisio Sousa de Oliveira	12
5	Adelson Menezes Lima	12
6	José Daniel Dantas Jales	12
7	Alisson Gadelha de Medeiros	12
8	Rafaely Angélica Fonseca Bandeira	12
9	Matheus Fernandes de Araújo Silva	12
10	Leonardo Henrique Borges de Oliveira	12

DISTRIBUIÇÃO DE CARGA HORÁRIA POR DOCENTE DO CURSO

1º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Algoritmos	4	2	8	Reudismam Rolim de Sousa/Marco Aurélio Diego Mesquita
Laboratório de Algoritmos	2	2	4	Reudismam Rolim de Sousa/Marco Aurélio Diego Mesquita
Introdução a Computação e aos Sistemas de Informação	4	2	8	Lenardo Chaves e Silva
Cálculo I	4	2	8	Profs de matemática
Análise e Expressão Textual	4	2	8	Profs de letras
Ética e Legislação	2	2	4	Kátia Cilene da Silva Santos
Seminário de Introdução ao Curso	2	1	2	Profs de TI

2º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Algoritmos e Estruturas de Dados I	4	2	8	Thiago Pereira Rique/José Ferdinandy Silva Chagas
Laboratório de Algoritmos e Estruturas de Dados I	2	2	4	Thiago Pereira Rique/Rodrigo Soares Semente
Arquitetura e Organização de Computadores	4	2	8	Kennedy Reurison Lopes/Vinicius Samuel Valério de Souza
Cálculo II	4	2	8	Profs de Matemática
Geometria Analítica	4	2	8	Profs de Matemática
Administração e Empreendedorismo	4	2	8	Anderson Queiroz Lemos
Sociologia	4	2	8	Glauber Barreto Luna

3º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Algoritmos e Estruturas de Dados II	4	1	4	Thiago Pereira Rique
Laboratório de Algoritmos e Estruturas de Dados II	2	1	2	Thiago Pereira Rique
Sistemas Operacionais	4	1	4	Kennedy Reurison Lopes
Matemática Discreta	4	2	8	Claudio Andrés Callejas Olguín
Introdução às Funções de Várias Variáveis	4	1	4	Profs de matemática
Álgebra Linear	4	1	4	Profs de matemática
Economia para Engenharias	4	1	4	Lauro Cesar Bezerra Nogueira

4º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Programação Orientada a Objetos	4	1	4	Lenardo Chaves e Silva
Banco de Dados	4	1	4	Laysa Mabel de Oliveira Fontes
Redes de Computadores	4	1	4	Vinicius Samuel Valério de Souza
Estatística	4	1	4	Estatística (Código a preencher)

5º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Engenharia de Software	4	1	4	Laysa Mabel de Oliveira Fontes
Sistemas Distribuídos	4	1	4	Walber José Adriano Silva
Computação Gráfica	4	1	4	Marco Aurélio Diego Mesquita
Filosofia da Ciência e Metodologia Científica	4	1	4	Claudio de Souza Rocha

6º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Análise e Projeto de Sistemas Orientados a Objetos	4	1	4	Laysa Mabel de Oliveira Fontes
Multimídia	4	1	4	Alysson Filgueira Milanez
Dependabilidade e Segurança	4	1	4	Walber José Adriano Silva

DE TI

	Professor	Total final de créditos por professor
1	Reudismam Rolim de Sousa	6
2	Marco Aurélio Diego Mesquita	10
3	Lenardo Chaves e Silva	12
4	Thiago Pereira Rique	12
5	José Ferdinandy Silva Chagas	4
6	Rodrigo Soares Semente	2
7	Kennedy Reurison Lopes	8
8	Vinicius Samuel Valério de Souza	8
9	Claudio Andrés Callejas Olguín	8
10	Laysa Mabel de Oliveira Fontes	12
11	Walber José Adriano Silva	8
12	Alysson Filgueira Milanez	4
13	Profs de matemática	32
14	Profs de letras	8
15	Kátia Cilene da Silva Santos	4
16	Profs de TI	2
17	Anderson Queiroz Lemos	8
18	Glauber Barreto Luna	8
19	Lauro Cesar Bezerra Nogueira	4
20	Claudio de Souza Rocha	4
21	Vaga do Prof André Rocha	4

