UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE ENGENHARIA DE SOFTWARE

ELYNEKER LUCIANI FREIRE DA SILVA

PRINCÍPIOS E TÉCNICAS DE MODELAGEM DE DOMÍNIOS: UM ESTUDO DE CASO APLICADO NA MONITORIA ACADÊMICA

RUSSAS

2021

ELYNEKER LUCIANI FREIRE DA SILVA

PRINCÍPIOS E TÉCNICAS DE MODELAGEM DE DOMÍNIOS: UM ESTUDO DE CASO APLICADO NA MONITORIA ACADÊMICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Ceará - Campus de Russas, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Software.

Orientadora: Profa. Dra. Patrícia Freitas Campos de Vasconcelos.

RUSSAS
2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S579p Silva, Elyneker Luciani Freire da.
Princípios e Técnicas de Modelagem de Domínios : um estudo de caso aplicado na monitoria acadêmica / Elyneker Luciani Freire da Silva. – 2021.
72 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2021.
Orientação: Profa. Dra. Patrícia Freitas Campos de Vasconcelos.

1. Requisitos. 2. Domínio-problema. 3. Domain-Driven Design. I. Título.

CDD 005.1

ELYNEKER LUCIANI FREIRE DA SILVA

PRINCÍPIOS E TÉCNICAS DE MODELAGEM DE DOMÍNIOS: UM ESTUDO DE CASO APLICADO NA MONITORIA ACADÊMICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Ceará - Campus de Russas, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Software.

Aprovado em: 25/08/2021

BANCA EXAMINADORA

_____

Profa. Dra. Patrícia Freitas Campos de Vasconcelos
Universidade Federal do Ceará (UFC)

_____

Profa. Dra. Rosineide Fernando da Paz
Universidade Federal do Ceará (UFC)

_____

Prof. MS. Pitágoras Graça Martins
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Agradeço a Deus, que no meio a tantas lutas tem me feito vencer. Aos meus pais, Epitácio Luciano e Jozilda Freire, por me amarem tanto e por não medirem esforços para a realização deste sonho. À minha tia, Maria Euda, que muito me ajudou nesta caminhada. Agradeço a oração e o apoio das minhas irmãs, Jéssika Geise e Jéssia Emmyli, por tanto amor, apoio e torcida por minha vitória.

A vida nos conduz por caminhos que muitas das vezes não sabemos explicar. E em um desses caminhos onde eu buscava o crescimento profissional, quis a vida me dar de presente um amor sincero que tem me apoiado em cada minuto. Obrigado Bebeto por estar ao meu lado e ser parte desta conquista.

À minha professora orientadora, Dra. Patrícia Freitas, que com muita paciência e carinho esteve presente em cada palavra e ideia deste trabalho. Sou grato por toda a sua atenção, incentivo e dedicação. Obrigado por acreditar em uma ideia de sala de aula que, ao encontrar o seu desejo de fazer a diferença, transformou em uma oportunidade para que mais alunos pudessem desenvolver suas habilidades. Aos participantes do Projeto PAE que disponibilizaram recursos essenciais a esta pesquisa.

À banca avaliadora formada pelos professores MS. Pitágoras Martins e a Dra. Rosineide da Paz, por suas considerações e melhorias sugeridas que estão presentes neste trabalho.

Aos professores e amigos que ao longo de toda essa jornada de aprendizagem, contribuíram para o meu crescimento pessoal e profissional. Obrigado por cada conselho, ensinamento e consideração ao longo deste tempo na sala de aula, nos projetos e nos bate-papos.

Quero deixar ainda um agradecimento todo especial aos meus amigos que estiveram presentes comigo na idealização, fundação e formalização da primeira diretoria executiva da Empresa Júnior Include Jr. Todos vocês foram divisores de água na minha vida profissional. Deixamos na história deste Campus e para as futuras gerações que estarão compondo esta empresa, uma marca que representa a determinação e o desejo de vencer.

Agradeço finalmente à Universidade Federal do Ceará, em especial a todos os profissionais que formam o Campus de Russas, por proporcionar um ensino de alta qualidade e excelência. Hoje, mais do que nunca, acredito na necessidade de expansão do ensino público superior, em uma universidade que seja livre para debater ideias e que esteja no caminho contrário a este que se impõe e que causa o retrocesso. Viva a ciência!

“Essencialmente, todos os modelos
estão errados, mas alguns são úteis”.
— George E. P. Box

RESUMO

Desenvolver um projeto de software exige das equipes um alto nível de entendimento do domínio da aplicação e para que isso ocorra é necessário manter uma boa comunicação com os especialistas de negócio. Várias técnicas propõem atividades para documentar essa comunicação, entretanto, equipes de desenvolvimento entendem de maneira superficial as complexidades do domínio e isso impacta na qualidade do software. Assim, é apresentado um problema comum de especificação de requisitos ocasionado devido a dificuldades de entendimento do domínio. Para solucionar este problema, é realizado um estudo exploratório dos princípios e técnicas do design orientado pelo domínio (DDD) para que seja proposta sua aplicação em um estudo de caso na modelagem de uma ferramenta de monitoria acadêmica. Ao final, espera-se que a compreensão destes princípios contribua no entendimento amplo das complexidades do domínio, esclarecendo dúvidas existentes entre equipe de desenvolvimento e especialistas de negócio através do uso de técnicas de mapeamento de histórias de usuário, apoiando ainda o desenvolvimento através da escrita de casos de testes e definição de arquitetura adequada ao projeto.

Palavras-chave: requisitos; domínio-problema; domain-driven design.

ABSTRACT

Developing a software project requires teams to have a high level of understanding of the application domain and for this to happen it is necessary to maintain good communication with business specialists. Several techniques propose activities to document this communication, however, development teams have a superficial understanding of the complexities of the domain and this impacts on the quality of the software. Thus, a common requirement specification problem is presented due to difficulties in understanding the domain. To solve this problem, an exploratory study of the principles and techniques of domain-oriented design (DDD) is carried out in order to propose its application in a case study in the modeling of an academic monitoring tool. In the end, it is expected that the understanding of these principles will contribute to a broad understanding of the complexities of the domain, clarifying doubts between the development team and business experts through the use of user story mapping techniques, also supporting development through writing test cases and definition of appropriate architecture for the project.

Keywords: requirements; problem domain; domain-driven design.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACM	Association for Computing Machinery
art.	Artigo
BBoM	Big Ball of Mud
CAFe	Comunidade Acadêmica Federada
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
DDD	Domain-Driven Design
f_{ac}	Frequência Acumulada
f_i	Frequência Relativa
IBM	International Business Machines Corporation
IdX	Interaction Design (Design da Interação)
IEEE	Institute of Eletrical and Eletronics Engineers
IES	Instituição de Ensino Superior
IHC	Interação Humano-Computador
MEC	Ministério da Educação
n_i	Frequência Absoluta
p.	Página
PAE	Projeto de Apoio ao Ensino
RF	Requisitos Funcionais
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus 2
SIGAA	Sistema Integrado de Gestão de Atividades Acadêmicas
TDD	Test Driven Development
TIC’s	Tecnologias da Informação e Comunicação
UFC	Universidade Federal do Ceará
UML	Unified Modeling Language (Linguagem de Modelagem Unificada)

SUMÁRIO

1 INTRODUÇÃO

1.1 Problemática

Cada vez mais a Engenharia de Software tem sido essencial para a pesquisa científica quanto para a produção industrial por entender que a demanda crescente por sistemas de software necessita de novos métodos e técnicas que apoiem a especificação dos requisitos, arquitetura e construção de sistemas complexos, visto que não há como definir uma padronização universal de engenharia capaz de abordar os diferentes tipos de software (SOMMERVILLE, 2018).

Assim, mesmo com inúmeras técnicas bem consolidadas de engenharia, torna-se ainda um grande desafio desenvolver sistemas, principalmente durante as fases iniciais do projeto que buscam compreender os requisitos que envolvem o domínio da aplicação. Sendo que esta fase inicial de obtenção de requisitos é essencial por desempenhar um papel importante para a qualidade do software, já que requisitos incompletos levam ao aumento dos custos e gera maior esforço de desenvolvimento (RAHARJANA *et. al*, 2019).

Para Sommerville (2018), as complexidades crescentes dos sistemas são um dos motivos que levam ao que é conhecido por "falhas de software". Por isso, existe a necessidade de novas técnicas para a sua construção. Para Martin Fowler, entre as várias complexidades no desenvolvimento de software, a principal dificuldade está no próprio domínio-problema, ou seja, a complexidade do domínio trata-se do problema que está tentando se resolver. Para controlar essa complexidade é preciso ter um bom entendimento do modelo do domínio de modo que vá além da visão superficial (EVANS, 2020).

Deste modo, torna-se importante explorar mais o contexto em que uma aplicação está inserida para poder conseguir especificar requisitos que permitam que as equipes de desenvolvimento consigam compreender exatamente o que os clientes desejam (VAZQUEZ; SIMÕES, 2016). Compreender e construir uma linguagem comum, mas rigorosa para o entendimento dos desenvolvedores e usuários, que vá além de uma visão superficial, é essencial para a especificação de requisitos que promovam a qualidade do software.

Portanto, para explorar o contexto em que o domínio-problema está inserido, neste trabalho é proposto a realização de uma pesquisa exploratória que busca descrever como os requisitos de software podem ser especificados, partindo do estudo de uma aplicação real. Para isso, propõe-se em um estudo de caso pesquisar o contexto de uma aplicação para monitoria acadêmica, entendendo o ambiente em que esta ferramenta será utilizada, a interação e

relevância para seus usuários e como os requisitos fora do contexto do domínio podem causar incertezas na aplicação. Observa-se que ao descrever um estudo que explora um domínio-problema e propõe um modelo de *design* não se busca padronizar as atividades, nem criar ou descrever um novo processo, mas sim demonstrar a importância destes princípios e técnicas de modelagem de domínio no desenvolvimento de uma solução, pois cada software possui suas particularidades e deve ser assim trabalhado.

Portanto, este estudo se mostra relevante pois poucos são os trabalhos que abordam especificamente a compreensão do domínio sem abordar as questões técnicas da implementação como os que são apresentados por Santos (2015) e Duc Minh Le (2016). Além disso, são poucas as ferramentas tecnológicas desenvolvidas exclusivamente para uso nas monitorias acadêmicas, o que tem motivado alunos e monitores a utilizarem ferramentas muitas vezes inadequadas para o que se propõem a monitoria. A falta destes estudos e ferramentas possibilitam um ambiente em que se pode aplicar o conhecimento de desenvolvimento baseado no domínio-problema.

1.2 Justificativa

A qualidade na Engenharia de Software é fortemente fundamentada nos conceitos de gerenciamento para a melhoria dos processos (HIRAMA, 2011). Os diferentes processos de desenvolvimento requerem abordagens distintas nas suas atividades de elicitação de requisitos, sendo alguns focados exaustivamente na produção de artefatos como documentos de requisitos, diagramas UML (*Unified Modeling Language* ou simplesmente Linguagem de Modelagem Unificada), casos de uso etc.

Na tentativa de evitar requisitos incompletos, técnicas como entrevistas com usuários, questionários, *brainstorming*, análise de documentos etc. são realizadas (RAHARJANA *et. al*, 2019). Entretanto, poucas metodologias e boas práticas de desenvolvimento de software, como por exemplo o *Domain-Driven Design* (DDD), focam no estudo abrangente do domínio-problema, na delimitação dos requisitos e o contexto a que eles pertencem.

Desta forma, a realização de um estudo exploratório que busca descrever como a compreensão do domínio-problema específico de uma aplicação, e o contexto em que ela está inserida, pode contribuir na fase de especificação dos requisitos por envolver ainda mais a aproximação do desenvolvedor com o domínio.

Para Evans (2020), os desenvolvedores não têm muito interesse em aprender sobre

um domínio específico, pois o trabalho com domínios exige novos conhecimentos que parecem não acrescentar em nada à suas capacidades. Entretanto, a realização de um estudo amplo do domínio possibilita para a equipe de desenvolvimento o conhecimento necessário para a especificação e modelagem de requisitos realmente essenciais a um projeto, diminuindo o surgimento de funcionalidades complexas que não estão dentro dos limites da aplicação.

Assim, realizar um estudo para entendimento destes domínios e após isso aplicá-los na elicitação dos requisitos e modelagem de um modelo de design possibilita que sejam desenvolvidas aplicações que objetivam solucionar problemas específicos focando em questões essenciais da complexidade do negócio.

1.3 Objetivo Geral e Específico

O objetivo geral deste trabalho é explorar como o amplo entendimento das complexidades do domínio do negócio proporcionam melhores requisitos de software, eliminando dúvidas e requisitos fora do contexto da aplicação.

São objetivos específicos:

- Descrever princípios e técnicas que centralizam o desenvolvimento no contexto do domínio de uma aplicação;
- Realizar um estudo de caso que demonstre em um cenário real os desafios para a elicitação de requisitos;
- Descrever o domínio-problema em que a ferramenta do estudo de caso está inserida;
- Especificar requisitos básicos.

1.4 Estrutura do Trabalho

Neste capítulo é apresentado o contexto introdutório da questão problema abordado neste trabalho e as justificativas para esta pesquisa. Também são apresentados os objetivos que se deseja alcançar com este estudo. Logo após, é apresentada a estrutura dos demais elementos textuais:

- **Capítulo 2:** É apresentada a Fundamentação Teórica que aborda a visão de outros estudos científicos em relação aos principais assuntos abordados nesta pesquisa: 2.1 - Requisitos de Software; 2.2 - Modelos de Domínios; 2.3 -

Desenvolvimento com Foco no Usuário; 2.4 – Arquitetura em Projetos Orientado pelo Domínio.

- **Capítulo 3:** Descreve-se a Revisão Bibliográfica que aborda o contexto das pesquisas dos principais autores que influenciaram este estudo.
- **Capítulo 4:** É apresentada a Metodologia de pesquisa deste estudo, onde é descrito: 4.1 - Planejamento da Pesquisa; 4.2 - Elaboração dos Elementos Textuais; 4.3 - Redação Final.
- **Capítulo 5:** Descreve neste capítulo os Resultados Preliminares desta pesquisa: 5.1 - Etapa 1: Realizar Estudo Preliminar; 5.2 - Etapa 2: Realizar Pesquisa Bibliográfica; 5.3 - Etapa 3: Definir Técnicas para Estudo do Domínio; 5.4 - Etapa 4: Estudo de Caso;
- **Capítulo 6:** É apresentado um espaço para a Proposta de Modelagem do Domínio: 6.1 – Descrição do Problema; 6.2 – Especificando a Interação dos Atores com o Software; 6.3 – História de Usuários e Critérios de Aceitação; 6.4 – Descrevendo a Arquitetura do Software; 6.5 – Definição de Casos de Testes.
- **Capítulo 7:** São apresentadas as Considerações Finais.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os principais pontos que envolvem este estudo: 1) Explanação sobre os requisitos de software e; 2) estudo de modelo de domínios e; 3) Desenvolvimento com foco no usuário e; 4) Arquitetura em projetos orientado pelo domínio. Todas estas informações contribuirão para o entendimento do objetivo de negócio que estará no centro do processo de identificação dos requisitos, ao final desta pesquisa.

2.1 Requisitos de Software

O uso repetitivo de técnicas para se obter requisitos de software que atendam aos objetivos de negócio são característicos na engenharia de requisitos (VAZQUEZ; SIMÕES, 2016). Mas embora esta seja uma atividade relativamente madura e considerada o principal fator para o sucesso no desenvolvimento de projetos, é uma das atividades mais deficientes e problemáticas na engenharia de software (MAVIN *et. al*, 2019).

Para Sommerville (2018), alguns dos problemas existentes na engenharia de requisitos ocorrem devido aos requisitos de usuário (que são as declarações em linguagem natural) e aos requisitos de sistema (que são as descrições mais detalhadas das funções dos serviços) não estarem separados claramente nos diferentes níveis de descrição. Esta falta de clareza dos requisitos põe em risco as demais etapas de desenvolvimento e consequentemente prejudica a qualidade do software. Para Pressman e Maxim (2020), existem inúmeros problemas que podem acontecer na fase de levantamento de requisitos, como: especificação de detalhes desnecessários, *stakeholders* que possuem problemas para transmitir às necessidades aos engenheiros, pouco entendimento do domínio etc.