DISTRIBUIÇÃO DE CARGA HORÁRIA POR DOCENTE DO CURSO DE

4º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Lógica Matemática	4	1	4	Claudio Andrés Callejas Olguín
Projeto Detalhado de Software	4	1	4	Jarbele Cássia da Silva Coutinho
Projeto e Design de Interfaces	4	1	4	Jarbele Cássia da Silva Coutinho

5º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Métodos Formais em Engenharia de Software	4	1	4	Alysson Filgueira Milanez
Teste de Software	4	1	4	Samara Martins Nascimento
Programação Concorrente e Distribuída	4	1	4	Vinicius Samuel Valério de Souza

6º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Processo de Software	4	1	4	José Ferdinandy Silva Chagas
Engenharia de Requisitos	4	1	4	Jarbele Cássia da Silva Coutinho
Qualidade de Software	4	1	4	Felipe Torres Leite

7º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Arquitetura de Software	4	1	4	Reudismam Rolim de Sousa
Manutenção de Software	4	1	4	José Ferdinandy Silva Chagas
Planejamento e Gerenciamento De Projetos	4	1	4	Felipe Torres Leite

8º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Metodologias Ágeis para Desenvolvimento de Software	4	1	4	Felipe Torres Leite
Gerência de Configuração e Mudanças	4	1	4	Alysson Filgueira Milanez
Optativa I	4	1	4	Reudismam Rolim de Sousa

9º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Optativa II	4	1	4	Patrick César Alves Terrematte
Optativa III	4	1	4	Reudismam Rolim de Sousa
Optativa IV	4	1	4	Samara Martins Nascimento

ENG SOFT

	Professor	Total final de créditos por professor
1	Claudio Andrés Callejas Olguín	4
2	Jarbele Cássia da Silva Coutinho	12
3	Alysson Filgueira Milanez	8
4	Samara Martins Nascimento	8
5	Vinicius Samuel Valério de Souza	4
6	José Ferdinandy Silva Chagas	8
7	Felipe Torres Leite	12
8	Reudismam Rolim de Sousa	8
9	Patrick César Alves Terrematte	4

DISTRIBUIÇÃO DE CARGA HORÁRIA POR DOCENTE DO CUR

Para o discente que entra em Eng de Computação pelo BITI

4º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Mecânica Clássica	4	1	4	Profs de física
Laboratório de Mecânica Clássica	2	1	2	Profs de física
Química Geral	4	1	4	Profs de química
Laboratório de Química Geral	2	1	2	Profs de química

5º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Ondas e Termodinâmica	4	1	4	Profs de física
laboratório de Ondas e Termodinâmica	2	1	2	Profs de física
Circuitos Digitais	4	1	4	Veronica Maria Lima Silva
Laboratório de Circuitos Digitais	2	1	2	Ádller de Oliveira Guimarães

6º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Eletricidade e Magnetismo	4	1	4	Profs de física
Laboratório de Eletricidade e Magnetismo	2	1	2	Profs de física
Sinais e Sistemas	6	1	6	Rodrigo Soares Semente

Para o discente que entra em Eng de Computação pelo BICT

3º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Introdução a Computação e aos Sistemas de Informação	4	1	4	Lenardo Chaves e Silva

4º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Arquitetura e Organização de Computadores	4	1	4	Kennedy Reurison Lopes/Vinicius Samuel Valério de Souza
Sinais e Sistemas	6	1	6	Rodrigo Soares Semente
Laboratório de Algoritmos	2	1	2	Reudismam Rolim de Sousa/Marco Aurélio Diego Mesquita