Inúmeras atividades como entrevistas, questionários, mapeamento de história de usuários, *event storming*, estudo da documentação etc. são propostas e aplicadas na tentativa de minimizar estes problemas durante as fases iniciais do projeto, principalmente na fase que trata do entendimento dos requisitos. Sejam estas atividades rotuladas como coleta de requisitos, captura de requisitos ou especificação dos requisitos, tem-se que os dois primeiros termos sugerem que os requisitos estão em algum lugar e basta simplesmente buscá-los, entretanto não é tão fácil e simples identificá-los (SHARP *et. al,* 2019). Para Vazquez e Simões (2016), a especificação de requisitos é uma composição de vários documentos que abrangem a visão geral das funcionalidades do sistema, glossário, os modelos do sistema que mostram os

relacionamentos entre componente, lista de requisitos funcionais e não funcionais, além das especificações detalhadas de todos estes requisitos.

Assim, para apoiar as equipes de desenvolvimento, os processos de software realizam uma padronização das atividades, fornecendo um roteiro para cada etapa. Mas, para Pressman e Maxim (2020), modelos prescritivos como o Cascata e o modelo V, possuem problemas por seguir um fluxo sequencial onde frequentemente é difícil para o cliente estabelecer todas as necessidades no início do projeto. Para Evans (2020), este modelo onde os especialistas de negócio conversam com os analistas, têm total responsabilidade de criar um modelo e repassar todo resultado de seu entendimento do domínio para os programadores é falha por não haver *feedback*.

Deste modo, elicitar e compreender os requisitos são processos difíceis já que os engenheiros de requisitos, por falta de experiência e entendimento do domínio, podem não entender termos e expressões do conhecimento implícito dos *stakeholders* (SOMMERVILLE, 2018). A dificuldade de compreensão pode levar a erros, aumentando a complexidade e impactando na qualidade do software. Para Evans (2020), quando a complexidade foge do controle, não é mais possível que os desenvolvedores entendam o software suficientemente bem para alterá-lo com facilidade e segurança.

Porém, mesmo que em geral as atividades e documentações que envolvem a engenharia de requisitos estejam relacionadas a um processo interativo, no qual são intercaladas a quantidade de tempo e esforço dedicados a cada atividade, a construção de um bom design pode oportunizar a exploração dessas complexidades do domínio, eliminando aquilo que é estranho e refletindo em um profundo conhecimento no modelo do negócio (SOMMERVILLE, 2018), (EVANS, 2020).

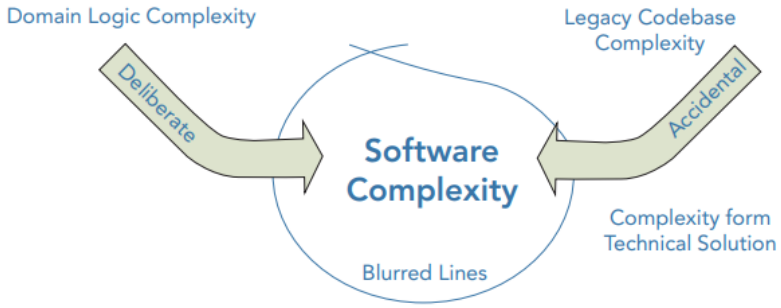
2.2 Modelos de Domínios

Os métodos de desenvolvimento de software, buscando criar um canal de comunicação uniforme entre os membros das equipes de desenvolvimento, adaptou-se ao longo dos anos acompanhando a evolução dos processos e estes evoluíram para explorar as capacidades pessoais dos indivíduos e as características específicas dos sistemas (HIRAMA, 2011). Buscou-se então garantir que a interação entre as partes envolvidas trouxesse maior entendimento do domínio e resultasse no desenvolvimento de software com qualidade.

Entende-se, portanto, que os domínios são as ideias, processos e demais representações de um problema que envolve um negócio. Os modelos de domínio são representações de um problema específico que está sendo trabalhado, onde este é modelado com objetos que possuem comportamento de negócios com significado literal e preciso (VERNON, 2016). A partir da compreensão desse modelo de domínio, é que os engenheiros de software, juntamente com os desenvolvedores, buscam criar soluções que gerem valor ao cliente.

Mas, as dificuldades durante o desenvolvimento tornam o software complexo, difícil de manter e de gerenciar. Estas complexidades no software demonstradas na Figura 01, são misturas de complexidades de domínios com complexidades técnicas (MILLETT; TUNE, 2015). Enquanto as complexidades técnicas são aquelas que envolvem tecnologia e a forma como os desenvolvedores decidem abordar o problema em um software, as complexidades de domínio são as mais significativas pois estão relacionadas ao próprio domínio, a atividade ou negócio do usuário (EVANS, 2020).

Figura 1: Complexidade no software



Fonte: MILLETT e TUNE, 2015, p.4

Para Evans (2020), é necessário atacar essas complexidades levando em consideração dois princípios norteadores:

1 – Na maioria dos projetos de software, o principal foco deve ser o domínio e a lógica do domínio;

2 – Designs de domínios complexos devem se basear em um modelo.

Estas premissas desenvolvidas por Eric Evans em 2003, são fundamentais na abordagem de design orientado pelo domínio ou simplesmente DDD (*Domain-Driven Design*)

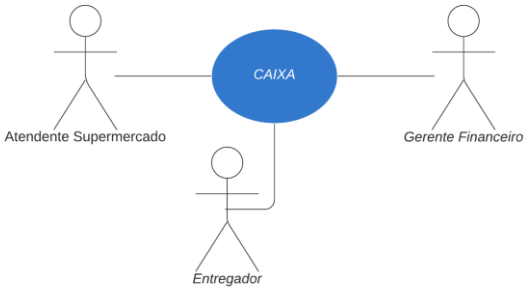
que segue os princípios do manifesto para desenvolvimento ágil de software[1] (SANTOS *et. al* 2015). Ainda no DDD são definidos três pilares para construção de modelos de domínios: linguagem ubíqua ou onipresente (*ubiquitous language*), contexto delimitado (*bounded contexts*) e o mapeamento de contextos (*context map*). Para Vernon (2016), estes pilares têm muito a oferecer pois permitem que a linguagem de uma equipe, em um contexto explícito e delimitado expresso como um modelo de domínio, agregue valor ao negócio e dê certeza da implementação correta do software.

Para Duc Minh Le *et. al* (2016), é preciso preencher lacunas existentes entre os modelos de domínios e as partes interessadas: especialistas de domínios, design e equipe de desenvolvimento. Neste estudo, para descrever como os requisitos de software podem ser especificados, aborda-se somente a complexidade do domínio por ser essencial para o desenvolvimento – já que nesta abordagem envolvem requisitos, divergências na interpretação de informações, linguagens ambíguas ou falhas além das documentações que não representam a ideia central de design de domínio.

2.2.1 Definindo Uma Linguagem Comum

Manter a comunicação entre especialistas de domínios, design e equipe de desenvolvimento é uma tarefa que necessita bastante interação para que termos e expressões possam ser compreendidos por todos (Figura 02).

Figura 2: Interpretação distinta de termo para diferentes atores



Fonte: Elaborado pelo autor.

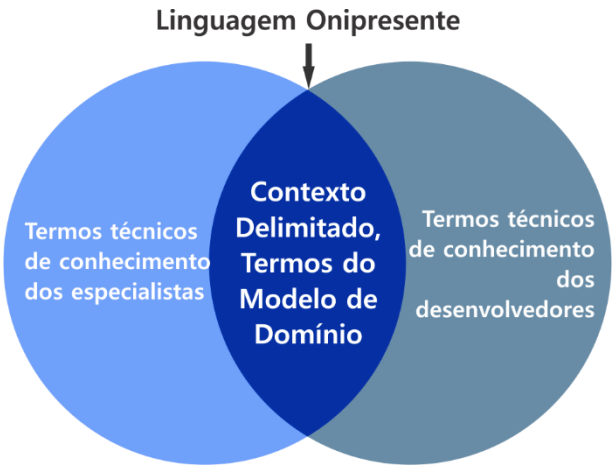
[1] Manifesto for Agile Software Development: https://agilemanifesto.org/ acessado no dia 28 de agosto de 2020.

Para Vernon (2016), a linguagem onipresente (*ubiquitous language*) é uma linguagem compartilhada por todos da equipe do projeto (sejam especialistas de domínio ou desenvolvedores). Para Santos *et. al* (2015), a linguagem onipresente, considerada a principal característica de DDD, proporciona uma melhor compreensão em relação ao domínio e comunicação mais eficaz entre os especialistas de domínio, design e a equipe de desenvolvedores. Essa compreensão é necessária visto que a linguagem de domínio eleva o nível de abstração ao expressar conceitos específicos, o que dificulta o entendimento do modelo (DUC MINH LE *et. at.* 2016).

A falta de uma linguagem compartilhada que atenda as características do domínio do problema resulta em códigos imprecisos, difícil de ler e manter, além de problemas de design na arquitetura semelhantes ao BBoM (*Big Ball of Mud*) que segundo Brian Foote e Joseph Yoder: é uma arquitetura casualmente estruturada, desleixada, espaguete etc. (MILLETT; TUNE, 2015).

Para Vernon (2016), a linguagem ubíqua utiliza conceitos e termos que incorporam os substantivos, adjetivos, verbos e expressões faladas que são padronizadas entre a equipe de desenvolvimento e os especialistas de domínios e garantem uma boa comunicação (Figura 03). Sem esta linguagem comum, os desenvolvedores têm que traduzir ideias para especialistas de domínios, que traduzem para outros especialistas e para desenvolvedores. Esta tradução confunde os conceitos do modelo levando o software a ser pouco confiável (EVANS, 2020).

Figura 3: Linguagem Onipresente



Fonte: Elaborado pelo autor, baseado em EVANS (2020)

Entretanto, não é necessário que a equipe de desenvolvedores e design se tornem especialistas do domínio, mas mantenham constante a interação entre os indivíduos e tenham interesse em conhecer cada vez mais o domínio da aplicação visto que esta ação está fortemente ligada a práticas de desenvolvimento ágil de software. Desta forma, é importante que a linguagem onipresente seja organizada para que todos os envolvidos no projeto possam entender seu significado e estejam identificadas em todos os artefatos produzidos.

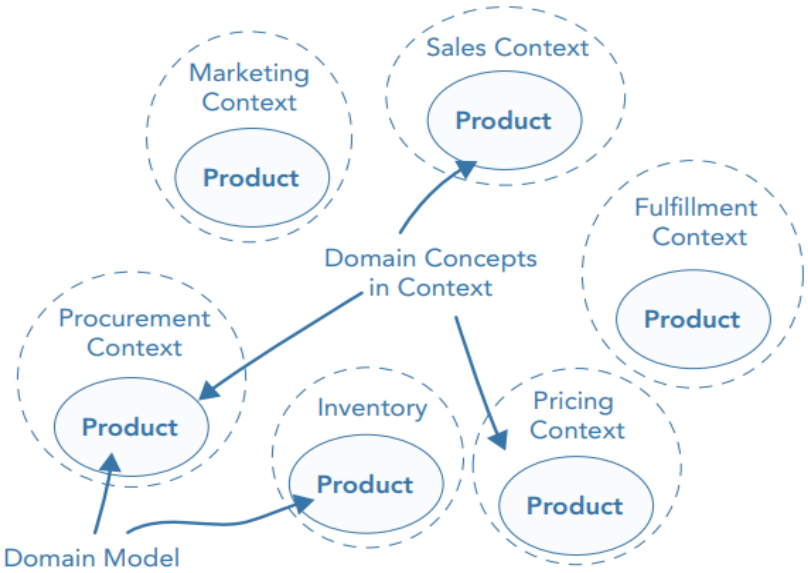
2.2.2 Contextos Delimitados

Durante a etapa de entendimento do domínio, muitos termos técnicos passam a estar presentes nas interações entre os especialistas e a equipe de desenvolvimento. Utilizando uma linguagem ubíqua, termos e expressões técnicas passam a ser utilizados na documentação e no código fonte demonstrando clareza em relação ao domínio.

Em aplicações de grande porte, conforme é explorado e entendido o domínio-problema, podem surgir subdomínios compondo o domínio principal. Estes subdomínios aumentam a complexidade do entendimento, sendo comum que termos iguais estejam presentes em subdomínios diferentes expressando ou não um mesmo sentido, de modo que cada modelo é construído para representar uma área distinta do problema (MILLET e TUNE, 2015).

Como mostrado na Figura 04, um termo, dependendo de seu contexto, pode ser utilizado de maneira diferente, impactando na lógica para desenvolvimento de uma solução. Isso é bastante relevante pois mesmo no desenvolvimento de um modelo conceitual orientado pelo domínio, as questões de implementação não devem ser desconsideradas (SANTOS *et. al.* 2015).

Figura 4: Modelos em diversos contextos

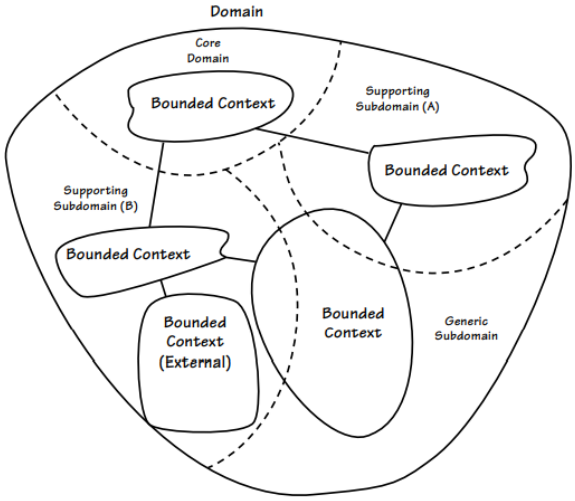


Fonte: MILLET e TUNE, 2015, p. 81

Para Vazquez e Simões (2016), o domínio do problema delimita o escopo inicial de uma solução em áreas funcionais e são pontos de partida para as atividades da engenharia de requisitos. Para Evans (2020), ao delimitar um contexto, os membros da equipe passam a ter o entendimento claro e compartilhado do que deve ser aplicado em um determinado modelo, de tal modo que seja consistente para que possa se relacionar com outros contextos.

Os modelos então devem estar vinculados a um contexto específico, de modo que sejam consistentes e significativos para o entendimento da complexidade, definidos de acordo com a linguagem utilizada pela equipe e especialistas do domínio (MILLETT; TUNE, 2015). Por isso, é importante que a equipe consiga reconhecer termos e expressões, utilizando uma linguagem onipresente para conseguir delimitar cada contexto presente no domínio principal e em seus subdomínios como visto na Figura 05.

Figura 5: Conceito abstrato de um modelo de domínio contendo subdomínios e contexto delimitado

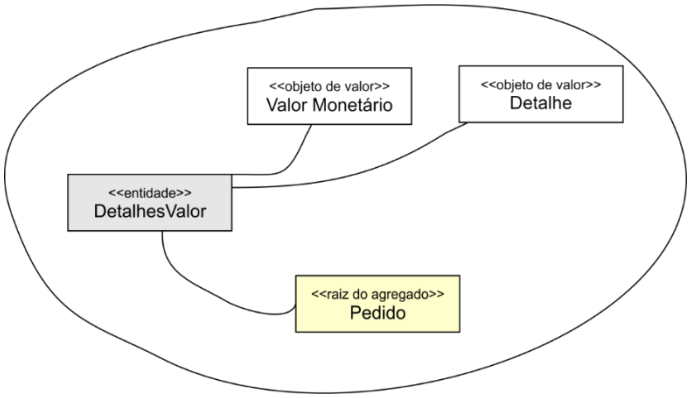


Fonte: VERNON, (2016), p. 50

Para delimitar esses contextos, é importante que se trabalhe para manter o modelo logicamente unificado, não se preocupando com a aplicabilidade fora do limite identificado. Assim, para entender onde o modelo se aplica é necessário examinar o projeto como ele é, e não como deveria ser no mundo ideal (EVANS, 2020).

Com esta modelagem estratégica do negócio e com a identificação do domínio principal e seus subdomínios e possíveis domínios genéricos, é possível explorá-los para identificar objetos de valor, raízes de agregados, entidades etc. Por exemplo, na Figura 06 observa-se como um contexto isolado é separado em: entidade (um objeto que pode ser alterado continuamente, contendo propriedades e métodos e pode ser formada por objetos de valor); objetos de valor (blocos de construção com atributos importantes que descrevem objetos quantificáveis e imutáveis dentro das entidades capazes de descrever coisas no domínio) e; agregados (uma entidade principal que representa outras entidades e/ou objetos de valor através de referências a elas dentro de um contexto, ou seja, são limites de consistência) (VERNON, 2016). Neste trabalho não busca se aprofundar nos detalhes de cada um destes e outros termos que estão relacionados ao DDD. Apenas mostra-se como o entendimento básico já impacta no pensamento da equipe de requisitos e na forma de pensar a solução para o problema de um projeto de software.

Figura 6: Exemplo do relacionamento entre objetos de valor que descrevem uma entidade pertencentes ao um contexto



Fonte: Elaborado pelo autor.

A entidade “Detalhes Valor” pode ser descrita em código para melhor compreensão da seguinte maneira como mostra a Figura 07. Vale ressaltar que neste trabalho de conclusão de curso, o foco não é a codificação, mas sim como o entendimento do negócio pode produzir uma documentação mais assertiva quanto ao domínio e os contextos que o envolve.

Figura 7: Exemplo em código de como as entidades e objetos de valor estão relacionados.

```java
//descreve o valor de alguma coisa em uma certa moeda
public final class ValorMonetario implements Serializable {
    private BigDecimal montante;
    private String moeda;

    public ValorMonetario(BigDecimal umMontante, String umaMoeda) {
        this.setMontante(umMontante);
        this.setMoeda(umaMoeda);
    }

    // ...
}

//entidade que possui um objeto de valor como um atributo
public class DetalhesValor {
    private ValorMonetario algumValor;
    private String detalhe;

    // ...
}
```

Fonte: Elaborado pelo autor.

Mas, além de entender os limites dos contextos, é importante fazer com que equipes de desenvolvimento, possam se comunicar, verificando as necessidades para cada subdomínio identificado, possibilitando uma aplicação coesa. Como o design orientado pelo domínio (DDD) segue princípios do desenvolvimento ágil de software, uma maneira para facilitar a compreensão dos requisitos é utilizando estórias de usuários (RAHARJANA *et. al*, 2019). Dessa forma, é possível que utilizando uma documentação mínima, os contextos estejam especificados e bem modelados para o entendimento das equipes, podendo ainda sofrer modificações sempre que necessário durante uma nova *sprint* de desenvolvimento.

Contudo, manter uma documentação robusta é ir contra os princípios do desenvolvimento ágil de software. Por isso, deve-se priorizar pela utilização de modelos objetivos que demonstram a ideia real do domínio. Para evitar os riscos de o documento não ser mantido é importante ser minimalista, se concentrar em abstrações fundamentais e suas interações, pois nesse nível o modelo é geralmente mais estável e capaz de mantê-lo (EVANS, 2020).

2.3 Desenvolvendo com Foco no Usuário

Ao idealizar um software é preciso levar em consideração a experiência do usuário, como ele se comporta e de que forma esta ferramenta será utilizada por pessoas do mundo real (SHARP *et. al*, 2019). Desta forma, é possível que o design desenvolvido para a aplicação represente as necessidades do negócio.

A equipe de desenvolvimento precisa entender as necessidades dos usuários para que possam compreender como as ferramentas serão mais bem utilizadas diante de necessidades reais dos usuários no contexto em que estão inseridos. Com isso, surge o design da interação (do inglês *Interaction Design* ou IxD), que é um campo relacionado à Interação Humano-Computador (IHC) que aborda como projetar tecnologias computacionais de forma a torná-las agradáveis e fáceis de utilizar (ALVES; MATOS, 2019).

Sharp (2019), define o design de interação, como a forma de projetar produtos interativos que apoiam o modo como as pessoas se comunicam e interagem em seus cotidianos. E, para projetá-los, há muitos aspectos da experiência do usuário que podem ser considerados e diversas formas de fazê-lo no design de produtos interativos.

Com isso, é necessário ter o foco nos objetivos que os usuários desejam alcançar com seu uso, ou seja, é preciso modelar pensando no usuário, analisando as necessidades, requisitos e os ambientes onde serão utilizadas. Para Barbosa (2010), um bom design de

interação só tem sentido no contexto de uma pessoa utilizando o sistema com algum objetivo claro. E por isso, os elementos-chave do processo de design são: objetivos e pessoas.

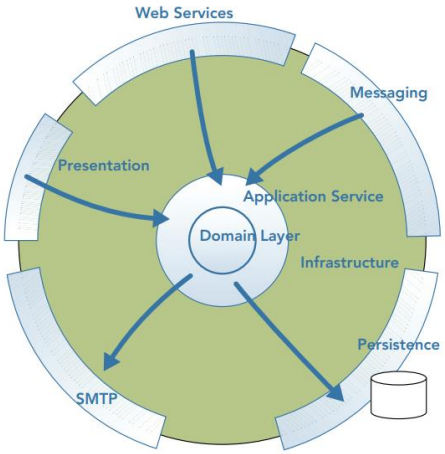
2.4 Arquitetura em Projetos Orientado pelo Domínio

Até aqui verifica-se que o entendimento dos detalhes de um domínio-problema gera melhores requisitos de software e orientam equipes a desenvolverem soluções que realmente agregam valor ao cliente. E, sem esse entendimento, erros podem surgir já nas fases iniciais, comprometendo a entrega de um software que não atende as reais necessidades dos objetivos do negócio nem atendem os atributos de qualidade desejados no design. Do mesmo modo, decidir por um modelo arquitetural antes desse entendimento, impacta na maneira como as equipes trabalharão um problema e como será mantido o ciclo de vida do sistema.

Para Martin (2018), a arquitetura de um sistema de software é a forma dada a esse sistema pelos seus criadores através da divisão e organização em componentes e no modo como eles se comunicam entre si. Entretanto, a arquitetura tem pouca relevância para o seu funcionamento, pois o problema não está na operação, mas na implementação, manutenção e desenvolvimento contínuo. Porém, a arquitetura exerce um papel importante no suporte do comportamento adequado, sendo o propósito primário dar suporte ao ciclo de vida do sistema, tornando fácil de entender, desenvolver, manter e implantar.

Utilizando domínios básicos, cuidadosamente criados e identificados no centro de um contexto delimitado, é possível que uma aplicação tenha influência arquitetônica de um ou mais modelos de design, não requerendo o uso de uma arquitetura específica. O objetivo é ter escolhas corretas através da capacidade de justificar estas arquiteturas através de requisitos funcionais disponíveis, como casos de uso ou histórias de usuários e cenários específicos para modelo de domínio. Ou seja, sem essas informações não pode se fazer escolhas arquitetônicas sólidas (VERNON, 2016).

Figura 8: Representação de camadas com diferentes responsabilidades de um aplicativo.



Fonte: MILLETT; TUNE, 2015 p. 10

Entretanto, o desenvolvimento de aplicações seguindo princípios do DDD é utilizado em muitos modelos arquiteturais que suportam o isolamento da lógica do domínio com outras camadas como mostrado na Figura 08. A camada de domínio contém a lógica do negócio, em torno está a camada de aplicação representando os casos de uso de negócio que estão isoladas e protegidas de complexidades por uma infraestrutura específica e necessária para a aplicação (MILLETT; TUNE, 2015).

Para Merson e Yoder (2020), estes isolamentos definidos por um limite modular de um contexto delimitado são possíveis candidatos a microsserviços que nem são muito grandes ou pequenos e, portanto, contêm a quantidade certa de funcionalidades. Segundo artigo publicado no site da IBM, um modelo de domínio pode ficar sobrecarregado com objetos complexos devido a atualizações e visualizações simultâneas. Com os microsserviços armazenando e gerenciando seus próprios dados, que são mais complexos do que um único modelo de domínio é capaz de suportar, é necessária uma abordagem sofisticada que trate dessas solicitações de forma independente[2].

[2] Command Query Responsibility Segregation (CQRS) pattern – IBM Cloud Architecture Center. Acessado em 10 de agosto de 2021: https://www.ibm.com/cloud/architecture/architectures/event-driven-cqrs-pattern/

3 REVISÃO BIBLIOGRÁFICA

Neste capítulo procurou-se descrever trabalhos que são importantes para fundamentar os conceitos abordados neste estudo. Estes autores, abordam conceitos como: especificação de requisitos, engenharia de requisitos e design orientado pelo domínio. Assuntos que estão em um contexto mais amplo em seus trabalhos, mas que se tornam relevantes para o entendimento desta pesquisa.

Para Mavin *et. al.* (2019), os requisitos são fatores determinantes para o sucesso ou a falha em um projeto de desenvolvimento de sistemas, referindo-se à engenharia de requisitos como um processo criativo para resolução destes problemas. Mas embora exista na engenharia de requisitos muitas formas, metodologias, ferramentas e técnicas para coletar os dados, relacioná-los continua sendo um grande desafio. Assim é proposto a criação de um quadro com uma visão geral dos métodos e abordagens utilizadas na engenharia de requisitos, propondo uma investigação teórica. Entretanto, o resultado do processo de criação deste quadro não foi bem-sucedido. Concluiu-se então, que existe um entendimento de que a engenharia de requisitos possui uma incapacidade de relacionar estas abordagens.

Segundo RAHARJANA *et. al.* (2019), uma boa obtenção dos requisitos é fundamental para a qualidade do software. No contrário, requisitos incompletos elevam o custo de desenvolvimento e esforço. Para evitar esse dilema e conseguir o conhecimento do domínio, algumas técnicas de obtenção de requisitos são realizadas. Entre essas técnicas, temos a exploração de requisitos em documentos de textos de fontes diversas e a exploração de requisitos por meio da web e das mídias sociais. Mas nestes métodos os requisitos coletados possuem limitações ao não apresentar o conhecimento geral do domínio da aplicação. RAHARJANA *et. al.* (2019), propõe que para a obtenção de requisitos nas diversas fontes da internet sejam utilizadas as notícias online e que estas estejam representadas no formato de história de usuários para que o conhecimento do domínio seja melhorado no entendimento dos desenvolvedores.

Para Santo *et. al.* (2015), diante de cenários em que os requisitos tendem a mudar constantemente, a abordagem de design orientado pelo domínio surge para o apoio a práticas ágeis de desenvolvimento e entrega de software. Esta abordagem não se trata de um processo ou metodologia, mas de uma forma de pensar, com o objetivo de esclarecer o entendimento de domínios complexos. Assim, Santos *et. al.* (2015), propõe um estudo de técnicas ágeis de testes de software que sejam complementares a esta forma de pensar o domínio de uma aplicação.

Para Duc Minh Le *et. al.* (2016), aplicar design orientado a domínios requer suprir lacunas de complexidades dos modelos de domínios percebidos entres os principais *stakeholders* (especialistas de domínios, designs da aplicação e programadores). São identificados problemas que causam complexidade de domínio entre os especialistas e complexidade de modelos entre os desenvolvedores, onde é proposto a utilização de meta-atributos (classes e interações que descrevem um objeto) para a construção de classes de domínios. Isso diminui a principal complexidade que é a de domínios entre os especialistas, tornando melhor o entendimento para os desenvolvedores do design da aplicação.

Além disso, de acordo com Merson e Yoder (2020), é possível a abordagem de design de domínios para a definição do escopo funcional em estilos arquiteturais de microsserviços. Entretanto, esta forma de abordar requisitos não é tão clara. Assim, Merson e Yoder (2020), descrevem como um modelo de domínio baseado em terminologias do DDD podem ser traduzidos em um design baseado em microsserviços.

A pesquisa neste trabalho utilizará conceitos e evidências observadas nos trabalhos destes e dos demais autores já citados, com a proposta de desenvolver uma visão crítica dos assuntos abordados e trazer para um estudo exploratório a importância de se pensar os requisitos a partir do design do modelo de domínios visto que essa é uma abordagem atual e que é uma boa prática ágil.

Na Tabela 01, observam-se resumidamente os principais assuntos abordados nos trabalhos citados e fazer um comparativo com o que é proposto nesta pesquisa:

Tabela 1: Abordagens entre os trabalhos relacionados com esta pesquisa

Abordagens/Autores	**RAHARJANA et. al (2019)**	**Mavin (2016)**	**Santos (2015)**	**Merson e Yoder (2020)**	**Duc Minh Le *et. al.* (2016)**	**Nesta Pesquisa (2020)**
DDD			X	X	**X**	**X**
Práticas Ágeis	X		X			**X**
Requisitos de Software	X	X				**X**
Modelagem de domínio			X		**X**	**X**

Fonte: Elaborado pelo autor.

Ao abordar requisitos de software, este trabalho fundamenta a importância de ter claro as necessidades do cliente para que a equipe de desenvolvimento consiga criar um modelo de design que agregue valor ao negócio. Para tanto, usando técnicas de modelagem de domínio, abordado fortemente em DDD, possibilita-se entre os especialistas de domínio e a equipe de desenvolvimento criar um canal de comunicação e interação compartilhado entre os indivíduos, fortalecendo as boas práticas ágeis. Fundamentando ainda a importância da modelagem de domínio, este trabalho torna evidente a necessidade da equipe em manter o foco no *core* da aplicação, ou seja, naquilo que realmente é essencial e que agrega valor ao cliente.

Com este comparativo, se tem uma visão crítica das principais necessidades e abordagens em torno do assunto principal deste trabalho. Delimitando o que é importante para a modelagem de uma aplicação.

4 METODOLOGIA

Neste trabalho, realizou-se uma pesquisa exploratória juntamente com um estudo de caso. De acordo com Martins Júnior (2017), a pesquisa exploratória deve ser realizada em três etapas subsequentes: **i) planejamento da pesquisa; ii) elaboração dos elementos textuais e; iii) redação final**.

Assim, a pesquisa exploratória investiga na literatura as principais ideias sobre o desenvolvimento de software baseados no domínio da aplicação e os conceitos delimitados na hipótese deste trabalho.

Para aplicar os conceitos de modelagem de domínio, visto na fundamentação teórica, realizou-se também um estudo de caso que analisa utilizando a documentação de um software, as contribuições que a modelagem de domínio pode trazer para o desenvolvimento de uma aplicação para monitoria acadêmica. Desta forma, possibilitando o entendimento do seu contexto e propondo alterações na especificação de requisitos.

4.1 Planejamento da Pesquisa

Nesta seção inicial define-se a abordagem que seria seguida para realização das etapas subsequentes desta pesquisa. Alguns questionamentos iniciais motivaram a elaboração das etapas.

(i) O que pesquisar?

(ii) Quais etapas seguir?

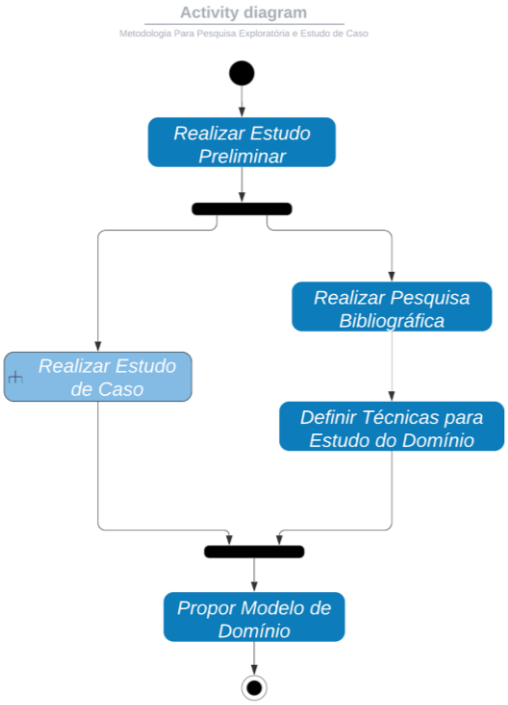
(iii) Quais ferramentas seriam utilizadas para coleta e tabulação de dados?

(iv) Quais as regras de exclusão e aceite para pesquisa bibliográfica?

4.2 Elaboração dos Elementos Textuais

Nesta seção, definem-se as principais etapas da metodologia de pesquisa deste trabalho e que estão descritos na Figura 09.

Figura 9: Especificação das Atividades da Elaboração dos Elementos Textuais da Metodologia



Fonte: Elaborado pelo autor.

4.2.1 Realizar Estudo Preliminar

Esta primeira etapa da metodologia tem por objetivo adquirir um conhecimento básico inicial sobre o tema deste trabalho através de um estudo preliminar de bibliografias especializadas em Engenharia de Software. Para isso, foram pesquisados na literatura assuntos relacionados a modelagem de domínio e requisitos de software para encontrar termos que pudessem ser utilizados como palavras-chave na etapa 4.2.2 (Realizar Pesquisa Bibliográfica). Tais termos podem ser vistos no capítulo de Resultados Preliminares (seção 5.1).