5º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Sistemas Operacionais	4	1	4	Kennedy Reurison Lopes
Algoritmos e Estruturas de Dados I	4	1	4	Thiago Pereira Rique/José Ferdinandy Silva Chagas
Laboratório de Algoritmos e Estruturas de Dados I	2	2	4	Thiago Pereira Rique/Rodrigo Soares Semente
Circuitos Digitais	4	1	4	Veronica Maria Lima Silva
Laboratório de Circuitos Digitais	2	1	2	Ádller de Oliveira Guimarães
Algoritmos e Programação I	4	1	4	Náthalee Cavalcanti de Almeida Lima

6º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Circuitos Elétricos	4	1	4	Francisco Carlos Gurgel da Silva Segundo
Matemática Discreta	4	1	4	Claudio Andrés Callejas Olguín
Cálculo Numérico	4	1	4	Patrick César Alves Terrematte
Estrutura de Dados e Programação	4	1	4	Náthalee Cavalcanti de Almeida Lima
Redes de Computadores	4	1	4	Vinicius Samuel Valério de Souza
Sistemas em Tempo-real	2	1	2	Veronica Maria Lima Silva
Algoritmos e Estrutura de Dados II	4	1	4	Thiago Pereira Rique
Laboratório de Algoritmos e Estrutura de Dados II	2	1	2	Thiago Pereira Rique
Segurança no Trabalho	4	1	4	Samara Martins Nascimento

2º ciclo de Eng de Computação

7º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Redes de Computadores	4	1	4	Vinicius Samuel Valério de Souza
Eletrônica Analógica	6	1	6	Pedro Thiago Valério de Souza
Sistemas Digitais	6	1	6	Veronica Maria Lima Silva
Sinais e Sistemas	6	1	6	Rodrigo Soares Semente
Paradigmas de Programação	4	1	4	Lenardo Chaves e Silva

8º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Sistemas Distribuídos	4	1	4	Walber José Adriano Silva
Instrumentação	4	1	4	Rodrigo Soares Semente
Sistemas de Controle I	4	1	4	Ádller de Oliveira Guimarães
Sistemas de Transmissão de Dados	4	1	4	Francisco Carlos Gurgel da Silva Segundo
Sistemas Avançados	4	1	4	Veronica Maria Lima Silva
Modelagem de Sistemas Dinâmicos	4	1	4	Ádller de Oliveira Guimarães
Banco de Dados	4	1	4	Laysa Mabel de Oliveira Fontes

9º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Sistemas Inteligentes	4	1	4	Marco Aurélio Diego Mesquita/Pedro Thiago Valério de Souza
Teoria da Computação	4	1	4	Patrick César Alves Terrematte
Sistemas de Controle II	4	1	4	Cecílio Martins de Sousa Neto
Processamento Digital de Sinais	4	1	4	Pedro Thiago Valério de Souza
Optativa I	4	1	4	Francisco Carlos Gurgel da Silva Segundo
Optativa II	4	1	4	Sem Professor

10º semestre

Componente	Créditos	Número de turmas	Creditos x Turmas	Professor responsável
Automação Industrial	4	1	4	Cecílio Martins de Sousa Neto
Optativa III	4	1	4	Cecílio Martins de Sousa Neto
Optativa IV	4	1	4	Sem professor

RSO DE ENG COMP

	Professor	Total final de créditos por professor
1	Profs de física	18
2	Profs de Química	6
3	Veronica Maria Lima Silva	16
4	Ádller de Oliveira Guimarães	10
5	Rodrigo Soares Semente	10
6	Náthalee Cavalcanti de Almeida Lima	8
7	Francisco Carlos Gurgel da Silva Segundo	12
8	Patrick César Alves Terrematte	8
9	Samara Martins Nascimento	4
10	Pedro Thiago Valério de Souza	12
11	Marco Aurélio Diego Mesquita	2
12	Cecílio Martins de Sousa Neto	12
13	**Sem professor**	8
componentes compartilhados com BTI		