4.2.2 Realizar Pesquisa Bibliográfica

O objetivo da pesquisa bibliográfica é a identificação e seleção de artigos e outros trabalhos que estejam relacionados com o tema deste estudo exploratório definidos na seção 4.2.1 (Realizar Estudo Preliminar). Para sua realização, foi utilizado o Portal de Periódicos da

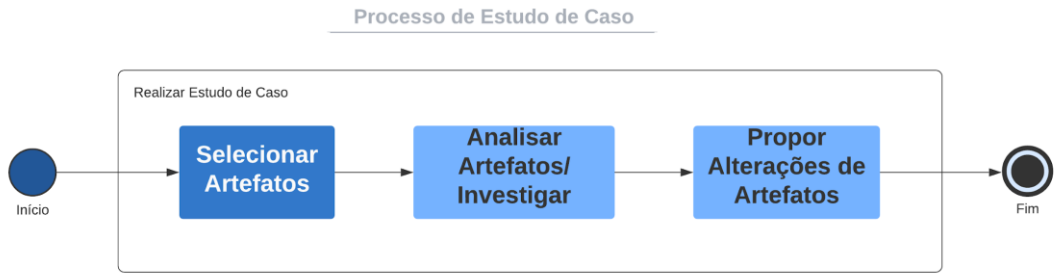
CAPES/MEC, utilizando o acesso remoto com a identificação de aluno da IES da UFC, para obter acesso aos periódicos do banco de dados da *ACM Digital Library* e preferencialmente da *IEEE Xplore*. Os resultados obtidos são descritos no capítulo de Resultados Preliminares (seção 5.2).

4.2.3 Realizar Estudo de Caso

A realização do estudo de caso objetiva representar um cenário real onde o contexto do problema é analisado e assim pode ser desenvolvido um modelo que supere suas complexidades. Nessa etapa, realizou-se então um estudo durante a fase de levantamento de requisitos de uma ferramenta para monitoria acadêmica como é descrito nos Resultados Preliminares (seção 5.4).

Seguindo o modelo de processo de pesquisa do estudo de caso demonstrado na Figura 10, foram utilizados artefatos para entender o domínio-problema da aplicação, sendo possível sugerir alterações na documentação, seguindo os princípios de modelagem de domínio vistos na Fundamentação Teórica (Capítulo 2).

Figura 10: Modelo de Processo de Pesquisa do Estudo de Caso



Fonte: Elaborado pelo autor.

4.2.4 Definir técnicas para Estudo do Domínio

Esta etapa da metodologia tem por objetivo utilizar o conhecimento adquirido com a Pesquisa Bibliográfica para fundamentar e definir os principais conceitos e técnicas utilizadas

neste estudo e que serão aplicados durante a etapa de “Propor Modelo de Domínio”. Assim, utilizando princípios de design de domínio (tais como: linguagem onipresente e contexto delimitado) e utilizando técnicas de elicitação de requisitos de software (como entrevista e documentação), entendidos a partir da pesquisa bibliográfica na seção 4.2.2, será possível um maior entendimento do domínio da aplicação sendo descritos no Estudo de Caso (seção 5.4).

4.2.5 Propor Modelo de Domínio

Esta etapa final do processo da metodologia tem por objetivo principal propor um modelo de design seguindo os princípios e técnicas definidas na seção 5.3 (Etapa 3: Definir Técnicas para Estudo do Domínio).

Nesta etapa final, será criado um modelo de aplicação utilizando princípios do desenvolvimento orientado pelo domínio, onde possua uma linguagem onipresente para os *stakeholders* que esteja delimitado dentro do contexto do problema que se deseja solucionar.

Para que isso aconteça, será realizado uma revisão da documentação existente (modelagem UML, documentos de requisitos etc.) para que possam ser compreendidas as principais necessidades e propostas as alterações, se necessárias, dos principais requisitos para atendimento ao domínio. Através de um modelo simples e com uma documentação enxuta (seguindo princípio do desenvolvimento ágil de software) onde será especificada a ferramenta com base no mapeamento do contexto do problema.

4.3 Redação Final

A última etapa de uma pesquisa exploratória consiste na redação final. O texto da redação final consistirá na descrição completa do modelo de domínio proposto, como um dos resultados desta pesquisa (ver capítulo 6).

5 RESULTADOS PRELIMINARES

Nesta seção apresentam-se os resultados preliminares obtidos através do estudo exploratório e do estudo de caso descritos no capítulo de Metodologia, além de abordar uma breve reflexão a respeito das monitorias acadêmicas.

5.1 Etapa 1: Realizar Estudo Preliminar

A partir da busca nas bibliografias de Sommerville (2018), Evans (2020), Millett e Tune (2015), e Vazquez e Simões (2016); foi possível identificar "*software requirements*", "*domain-driven design*", "*requirements specification*" e "*use story*", como sendo os principais termos a se relacionar com a proposta desta pesquisa podendo ser utilizadas como palavras-chave na busca por outros trabalhos na pesquisa bibliográfica.

5.2 Etapa 2: Realizar Pesquisa Bibliográfica

Com os termos identificados no estudo preliminar e utilizando o Portal de Periódicos da CAPES/MEC como descrito na etapa 4.2.2 da Metodologia, foram realizadas buscas nas bases de dados por trabalhos que se relacionam com o assunto para desenvolver todos os temas relacionados no capítulo de Fundamentação Teórica. Para tanto, foi observado os seguintes critérios de exclusão:

i. Os artigos e demais trabalhos científicos devem ter preferencialmente data de lançamento entre os anos de 2015 e 2020;
ii. Os artigos e demais trabalhos científicos devem conter no título e/ou no abstract uma das palavras-chave identificadas na seção 5.1.

Ainda para a compreensão sobre o assunto "monitoria acadêmica", abordado na etapa 5.2.1, foi realizado uma nova pesquisa utilizando a Plataforma de Periódicos da Capes para encontrar artigos e outros trabalhos relacionados. Utilizando uma pesquisa avançada, contendo no campo título a *string* "monitoria" e no campo assunto, contendo a *string* "ensino", a pesquisa

retornou[3] um total de 16 trabalhos. Após leitura de todos os títulos de trabalhos constantes nos resultados, foram excluídos 14 artigos por apresentarem em seus temas experiências de monitoria aplicada a algum curso de graduação ou em alguma instituição de ensino específica. Portanto, foram escolhidos apenas os dois trabalhos que abordavam a monitoria como um assunto mais amplo e genérico. Dentre estes trabalhos destacou-se Frison (2016): "Monitoria: uma modalidade de ensino que potencializa a aprendizagem colaborativa e autorregulada"; que foi escolhido para fundamentar a breve reflexão sobre monitoria acadêmica, visto que esta autora possui 486 citações, sendo 383 desde o ano de 2015 e no último ano 91 citações[4] indicado pela ferramenta *Scholar Google*.

5.2.1 Uma Reflexão Sobre Monitoria Acadêmica

A monitoria acadêmica comumente aplicada em projetos de extensão universitária e em programas de iniciação à docência, não é uma prática nova das universidades, mas sim, uma ação que ao longo dos tempos foi utilizada para transmitir o conhecimento dos alunos mais experientes para aqueles com pouco conhecimento.

De acordo com Paizani (2011), às atividades de mentoria iniciaram na idade média com o pensamento escolástico e teve, a partir do século XII, pensadores como Tomás de Aquino, Mestre Eckhart, Dante Alighieri e Marsílio de Pádua. Estes modelos de ensino e suas evoluções durante os séculos, como por exemplo o Método Monitoral de Lancaster, viriam a ser os primórdios das monitorias acadêmicas como se conhece hoje.

No Brasil os alunos monitores historicamente foram utilizados na estratégia de classes multisseriadas, assim, a experiência dos alunos mais avançados ajudaria com àqueles das séries iniciais. Já nas universidades brasileiras, somente na década de 1960, após a Lei nº 5.540 de 28 de novembro de 1968, que fixava normas de organização e funcionamento do ensino superior (Lei da Reformulação do Ensino Superior)[5] é que a figura do aluno monitor no art. 41 foi oficialmente instituída.

Diante do contexto atual da educação brasileira, que ainda não oferece uma formação de qualidade na Educação Básica e no Ensino Médio/Técnico, de maneira que os alunos ao chegarem na universidade possuam um nível semelhante de conhecimento, as instituições de

---

[3] Revisão da pesquisa realizada no Portal de Periódicos CAPES, no dia 1 de setembro de 2020. https://www.periodicos.capes.gov.br/ com acesso remoto via CAFe para acesso aos alunos da UFC-CE.

[4] De acordo com https://scholar.google.com/citations?user=VG9Ptc8AAAAJ;hl=pt-BR;oi=sra , acessado no dia 17 de junho de 2020.

[5] Portal do Planalto Federal: [http://www.planalto.gov.br/ccivil_03/Leis/L5540.htm], acessado em 5 de setembro de 2020.

ensino superior são motivadas a investirem em programas de monitorias acadêmicas, visto que tem demonstrado resultados úteis nas disciplinas com altos índices de reprovações. Deste modo, cada instituição elabora e respalda em leis e resoluções próprias como se dará a regulamentação de seus programas de monitoria em disciplinas específicas.

5.3 Etapa 3: Definir Técnicas para Estudo do Domínio

Após o estudo realizado com a Pesquisa Bibliográfica e abordado os principais pontos relevantes a este estudo no capítulo da Fundamentação Teórica, pode se definir as técnicas adequadas para propor o modelo de domínio como sendo:

1) Modelagem de domínio (a ser desenvolvido em etapas posteriores): utilizando diagramas UML, propõe-se a descrever um design orientado pelo domínio, utilizando os princípios do DDD (contextos delimitados e linguagem onipresente) descritos na Fundamentação Teórica;
2) Avaliação de artefatos (Documento de Especificação de Requisitos, Diagramas, Protótipos, Entrevistas com usuários etc.) do projeto PAE para compreender os requisitos identificados pelos desenvolvedores (ver seção 5.4.2);
3) Pesquisa com usuários para compreender seu envolvimento com a atividade de monitoria acadêmica durante sua graduação (ver seção 5.4.3).

5.4. Etapa 4: Estudo de Caso

Para melhor compreensão, esta seção foi dividida nos pontos subsequentes de maneira a apresentar o ambiente onde foi realizado o estudo de caso e, logo após isso, apresentar os resultados pertinentes à monitoria acadêmica.

5.4.1 Ambiente do Estudo de Caso

Utilizando os artefatos (Documento de Especificação de Requisitos, Diagramas, Protótipos, Entrevistas com usuários etc.) produzidos no Projeto de Apoio ao Ensino (PAE) - que é um programa de extensão da Universidade Federal do Ceará Campus de Russas iniciado em 2020 – realiza-se um estudo de caso para obter dados relevantes a pesquisa exploratória e a

partir disso, conseguir aplicar os princípios fundamentais de modelagem de domínio abordados neste trabalho científico.

O projeto PAE atualmente conta com seis alunos pesquisadores/desenvolvedores, uma professora colaboradora e uma professora orientadora que realiza reuniões semanais para debater os resultados de pesquisas relacionadas ao levantamento de requisitos, validação e prototipação.

A equipe do projeto seguiu durante o ano de 2020 um processo de desenvolvimento no modelo cascata. Para Pressman (2011), o modelo cascata, chamado de ciclo de vida clássico, segue uma abordagem sequencial e sistemática, começando com o levantamento de todas as necessidades, avançando por fases de planejamento, modelagem e construção (desenvolvimento) utilizando *feedback* para validar cada fase do desenvolvimento e permitir seguir para a fase seguinte. Conforme as etapas eram realizadas, artefatos como documento de requisitos, especificação de casos de uso e matriz de rastreabilidade foram desenvolvidos. Contudo, buscando melhorias e amadurecimento em seu processo de desenvolvimento, em 2021 a equipe resolveu utilizar-se de um processo ágil, adotando assim o SCRUM. Para Sommerville (2018), a metodologia ágil permite que o time de desenvolvimento se concentre no próprio software em vez da documentação, reduzindo assim a burocracia do processo ao evitar o trabalho com valor duvidoso no longo prazo ao eliminar a documentação que provavelmente não será utilizada.

Seguindo o processo descrito (Figura 10: Modelo de Processo de Pesquisa do Estudo de Caso), realiza-se a obtenção dos artefatos produzidos no projeto pesquisado. Após isso, iniciou-se o processo de catalogar todos os requisitos funcionais desta documentação (ver seção 5.4.2). Em momento oportuno será aplicado às demais etapas previstas.

5.4.2 Atividade: Selecionar artefatos - Artefatos de Requisitos do Projeto PAE

Durante a fase de elicitação de requisitos, feito pelos desenvolvedores no projeto PAE, foram documentados requisitos funcionais (RF) e não-funcionais. Para este estudo consideram-se apenas os requisitos funcionais descritos no Quadro 01.

Quadro 1: Lista dos Requisitos Funcionais do projeto PAE

LISTA DOS REQUISITOS FUNCIONAIS							
Identificador	Título do Requisito	Atores Participantes					Prioridade
		Aluno	Monitor	Professor	Administrador	Sistema	
RF001	Manter Cursos				■		Essencial
RF002	Manter Disciplinas				■		Essencial
RF003	Manter Professores				■		Essencial
RF004	Manter Alunos			■			Essencial
RF005	Manter Monitores			■			Essencial
RF006	Manter Monitorias			■			Essencial
RF007	Manter Turmas			■			Essencial
RF008	Importar Planilha dos Alunos			■			Essencial
RF009	Manter Dúvida	■	■	■			Indefinido
RF010	Responder Dúvida	■	■	■			Indefinido
RF011	Buscar Dúvida	■	■	■			Indefinido
RF012	Filtrar Dúvidas					■	Indefinido
RF013	Seguir Dúvida	■	■	■			Indefinido
RF014	Deixar de Seguir Dúvida	■	■	■			Indefinido
RF015	Fechar Dúvida	■	■	■			Indefinido
RF016	Votar na Dúvida	■	■	■			Indefinido
RF017	Votar na Resposta	■	■	■			Indefinido
RF018	Confirmar Resposta Correta	■	■	■			Indefinido
RF019	Desmarcar Resposta Correta Confirmada	■	■	■			Indefinido
RF020	Solicitar um Agendamento	■					Importante
RF021	Visualizar Agendamentos	■					Importante
RF022	Filtrar Agendamentos					■	Importante

Fonte: Projeto PAE, adaptado pelo autor.

Outros requisitos funcionais importantes foram identificados e elicitados pelo projeto PAE e podem ser observados nos anexos.

5.4.3 Atividade Analisar artefatos: Pesquisa de Interação nas Monitorias

Para a atividade de análise dos artefatos é necessário um bom entendimento do domínio da ferramenta em desenvolvimento pela equipe do projeto PAE, objeto do estudo de caso. Para tanto, realiza-se uma pesquisa online entre os dias 29 de abril à 08 de maio de 2020, onde foram convidados os 371 alunos[6] do curso de Engenharia de Software por meio do SIGAA (Sistema Integrado de Gestão de Atividades Acadêmicas), que é a plataforma oficial para comunicação da IES.

A pesquisa obteve apenas 33 respostas, muito abaixo do esperado. Ainda é importante ressaltar que os dados podem conter erros significativos. Desta forma, eles são apresentados a seguir como sendo útil para uma reflexão e possível motivação para estudos futuros já que se trata de informações não probabilísticas, obtidas por uma amostragem por conveniência, que é um critério estatístico. Podendo assim, serem contrastados posteriormente em trabalhos futuros.

Os dados da Tabela 02, representando a frequência simples, onde: (*ni*) representa a frequência absoluta, (*fi*) a frequência relativa e (*fac*) a frequência acumulada; apresenta uma visão geral dos alunos participantes da pesquisa que pode ser representada no Gráfico 01. Na coluna Semestres é apresentado o ano seguido do indicativo 1 ou 2 para representar o período semestral do ano correspondente. Com os dados obtidos é possível verificar que cerca de 57% dos participantes são alunos que ingressaram no curso até 2018.1. A maior adesão de participação é de alunos pertencentes aos semestres 2016.2 e 2019.1 com 18% em cada semestre.

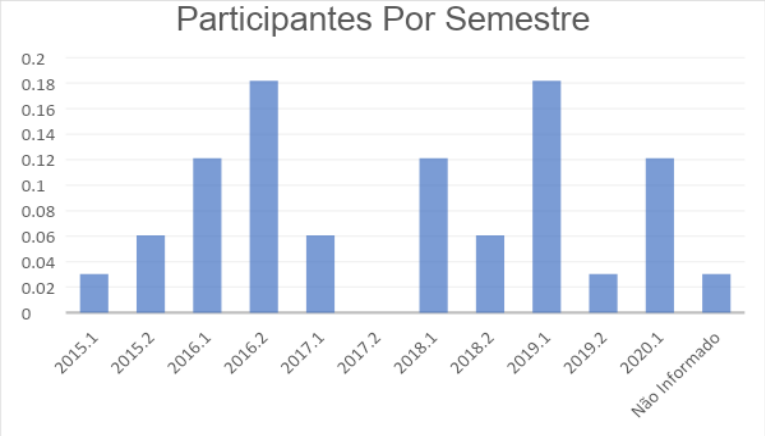
[6] Quantidade de alunos com matrícula ativa no dia 01 de maio de 2020 segundo a Coordenação do Curso de Engenharia de Software da Universidade Federal do Ceará Campus de Russas.

Tabela 2: Alunos Participantes Por Semestre de entrada no Curso.

Semestres	Participantes da Pesquisa (n_i)	f_i	f_{ac}
2015.1	1	0,03	0,03
2015.2	2	0,06	0,09
2016.1	4	0,12	0,21
2016.2	6	0,18	0,39
2017.1	2	0,06	0,45
2017.2	-	-	-
2018.1	4	0,12	0,57
2018.2	2	0,06	0,63
2019.1	6	0,18	0,81
2019.2	1	0,03	0,84
2020.1	4	0,12	0,96
Não Informado	1	0,03	1
Total (n)	33	1	

Fonte: O Autor.

Gráfico 1: Gráfico de participantes da pesquisa separados por semestre



Fonte: O Autor.

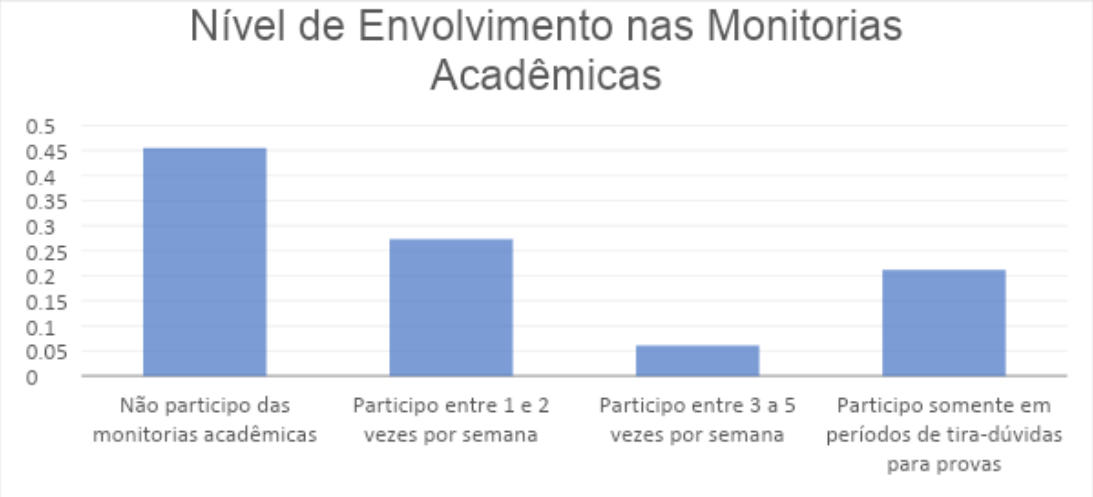
Dos alunos, cerca de 45% informaram que não participam das monitorias acadêmicas como pode ser visto na Tabela 04. Este dado, apresentado no Gráfico 02, é bastante relevante e necessita de maior investigação para entender os motivos que levam a essa alta taxa de alunos que não possui em suas rotinas acadêmicas a participação nas monitorias, visto que não é o foco principal deste trabalho.

Tabela 3: Nível de Participação nas monitorias acadêmicas

Vezes por Semana	**Participantes da Pesquisa (n_i)**	f_i
Não participo das monitorias acadêmicas	15	0,45
Participo entre 1 e 2 vezes por semana	9	0,27
Participo entre 3 a 5 vezes por semana	2	0,06
Participo somente em períodos de tira-dúvidas para provas	7	0,21
Total (n)	33	1

Fonte: O Autor.

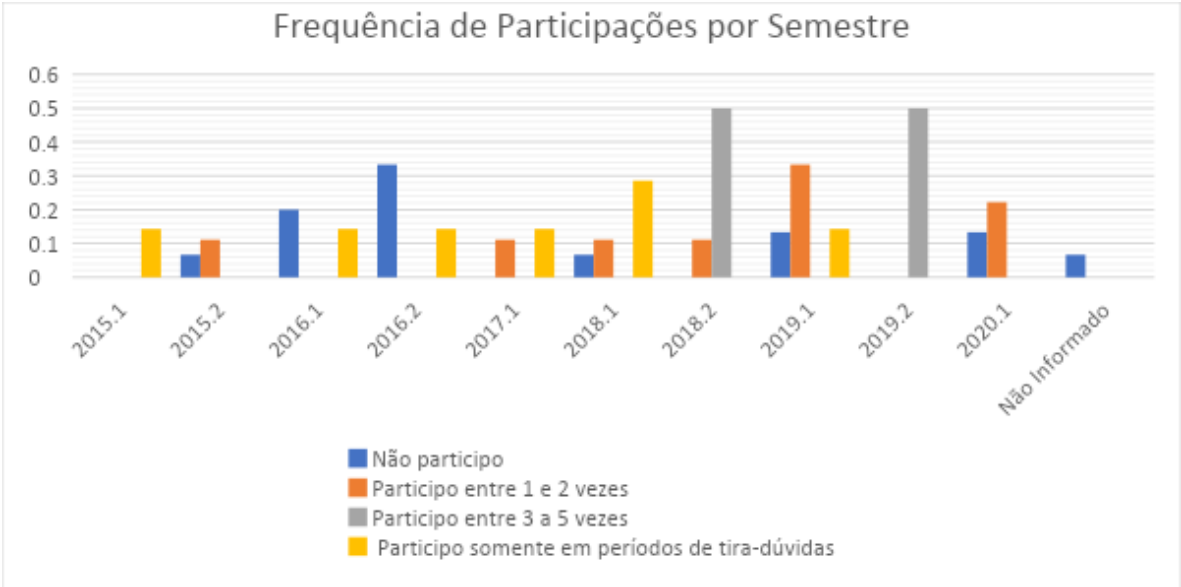
Gráfico 2: Nível de envolvimento nas monitorias acadêmicas



Fonte: O Autor.

O Gráfico 03, apresenta a frequência de participação dos alunos de acordo com cada semestre.

Gráfico 3: Frequência de Participações na monitoria acadêmica por semestre de ingresso no Curso.



Fonte: O Autor.

Os dados obtidos com a pesquisa podem motivar trabalhos futuros que investiguem como está sendo a participação do aluno após o desenvolvimento e implantação de uma ferramenta que auxilia no acompanhamento das atividades da monitoria acadêmica. É importante que esta ferramenta possua meios para gerar dados que possam ser utilizados na construção de indicadores do envolvimento do aluno na monitoria. Observando os requisitos funcionais adequados à esta finalidade durante o estudo de caso, pode-se coletar dados que contribui na identificação dos motivos para que alunos não tenham a rotina de procurar a monitoria, investigando também a eficiência da metodologia utilizada pelos monitores.

Outro ponto importante do estudo junto a possíveis usuários, é a possibilidade de entender o perfil do aluno que busca ajuda na monitoria acadêmica. Assim, a equipe de desenvolvimento pode aprofundar ainda mais no entendimento das necessidades destes usuários, visto que, em cada período letivo, novos desafios surgem no decorrer do avanço acadêmico.

6. PROPOSTA DE MODELAGEM DO DOMÍNIO

Após o estudo teórico para entendimento da importância dos requisitos, das técnicas de modelagem de domínios e do contexto que os envolvem, dedica-se agora a aplicar este conhecimento na especificação de requisitos de uma ferramenta para monitoria acadêmica. Com isso, mostra-se como o estudo do domínio de uma aplicação contribui na especificação dos requisitos, elaborando, assim, uma documentação comum e de fácil entendimento por todos da equipe.

Neste capítulo serão abordados passos importantes para a especificação destes requisitos: 1) Descrição do problema; 2) Especificando a Interação dos Atores com o software; 3) História de Usuários e Critérios de Aceitação; 4) Descrevendo a arquitetura do software e; 5) Definição de Casos de Testes.

6.1 Descrição do Problema

Quando se realiza os primeiros contatos com o cliente de um projeto, costuma-se ouvir uma descrição ou narrativa de tudo àquilo que está motivando a buscar uma solução à questão do seu problema. Essa descrição ou narrativa geralmente contém informações que são o *core business* do seu negócio, que para o cliente são óbvias, mas para uma outra pessoa que não possui o conhecimento do negócio pode ser algo confuso, o que poderia levar a uma interpretação errada de suas necessidades.

Desta forma, no Quadro 02, inicia-se com a descrição da principal problemática que envolve a monitoria acadêmica, destacando pontos importantes e, a partir deles, mantêm-se o foco nas reais necessidades, fugindo das complexidades e de requisitos que não são prioritários e/ou necessários ao projeto.

Quadro 2: Descrição da problemática que envolve a monitoria acadêmica.

PROBLEMÁTICA
A **MONITORIA ACADÊMICA** é um espaço onde os **alunos** podem **TIRAR DÚVIDAS SOBRE OS CONTEÚDOS** das **disciplinas** em horários estabelecidos pelo **monitor**. O monitor estabelece seus **horários de atendimento** e disponibiliza canais de **comunicação e avisos**. Mas devido a extensa grade de horários das disciplinas, muitos alunos não conseguem se manter informados de **quando** e **onde** será o atendimento do monitor e por isso poucos mantém a rotina de ir até a monitoria, fazendo com que busquem **ajuda** por e-mail ou por **mensagens** em aplicativos de bate-papo. Sendo assim, criar mecanismos que mantenham os alunos atualizados e que possibilitem maior interação com os monitores permitirá um maior alcance da quantidade de alunos atendidos, possibilitando uma melhora no **desempenho** acadêmico.

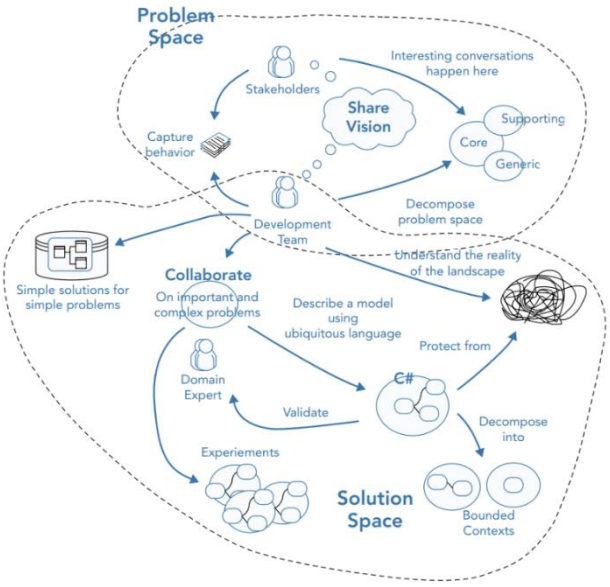
Fonte: Elaborado pelo autor com base na documentação do projeto PAE.

Como destacado no texto, tem-se que o domínio principal é a “Monitoria acadêmica”. Além disso, outros termos destacados são importantes pois demonstram como o domínio funciona. Estes são subdomínios que envolvem o domínio principal.

Para Vernon (2016), em um sentido amplo, o domínio é o que a organização faz e o mundo que ela cria, possuindo várias maneiras de fazer as coisas e métodos para realizar suas operações. Este domínio pode se referir a uma totalidade do negócio ou apenas a uma área básica. Assim, como há diferentes funções, é vantajoso pensar sobre cada uma delas separadamente, observando que existem “subdomínios de suporte” (são essenciais ao negócio e relativamente especializados, mas não básicos) e “subdomínios genéricos” (não captura algo especial se não for necessário para a solução geral ao negócio).

Vale ressaltar que não é preciso que sejam abordados todos os pontos e questões que envolvem os interessados nessa aplicação em um único momento, nem definir todos os pontos de problemas, muito menos descrever uma documentação coesa, ampla, que detalhe todos os casos. Precisa-se na verdade é compreender o principal “ponto de dor” para desenvolver algo que esteja gerando valor ao cliente. Para Millet e Tune (2015), existem várias etapas de aplicação dos princípios do DDD durante o desenvolvimento que permitirá melhorias através de refatoração (Figura 11). Isso reforça o pensamento de equipes que trabalham com desenvolvimento ágil de software.

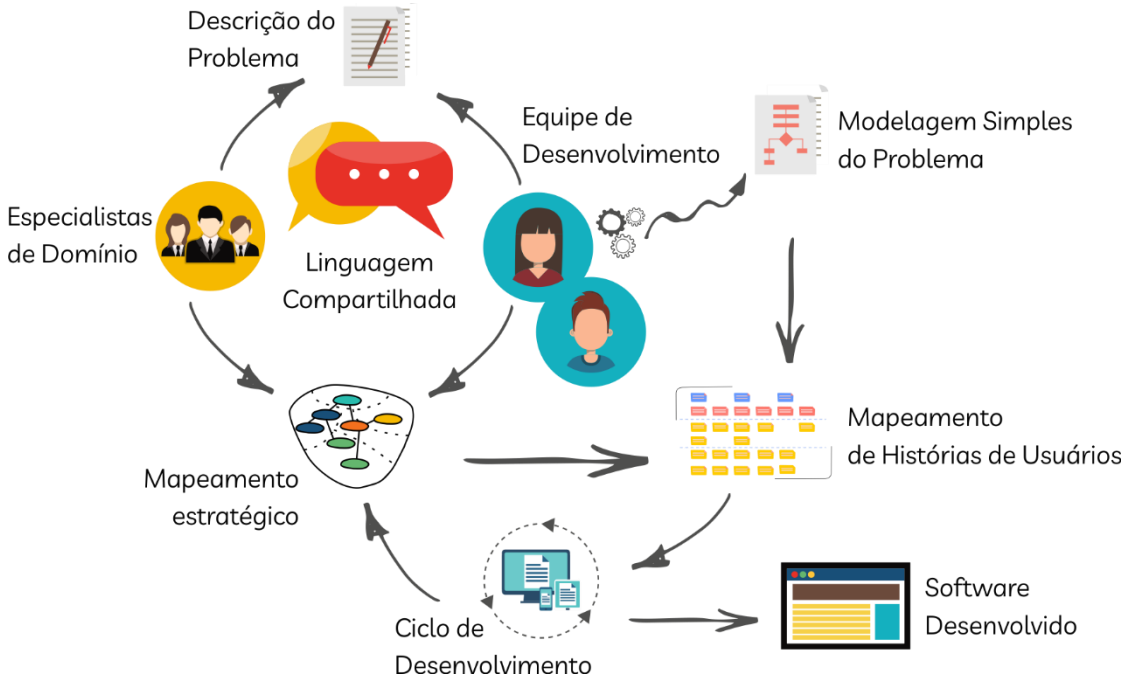
Figura 11Etapas de aplicação dos princípios e práticas do DDD.



Fonte: Millet e Tune (2015) p. 141

De modo geral, na Figura 12, as etapas neste trabalho seguem um fluxo de desenvolvimento ágil de software, com a participação dos especialistas de domínio e da equipe de desenvolvimento em todas as fases do processo até a entrega final do produto. Desta forma, entende-se que a partir da linguagem compartilhada e do entendimento do problema, é possível juntamente com os especialistas de negócio mapear o domínio para que a equipe de desenvolvimento consiga desenvolver as histórias de usuário e demais documentos que serão importantes em todo o ciclo de desenvolvimento utilizando uma modelagem simplificada.

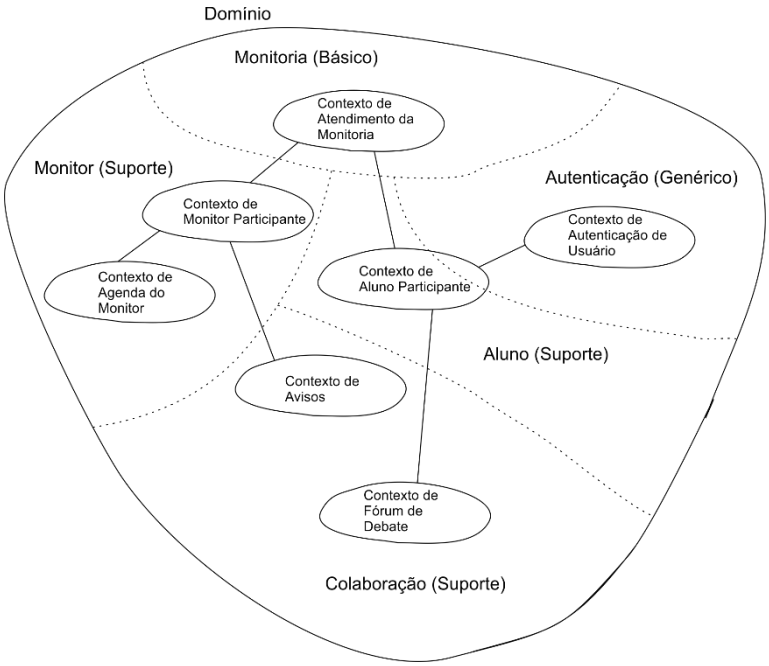
Figura 12: Fluxo de etapas da modelagem e desenvolvimento aplicado neste trabalho



Fonte: Desenvolvido pelo autor.

Logo, com os detalhes da descrição do problema e agrupando os subdomínios e seus contextos, é possível criar uma modelagem estratégica mostrando como funciona o negócio como demonstrado na Figura 13.

Figura 13: modelagem estratégica do negócio



Fonte: elaborado pelo autor

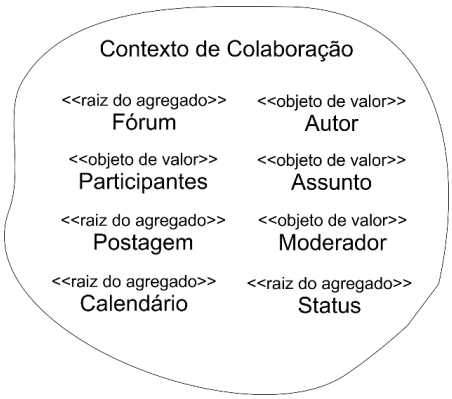
Esta modelagem não necessariamente precisa ser uma visão final do entendimento do negócio. Por seguir uma abordagem ágil de desenvolvimento de software, a refatoração da modelagem pode ocorrer a cada interação e novo entendimento do negócio. Segundo Millett e Tune (2015), manter a complexidade da solução do software o mais baixo possível através dessa contínua evolução, mantendo a simplicidade do design à medida que as iterações mudam, evoluindo para atender aos novos comportamentos das partes interessadas é o objetivo geral de práticas como o DDD.

Iniciar o processo de desenvolvimento com essa abordagem já permite que todos da equipe compreendam cada parte, diminuindo suas complexidades. Entendendo como o negócio funciona pode-se então visualizar contextos que possuem elementos próprios com suas próprias linguagens. Estes contextos estão repletos de objetos de valor, entidades, agregados etc. Por exemplo na Figura 14 e 15, visualiza-se com mais detalhes o contexto de Atendimento e o contexto de colaboração cada um contendo sua própria linguagem.

Figura 14: contexto de atendimento contendo o que existe dentro de seus limites (pode existir outros elementos)

Fonte: elaborado pelo autor

Figura 15: contexto de colaboração contendo o que existe dentro de seus limites (pode existir outros elementos)



Fonte: elaborado pelo autor

6.2 Especificando a Interação dos Atores com o Software

A atividade de selecionar artefatos (5.4.2) apresentou uma alta complexidade entre os atores participantes do sistema, que podem provocar erros típicos na descrição dos requisitos e uma falta de clareza do *core business* da aplicação. Devido à falta de clareza, verifica-se inúmeros requisitos que demonstram preocupações conflitantes, contendo vários modelos lógicos que precisam crescer demais para acomodar novas funcionalidades, impedindo o progresso de cada um dos outros modelos e principalmente o atraso por parte da equipe. É exatamente isso que pode acontece quando as preocupações de software não estão claramente separadas (VERNON, 2016).

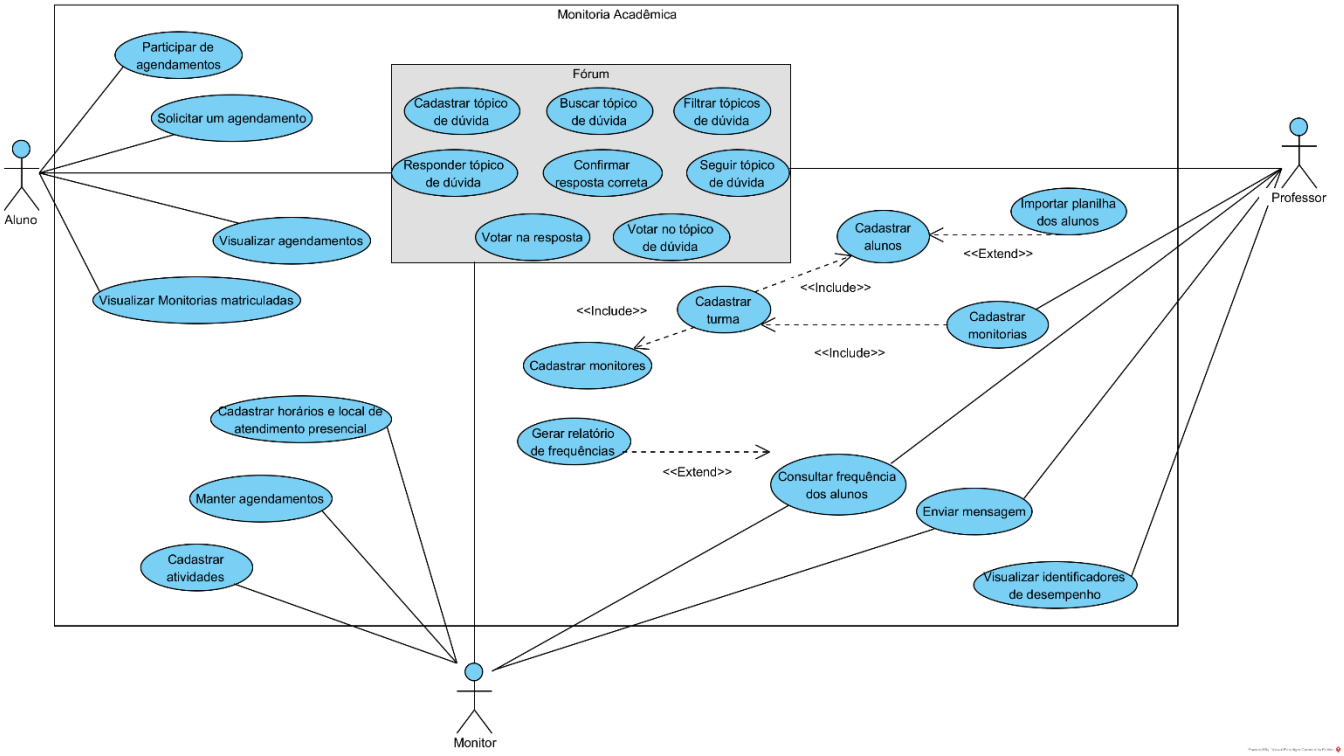
Ao utilizar o diagrama de Caso de Uso para compreender os atores inseridos no contexto da ferramenta com foco na principal necessidade da monitoria busca-se: atender as demandas, sanar dúvidas e possibilitar o debate entre os alunos.

Os dois atores principais envolvidos: alunos e monitores – neste caso, monitores também são alunos selecionados por professores para o suporte da disciplina. O ator “professor” mantém relacionamento com o ator “monitor” – esclarecendo as dúvidas e orientando nos assuntos da disciplina. Este comportamento deveria ter sido analisado como uma separação lógica pertencendo a um outro domínio para que não tornasse esta aplicação tão complexa conforme surgia novos requisitos.

A Figura 16 apresenta uma visão geral, da equipe do projeto, em como estes atores se relacionam com algumas funcionalidades dos requisitos funcionais definidas como essenciais

e importantes. Note que a complexidade em agrupar os casos de uso – que pertencem a diversos contextos – dificulta a compreensão, conduz a erros e não proporciona a definição de limites dos objetivos de negócio.

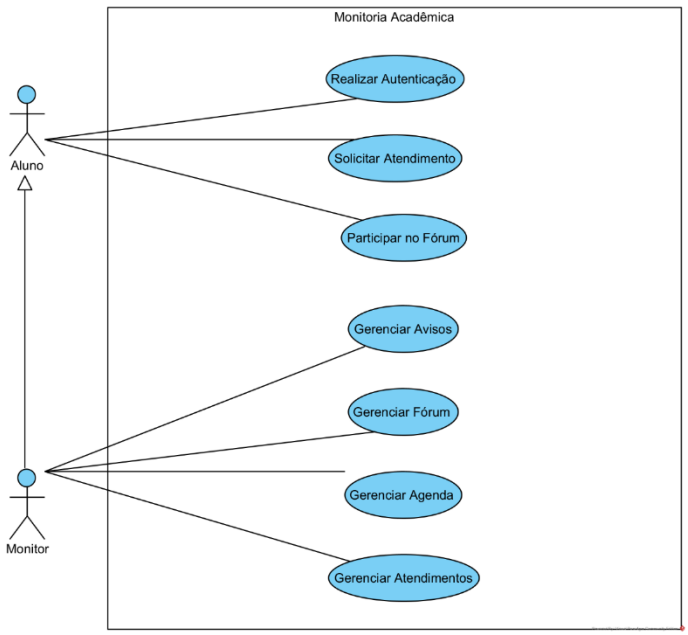
Figura 16: Visão geral do relacionamento de atores com casos de uso essenciais e importantes



Fonte: baseado na Documentação do projeto.

Na Figura 17, são apresentados os especialistas do domínio a partir de suas visões externas mais objetivas do comportamento em seus contextos, abordando através de um diagrama de Caso de Uso os elementos essenciais do sistema refletidos no mapeamento estratégico (ver Figura 13).

Figura 17: diagrama de Caso de Uso

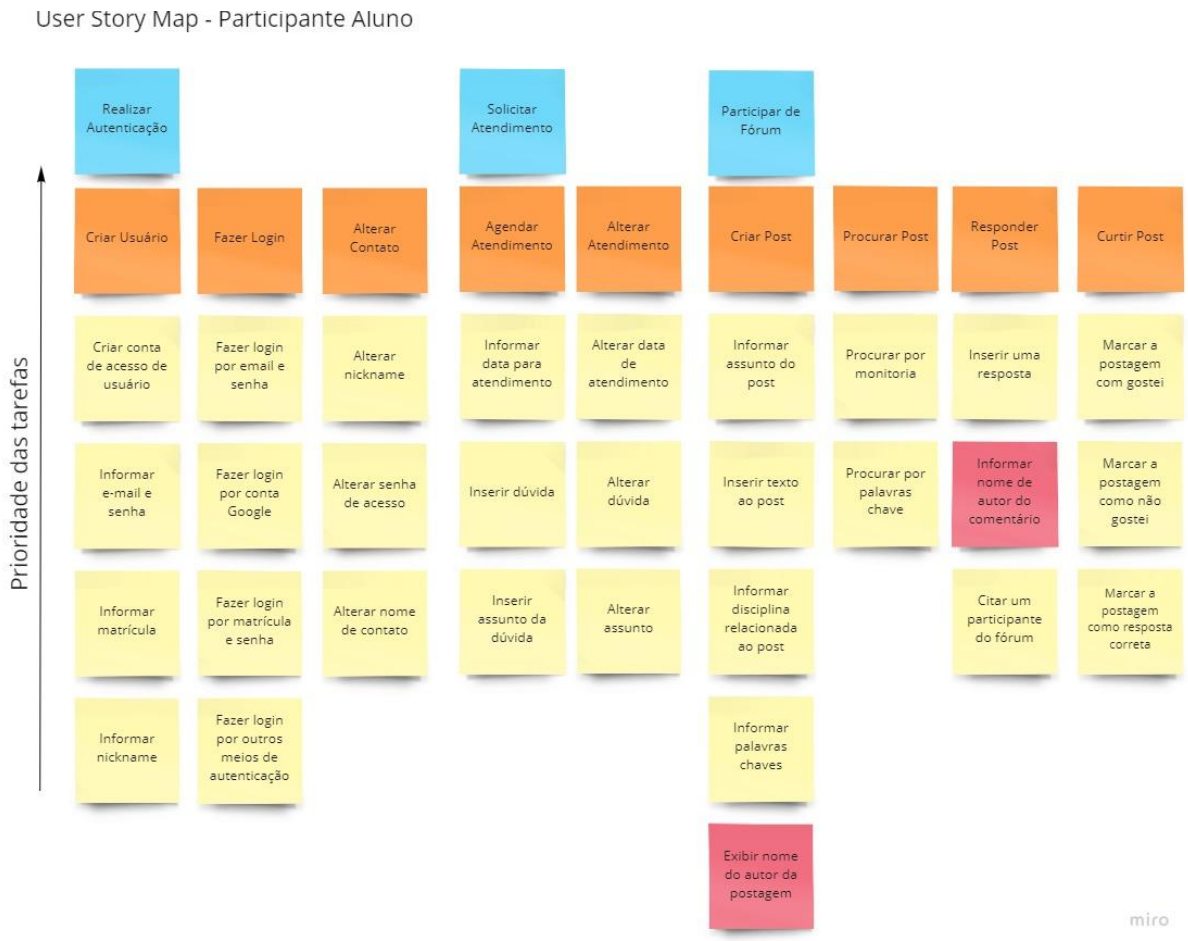


Fonte: elaborado pelo autor

A partir desta visão geral, podem ser identificadas as tarefas de usuários que se tornarão histórias de usuário a partir de técnicas de *User Story Mapping* (mapeamento de história de usuário) por ser uma técnica de fácil utilização e aceita no contexto ágil de desenvolvimento de software. Não é necessário criar um mapeamento único contendo todos os atores e histórias envolvidas. Para alcançar melhores resultados, pode ser criado um mapeamento para cada ator. Assim, é possível identificar os diferentes contextos sem inserir tantas informações que poderão causar erros no momento de priorizar as tarefas para cada *sprint* de desenvolvimento.

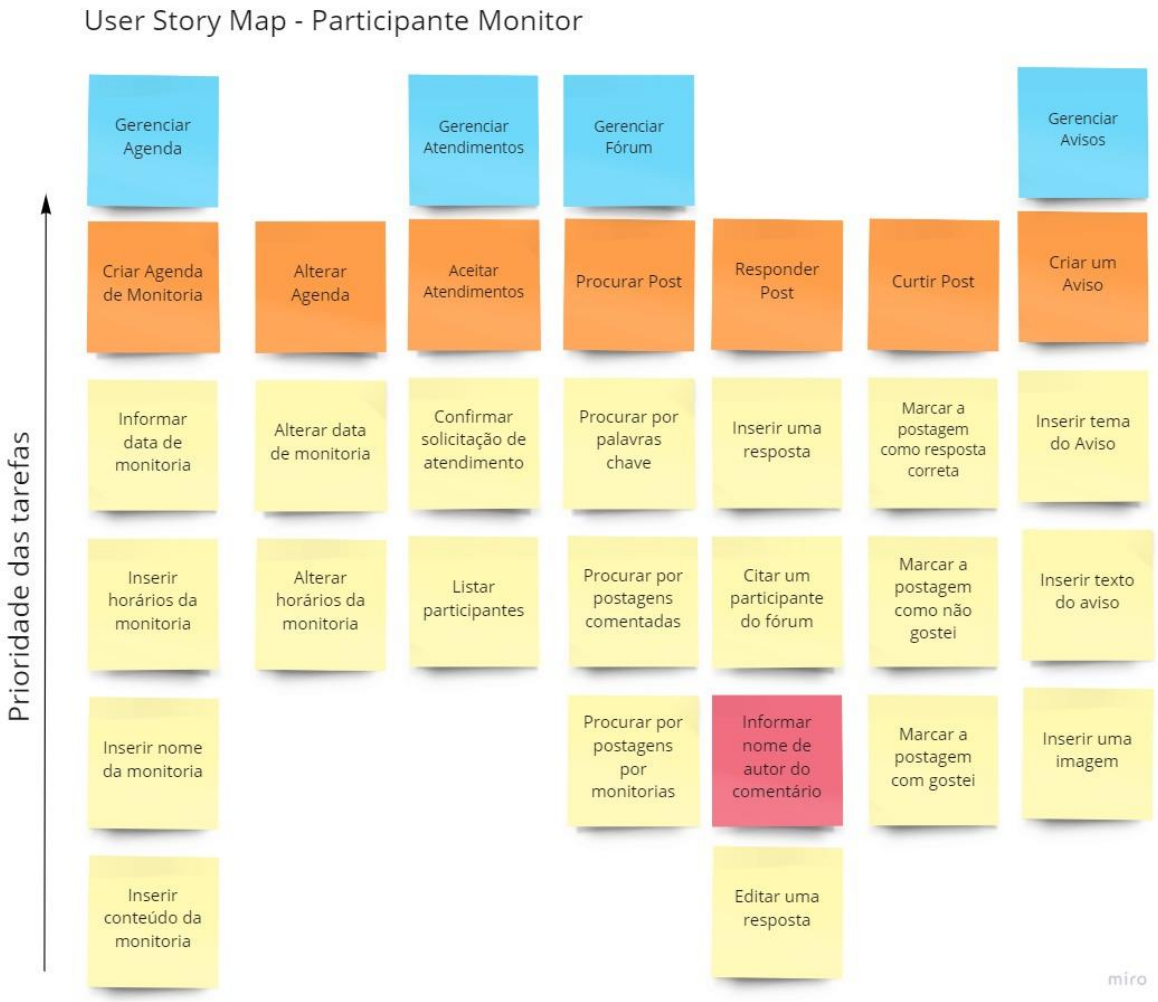
A seguir, nas Figuras 18 e 19, são demonstrados dois exemplos de *User Story Mapping* que envolvem os atores aluno e monitor. Conforme o entendimento vai sendo ampliado, novas histórias poderão ser mapeadas.

Figura 18: Mapa de história de usuário do participante aluno.



Fonte: Elaborada pelo autor.

Figura 19: Mapa de história de usuário do participante monitor



Fonte: Elaborada pelo autor.

Com o mapeamento realizado, é possível então definir as histórias de usuário a partir da prioridade das tarefas, onde quanto mais alto estas prioridades, mais valor está sendo entregado ao projeto.

6.3 História de Usuários e Critérios de Aceitação

Após o mapeamento de história de usuário, é possível organizá-los através de épicos, que são histórias de usuário que são grandes demais e que podem ser divididas em tarefas menores, definindo os critérios de aceitação para cada uma. Os Quadros 03, 04, 05 e 06, apresentam algumas destas histórias de usuários e os critérios de aceitação daquelas que possuem maior prioridade ao desenvolvimento do projeto.

Quadro 3: Exemplos de Histórias de Usuário com seus critérios de aceitação baseados no comportamento de gerenciar agenda.

Épico Monitor: Como monitor desejo gerenciar a agenda de monitorias para manter os participantes informados.	
História de Usuário	Critérios de Aceite
Como monitor, eu quero informar uma data na agenda para informar o dia e a hora da monitoria.	- Dado que o monitor necessita informar os dias de seu atendimento na monitoria, quando acessar a agenda deve informar um horário e dia de sua monitoria.
Como monitor, eu quero informar o nome da monitoria para possibilitar a identificação pelos participantes.	- Dado que o monitor precisa criar uma monitoria, quando preencher o formulário de criação, deve informar o nome da monitoria que ele vai ministrar.
Como monitor, eu quero inserir um tema na monitoria para possibilitar que os participantes entendam o conteúdo que será abordado.	- Dado que o monitor deseja informar o conteúdo da monitoria, quando estiver preenchendo o formulário de criação deve inserir um tema do conteúdo para ser identificado pelos participantes.

Fonte: Elaborado pelo autor.

Quadro 4: Exemplos de Histórias de Usuário com seus critérios de aceitação baseados no comportamento de solicitar atendimento na monitoria.

Épico Aluno: Como aluno desejo solicitar atendimento para a monitoria acadêmica.	
História de Usuário	Critérios de Aceite
Como aluno, eu quero informar uma data de agendamento para participar de uma aula de monitoria.	- Dado que o aluno necessita de um atendimento na monitoria, quando acessa a agenda de monitorias do mês, deve selecionar um dia para continuar a sua solicitação de atendimento. - Dado que o aluno acessa a agenda e não possui datas disponíveis na agenda, deve ser apresentado uma mensagem informando “monitorias indisponíveis”.
Como aluno, eu quero informar um assunto para facilitar a organização do atendimento.	- Dado que o aluno necessita de um atendimento na monitoria, quando preencher o formulário de solicitação de atendimento deve informar o assunto geral do pedido.
Como aluno, eu necessito inserir uma dúvida para que o monitor possa me ajudar sobre este conteúdo.	- Dado que o aluno necessita de um atendimento na monitoria, quando estiver preenchendo o formulário de solicitação de atendimento pode inserir uma dúvida sobre um determinado conteúdo.

Fonte: Elaborado pelo autor.

Quadro 5: Exemplos de Histórias de Usuário com seus critérios de aceitação baseados no comportamento de participar do fórum

Épico Aluno: Como aluno desejo participar do fórum.	
História de Usuário	Critérios de Aceite
Como aluno, eu quero participar do fórum de discussão para compartilhar conteúdos e dúvidas para que outros alunos possam debater sobre um tema proposto.	- Dado que o aluno acessa o sistema, quando está devidamente autenticado pode participar de fóruns de debate sobre os temas cadastrados. - Dado que o aluno acessa o sistema, quando acessa o fórum pode ler postagens de outros participantes.
Como aluno, eu quero informar um tema de postagem no fórum para que outros alunos possam identificar minha postagem.	- Dado que o aluno deseja criar uma postagem, deve informar qual o tema geral de sua postagem.
Como aluno, eu quero inserir um conteúdo na postagem, para que outros alunos possam ler e interagir.	- Dado que o aluno deseja criar um post, quando estiver no formulário do fórum, deve inserir um conteúdo para ser lido pelos demais participantes.
Como aluno, eu quero comentar postagens de outros participantes para interagir com o debate no fórum.	- Dado que o aluno deseja interagir com outras postagens, quando selecionar uma postagem cadastrada, pode inserir um comentário a respeito do tema e conteúdo do post.

Fonte: Elaborado pelo autor.

Quadro 6: Exemplos de Histórias de Usuário com seus critérios de aceitação baseados no comportamento de gerenciar fórum

Épico monitor: Como aluno desejo gerenciar o fórum.	
História de Usuário	Critérios de Aceite
Como monitor, eu quero procurar postagens no fórum para leitura.	- Dado que o monitor deseja ler as postagens, quando estiver no formulário de pesquisa, deve informar uma palavra chave que identifica a postagem. - Dado que o monitor deseja ler as postagens comentadas, quando estiver no formulário de pesquisa, deve buscar por postagens comentadas.
Como monitor, eu quero responder uma postagem no fórum para interagir com outros participantes.	- Dado que o monitor deseja interagir com os participantes, quando estiver lendo uma postagem, deve inserir um comentário para responder à postagem no fórum.
Como monitor, eu desejo marcar uma postagem como correta para que os participantes entendam que a postagem soluciona uma dúvida da monitoria.	- Dado que o monitor deseja marcar uma resposta do fórum como correta, quando estiver na postagem, deve selecionar a opção que marca como resposta correta à dúvida da postagem.
Como monitor, eu desejo marcar uma postagem como gostei, para que os participantes visualizem que eu li a postagem.	- Dado que o monitor deseja que os participantes vejam que leu uma postagem, quando estiver em uma postagem no fórum, deve marcar a opção de resposta como gostei.

Fonte: Elaborado pelo autor.

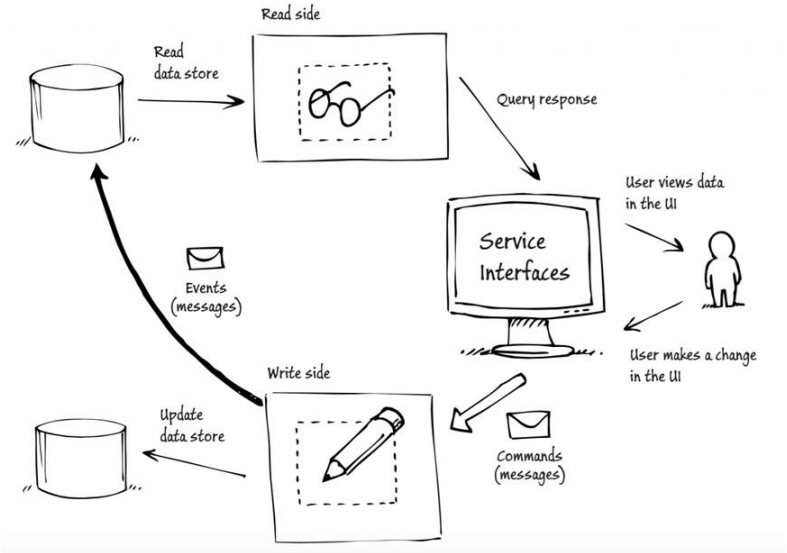
6.4 Descrevendo a Arquitetura do Software

Segundo Vernon (2016), as demandas por qualidade devem orientar o uso dos estilos e padrões arquitetônicos. Dessa forma, a arquitetura é utilizada apenas para aliviar os riscos de falhas, não para aumentar o risco de fracasso usando um estilo ou padrão sem justificativa. Assim, deve-se pensar na arquitetura que consiga mapear diferentes dados de domínio.

Ao longo desta pesquisa, com o surgimento de inúmeros desafios para aproximar a comunidade acadêmica durante o período de crise sanitária[7], evidenciou-se a necessidade de escalabilidade do projeto, para que pudesse atender a inúmeras demandas de consumo de dados com operações de leitura e escrita. Sendo assim, sugere-se a utilização do padrão CQRS – *Command Query Responsability Segregation* (Segregação de Responsabilidades por Comandos e Consultas).

Segundo Millet e Tune (2015), CQRS é simplesmente um padrão arquitetônico onde você pode aplicar princípios de contexto delimitado, separando os modelos de consultas do processamento através de comandos. Ou seja, um modelo é construído para lidar e processar comandos enquanto outro é construído para necessidades de apresentação (Figura 20).

Figura 20: Representação básica do padrão CQRS – separação entre escrever e ler através da modificação do usuário em uma página, resultando em um comando enviado que é sinalizado por meio de um evento.

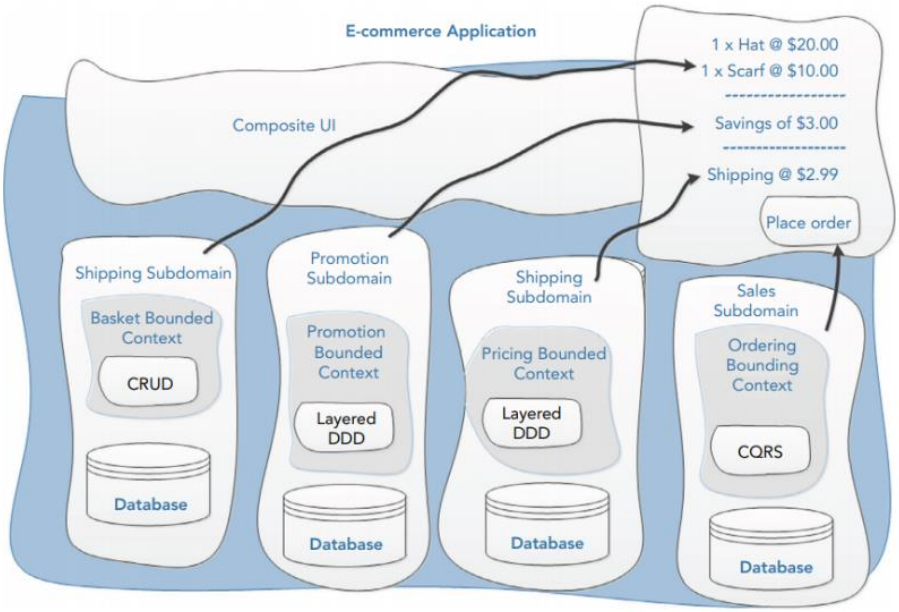


Fonte: Blog Eleven Labs[8]

[7] Crise sanitária causada pela Pandemia de COVID-19 (SARS-CoV-2) iniciada em fevereiro de 2020 e que dura até o momento de escrita e conclusão deste trabalho acadêmico.
[8] CQRS Pattern. Acessado no dia 14 de agosto de 2021: https://blog.eleven-labs.com/en/cqrs-pattern-2/

Em seu site[9], Martin Fowler define como natural a abordagem de CQRS com alguns outros padrões arquitetônicos, sendo adequado para domínios complexos do tipo que se beneficiam do design orientado por domínio (DDD). Devendo ser usado em porções específicas de um contexto delimitado e não no sistema como um todo, pois cada contexto delimitado precisa de suas próprias decisões sobre como deve ser modelado (Figura 21).

Figura 21: Exemplo de aplicação de diferentes padrões arquitetônicos em diferentes contextos delimitados de uma aplicação comercial



Fonte: (MILLET; TUNE, 2015) p. 88

A abordagem utilizando CQRS deve ser implementada em um cenário que esteja claro a necessidade de, entre outras coisas: manter domínios colaborativos com muitos usuários acessando os mesmos dados paralelamente; onde o desempenho de leitura seja ajustado separadamente do desempenho de gravação de dados, especialmente com maiores demandas de leituras do que escrita; onde equipes distintas estão trabalhando no complexo modelo de domínio para gravação e outra equipe se concentra no modelo de leitura e interface com o usuário; onde a falha temporal de um subsistema não afete a disponibilidade dos outros[10].

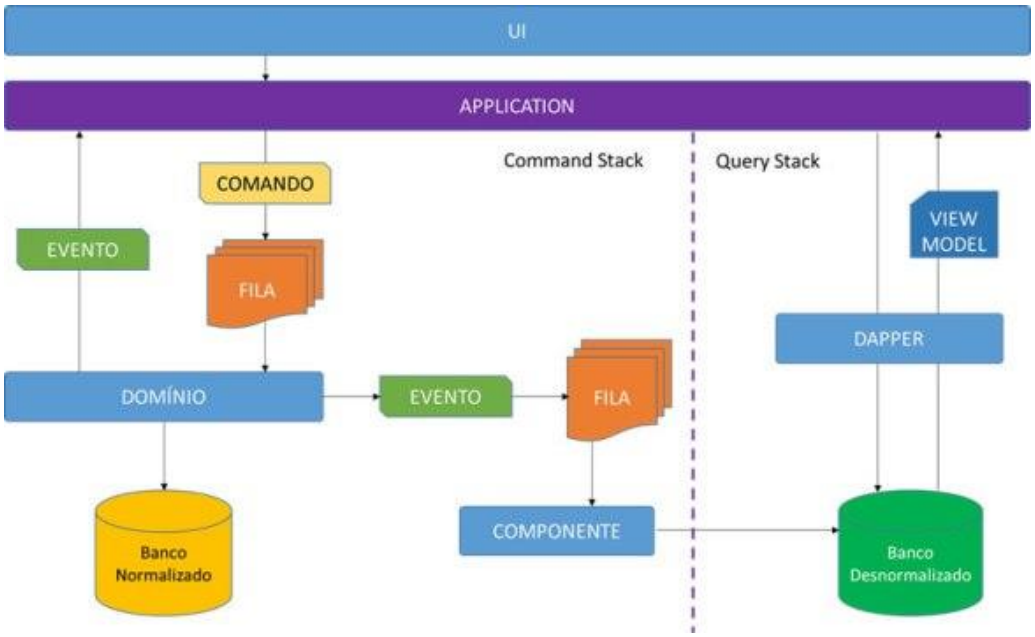
Uma decisão arquitetural de alto nível como apresentado na Figura 22, proporciona a divisão de responsabilidades de contextos específicos, possibilitando separação estrutural dos

[9] CQRS, acessado em 14 de agosto de 2021: https://martinfowler.com/bliki/CQRS.html
[10] O que é o padrão CQRS? - Azure Architecture Center | Microsoft Docs. Acessado em 14 de agosto de 2021: https://docs.microsoft.com/pt-br/azure/architecture/patterns/cqrs

dados, com consultas realizadas de forma síncrona em uma base de dados desnormalizada e gravações assíncronas em banco de dado normalizado partindo isso de uma interação com uma interface pensada em atender as necessidades do usuário.

Assim o projeto consegue manter um alto nível de escalabilidade, atendendo a inúmeras demandas dos usuários e ainda permitindo uma evolução coerente, que possibilita o desenvolvimento de novas funcionalidades e a interação com outros componentes de sistemas.

Figura 22: Modelo design utilizando CQRS com separação dos dados para leitura e escrita



Fonte: CQRS - O que é? Onde aplicar? – Eduardo Pires - Microsoft Regional Director & MVP[11]

[11] CQRS - O que é? Onde aplicar? – Eduardo Pires - Microsoft Regional Director & MVP. Acessado em 04 de agosto de 2021: https://www.eduardopires.net.br/2016/07/cqrs-o-que-e-onde-aplicar/

6.5 Definição de Casos de Teste

Com as histórias de usuários é possível definir casos de testes para orientar a equipe de desenvolvimento. Trabalhando com domínios, fica evidenciado que a equipe está em constante aprendizado das necessidades dos requisitos. Com isso, podem, então, serem descritos casos de testes unitários que complementem as funções ou métodos iniciais para o desenvolvimento do projeto. Os Quadros 07, 08 e 09, mostram exemplos de como são descritos alguns casos de testes para casos de uso.

Quadro 7: Casos de testes de requisitos baseados no caso de uso gerenciar monitoria, pertencente ao ator monitor.

Caso de Uso	Descrição	Tipo de Teste	Valor de Entrada	Resultado Esperado
Gerenciar Agenda	Uma data deve conter horário, dia, mês, ano	Unidade	Campo de data vazio	Este campo é obrigatório para solicitar um atendimento
	Um título de monitoria é criado	Integração	Título com menos de 5 caracteres	Este campo deve conter pelo menos uma palavra com no mínimo 5 caracteres
	No campo de local de atendimento deve constar uma sala	Unidade	O campo de local de atendimento vazio	Este campo é obrigatório para criar um evento de monitoria
	No campo monitor deve conter o nome do responsável pela monitoria	Unidade	O campo monitor está vazio	Este campo é obrigatório para identificação do responsável pela monitoria
	Uma lista de dias de monitoria é criada	Integração	Campos: data, título, local, monitor é preenchido	Uma solicitação de atendimento cadastrada

Fonte: elaborado pelo autor

Quadro 8: Casos de testes de requisitos baseados no caso de uso solicitar atendimento, pertencente ao ator aluno.

Caso de Uso	Descrição	Tipo de Teste	Valor de Entrada	Resultado Esperado
Solicitar Atendimento	Uma data deve conter horário, dia, mês, ano	Unidade	Campo de data vazio	Este campo é obrigatório para solicitar um atendimento
	Um assunto deve conter no mínimo 5 caracteres	Unidade	Assunto com menos de 5 caracteres	Este campo deve conter pelo menos uma palavra com no mínimo 5 caracteres
	No campo de dúvida, deve conter um texto	Unidade	O campo de dúvida vazio	Este campo é obrigatório para criar uma solicitação de atendimento.
	Uma solicitação de atendimento é criada	Integração	Campos: data, assunto e dúvida preenchidos	Uma solicitação de atendimento cadastrada

Fonte: elaborado pelo autor

Quadro 9: Casos de testes de requisitos baseados no caso de uso participar do fórum, pertencente ao ator aluno.

Caso de Uso	Descrição	Tipo de Teste	Valor de Entrada	Resultado Esperado
Participar do Fórum	Uma postagem no fórum deve ter um título	Unidade	Campo de título vazio	Este campo é obrigatório
	Uma postagem deve possuir um conteúdo	Unidade	Campo de conteúdo vazio	Este campo é obrigatório para criação de uma postagem
	Um post no fórum pode ser visualizado	Integração	Pelo menos uma postagem criada no fórum	Uma lista de postagens criadas
	Uma postagem pode ter comentários	Integração	Pelo menos um comentário cadastrado em um post	Uma lista de comentários

Fonte: elaborado pelo autor.

Seguindo a metodologia ágil de desenvolvimento de software, a equipe inicia a codificação com base na descrição dos casos de testes. Este método é conhecido por Desenvolvimento Dirigido por Testes (TDD, *"Test Driven Development"*) na qual se intercalam testes e desenvolvimento de código. Ou seja, é realizado a escrita de um teste para uma função ou método, depois este código é executado e refatorado até que o teste passe corretamente. Este processo de refatoração contribui para manter uma melhor qualidade do software e possibilite acrescentar novo código ao que já existe (SOMMERVILLE, 2018).

7. CONSIDERAÇÕES FINAIS

Através do estudo exploratório é possível compreender um pouco mais da importância que o amplo entendimento do domínio de uma aplicação, com a manutenção de uma linguagem compartilhada entre os especialistas de negócio e a equipe de desenvolvimento, tem para a obtenção e modelagem dos requisitos de software. Aplicando técnicas de design orientado pelo domínio (DDD), a equipe de desenvolvimento consegue focar seus esforços no que é realmente importante e que tem valor ao cliente.

Com a fundamentação teórica obtida pela pesquisa, é possível identificar com mais facilidade a necessidade de se manter uma documentação minimalista para utilização da equipe de desenvolvimento. Isso é importante para que seja possível conseguir manter alinhadas a equipe com as necessidades do projeto, fugindo de complexidades e evitando erros no escopo.

Além disso, verifica-se que o desenvolvimento de software requer a aplicação de ideias estratégicas de modelagem, de modo que tanto os especialistas de negócio como especialistas técnicos consigam superar as complexidades que as diferentes visões impõem ao projeto. Assim, manter o foco em uma documentação ampla pode não satisfazer as necessidades dos envolvidos, já que nem sempre é possível descrever e validar todos os requisitos antes da codificação.

Dessa forma, o estudo de caso realizado com apoio do projeto PAE contribuiu ao aprofundar estas ideias e técnicas de pensar na solução do problema apresentado a partir do conhecimento de domínios e seus contextos. Com o mapeamento das histórias de usuário e uma simples modelagem foi possível criar condições de entendimento do problema que são facilmente descritas nos artefatos e que colaboram com o desenvolvimento fundamentado pelos testes e ainda justificar uma arquitetura coerente que visa atender aos contextos delimitados da aplicação.

Portanto, este estudo tem valor ao demonstrar como os requisitos obtidos pela análise do domínio é eficiente e contribui na comunicação e no desenvolvimento do projeto analisado. Demonstra-se como os princípios e práticas do DDD podem ser utilizados em pequenos projetos para a modelagem do domínio e a construção da linguagem compartilhada entre as equipes para desenvolver uma documentação simples que segue as boas práticas do desenvolvimento ágil de software.

Sugere-se como trabalhos futuros a pesquisa e o estudo dos indicadores que demonstrem os impactos da utilização desta ferramenta por parte da comunidade acadêmica após sua implantação. Também se sugere um estudo a respeito das lições aprendidas por

equipes de desenvolvimento que utilizam os princípios e técnicas da modelagem de domínios, podendo realizar para tanto comparativos com modelos tradicionais de desenvolvimento de software, buscando vantagens e possíveis desvantagens encontradas. Além disso, devido a impossibilidade de realizar uma pesquisa mais ampla para trazer dados da situação atual das monitorias, conclui-se que a eventual presença dos dados poderia contribuir no amplo entendimento do domínio-problema, ao buscar compreender melhor as necessidades dos alunos que são os principais atores e interessados desta aplicação.

REFERÊNCIAS

ALVES, D. R., & MATOS, E. (2019). ***A survey on interaction design in distributed software development***. *18 th Brazilian Symposium on Human Factors in Computing Systems (IHC'19)*.

BARBOSA, S. D. (2010). ***Interação Humano-Computador***. Rio de Janeiro - RJ: Elsevier Editora Ltda.

DUC MINH LE, Duc-Hanh Dang and Viet-Ha Nguyen (2016). **Domain-driven design using meta-attributes: A DSL-based approach**. *Eighth International Conference on Knowledge and Systems Engineering (KSE)*, 2016, pp. 67-72, doi: 10.1109/KSE.2016.7758031.

EVANS, E. (2020). ***Domain-driven Design: Atacando as Complexidades no Coração do Software*** (3 ed.). Rio de Janeiro: Alta Books.

FRISON, L. M. (2016). ***Monitoria: uma modalidade de ensino que potencializa a aprendizagem colaborativa e autorregulada***. *Pro-Posições*, 133-153.

HIRAMA, K. (2011). ***Engenharia de Software: Qualidade e Produtividade com Tecnologia*** (1 ed.). Rio de Janeiro: Elsevier.

MARTIN, ROBERT (2018). **Arquitetura Limpa: O Guia do Artesão para Estruturas e Design de Software / Robert Martin**. Traduzido por Samantha Batista. – Rio de Janeiro: Alta Books, 2018.

MARTINS JUNIOR, J. (2017). ***Como escrever trabalhos de conclusão de curso: instruções para planejar e montar, desenvolver, concluir, redigir e apresentar trabalhos monográficos e artigos.*** (2 ed.). Petrópoles - RJ: Vozes.

MAVIN, A., Mavin, S., Penzenstadler, B., & Venters, C. C. (2019). **Towards an Ontology of Requirements Engineering approaches**. *27th International Requirements Engineering Conference (RE)*.

MERSON, P., & YODER, J. (2020). ***Modeling Microservices with DDD***. *International Conference on Software Architecture Companion (ICSA-C)*.

MILLETT, S., & TUNE, N. (2015). ***Patterns, Principles, and Practices of Domain-Driven Design*** (1 ed.). New York, United States: John Wiley & Sons, Inc.

PAIZANI, G. F. (2011). ***A Filosofia Medieval como produto das relações intermitentes entre Ocidente e Oriente***. *International Congress Of History*.

PRESSMAN, R., & MAXIM, B. (2020). ***Software Engineering: A Practitioner's Approach*** (9 ed.). Mc Graw Hill.

RAHARJANA, I. K., SIAHAAN, D., & FATICHAH, C. (2019). ***User Story Extraction from Online News for Software Requirements Elicitation: A Conceptual Mode***l. *Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia; Information Systems, Universitas Airlangga, Surabaya, Indonesia*.

SANTOS, E. C., Beder, D. M., & Penteado, R. A. (2015). **A Study of Test Techniques for Integration with Domain Driven Design**. *12th International Conference on Information Technology - New Generations*.

SHARP, H., PREECE, J., & ROGERS, Y. (2019). ***Interaction Design: Beyond Human-Computer Interaction*** (5 ed.). Wiley.

SOMMERVILLE, I. (2018). ***Engenharia de Software*** (10 ed.). São Paulo: Pearson Education do Brasil.

VAZQUEZ, C. E., & SIMÕES, G. S. (2016). ***Engenharia de Requisitos: Software Orientado ao Negócio*** (1 ed.). Rio de Janeiro: Brasport Livros e Multimídia Ltda.

VERNON, VAUGHN. (2016). ***Implementando Domain-Driven Design / Vaughn Vernon. – Rio de Janeiro, RJ: Alta Books, 2016.***

APÊNDICE A
INSTRUMENTO DE COLETA DE DADOS DA PESQUISA ONLINE

Pesquisa: Ferramentas de gerenciamento de atividades na Engenharia de Software	
Olá, tudo bem? Obrigado por participar desta pesquisa. Com estes dados buscamos entender quais ferramentas são utilizadas para o gerenciamento de atividades e como elas impactam no processo de aprendizagem. Respeitamos sua privacidade e garantimos que não há meios para identificar você e sua resposta. Responder este questionário leva cerca de 5 minutos.	
Qual o seu ano e semestre de entrada no curso de engenharia de software? Exemplo: 2018.1	Resposta:
*Quais ferramentas você utiliza para gerenciamento de atividades diárias das disciplinas do curso? Selecione as opções que achar necessário ou selecione apenas "Não utilizo nenhuma ferramenta para gerenciar minhas atividades diárias"	□ Não utilizo nenhuma ferramenta para gerenciar minhas atividades diárias □ Trello □ Google Agenda/calendário □ To-Do □ Planilhas □ Pipefy □ WhatsApp □ Outros
Cite quais outras ferramentas de gerenciamento de atividades diárias você utiliza/ou conhece para gerenciar atividades das disciplinas. Informe ferramentas das quais não constam como opção na pergunta anterior.	Resposta:
*As ferramentas para gerenciamento de atividades diárias nas disciplinas do curso contribuem para o processo de aprendizagem. Selecione apenas uma opção. Selecione apenas uma opção.	□ Concordo totalmente □ Concordo parcialmente □ Discordo parcialmente □ Discordo totalmente □ Não concordo nem discordo

*Qual o seu nível de envolvimento nas monitorias acadêmicas? Selecione apenas uma opção.	□ Não participo das monitorias acadêmicas □ Participo entre 1 e 2 vezes por semana □ Participo entre 3 a 5 vezes por semana □ Participo somente em períodos de tira-dúvidas para provas
*Em uma escala de 1 (para totalmente insatisfeito) a 5 (para totalmente satisfeito), qual o seu nível de satisfação com as monitorias acadêmicas? Selecione apenas uma opção	□ 1 □ 2 □ 3 □ 4 □ 5
As monitorias acadêmicas contribuem no processo de aprendizagem dos conteúdos visto em sala de aula. Selecione apenas uma opção	□ Concordo totalmente □ Concordo parcialmente □ Discordo parcialmente □ Discordo totalmente □ Não concordo nem discordo
*O conteúdo das disciplinas retrata assuntos importantes e atuais da área de formação. Selecione apenas uma opção	□ Concordo totalmente □ Concordo parcialmente □ Discordo parcialmente □ Discordo totalmente □ Não concordo nem discordo
*Os professores motivam os alunos a utilizar ferramentas de gerenciamento de atividades em suas disciplinas. Selecione apenas uma opção.	□ Concordo totalmente □ Concordo parcialmente □ Discordo parcialmente □ Discordo totalmente □ Não concordo nem discordo

*Você concorda que é necessário reformular os métodos de ensino para que o uso de ferramentas tecnológicas estejam mais presentes durante as aulas? Selecione apenas uma opção.	□ Concordo totalmente □ Concordo parcialmente □ Discordo parcialmente □ Discordo totalmente □ Não concordo nem discordo
LEIA COM ATENÇÃO O TEXTO ABAIXO PARA RESPONDER A PRÓXIMA PERGUNTA. TEXTO 01:	
Existem diversos métodos de ensino que podem ser empregados para transmitir e gerar conhecimento nos alunos. Um dos mais usados na graduação é o método tradicional, no qual o professor é o sujeito ativo no processo de ensino-aprendizagem, repassando seu conhecimento aos alunos, normalmente por meio de aula teórica. (metodista.br)	
TEXTO 02: Metodologia ativa de aprendizagem é um processo amplo e possui como principal característica a inserção do aluno/estudante como agente principal responsável pela sua aprendizagem, comprometendo-se com seu aprendizado. Wikipédia.	
*Você concorda que metodologias ativas é mais importante hoje no processo de aprendizagem do que os métodos tradicionais de ensino? Responda com base na sua interpretação do Texto 01 e do Texto 02.	□ Concordo totalmente □ Concordo parcialmente □ Discordo parcialmente □ Discordo totalmente □ Não concordo nem discordo
Obrigado por participar desta pesquisa. Para finalizar clique em enviar. (Formulário online do Pipefy);	

ANEXOS

Continuação da Lista dos Requisitos Funcionais do projeto PAE

Lista dos Requisitos Funcionais							
Identificador	**Título do Requisito**	**Atores Participantes**					**Prioridade**
		Aluno	**Monitor**	**Professor**	**Administrador**	**Sistema**	
RF023	Participar de Agendamentos	■					Importante
RF024	Cancelar Solicitação de Agendamento	■					Indefinido
RF025	Visualizar as Solicitações de Agendamentos dos Alunos		■				Importante
RF026	Atender Solicitações de Agendamento		■				Importante
RF027	Desabilitar Horários ou Datas não Disponíveis em Agendamentos		■				Desejável
RF028	Realizar Frequência Presencial dos Alunos		■				Essencial
RF029	Pré-confirmar Presenças na monitoria presencial	■					Desejável
RF030	Consultar Pré-confirmação presencial dos alunos		■				Desejável
RF031	Consultar Frequência dos Alunos		■	■			Essencial
RF032	Gerar Relatório de Frequências		■	■			Desejável
RF033	Cadastrar Horários e Local de Atendimento Presencial		■				Essencial
RF034	Visualizar Monitorias Matriculadas	■					Essencial
RF035	Visualizar Horários das Monitorias	■					Essencial
RF036	Visualizar Monitorias da Universidade	■					Essencial
RF037	Visualizar Atividades	■					Desejável
RF038	Realizar Atividades	■					Desejável

Fonte: Projeto PAE, adaptado pelo autor.

Continuação da Lista dos Requisitos Funcionais do projeto PAE

Lista dos Requisitos Funcionais							
Identificador	**Título do Requisito**	**Atores Participantes**					**Prioridade**
		Aluno	**Monitor**	**Professor**	**Administrador**	**Sistema**	
RF039	Consultar Nota da Atividade	■					Desejável
RF040	Cadastrar Atividades		■				Desejável
RF041	Alterar a Senha	■	■	■	■		Essencial
RF042	Notificar Usuários					■	Essencial
RF043	Autenticar Usuário	■	■	■	■		Essencial
RF044	Redefinir a Senha	■	■	■	■		Essencial
RF045	Visualizar Identificadores de Desempenho		■	■			Desejável
RF046	Moderar Usuários				■		Importante
RF047	Visualizar Logs				■		Essencial
RF048	Criar Tópicos de Dúvidas da Semana	■		■			Importante
RF049	Votar nos Tópicos de Dúvidas da Semana	■					Importante
RF050	Consultar Tópicos de Dúvidas	■	■				Importante
RF051	Enviar Mensagem		■	■			Importante
RF052	Bonificar os Alunos Mais Frequentes			■			Desejável
RF053	Notificar Mensagem por e-mail					■	Desejável

Fonte: Projeto PAE, adaptado pelo autor.