UNIVERSIDADE FEDERAL FLUMINENSE

CARLOS GRACIOLI NETO

UMA ABORDAGEM BASEADA EM DEFINIÇÃO DE ESCOPO E ELEMENTOS ESPERADOS PARA APOIAR INSPEÇÕES DE MODELOS DE *SOFTWARE*

NITERÓI
2019

CARLOS GRACIOLI NETO

UMA ABORDAGEM BASEADA EM DEFINIÇÃO DE ESCOPO E ELEMENTOS ESPERADOS PARA APOIAR INSPEÇÕES DE MODELOS DE *SOFTWARE*

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: ENGENHARIA DE SISTEMAS E INFORMAÇÃO.

Orientador: Prof. D.Sc. DANIEL CARDOSO MORAES DE OLIVEIRA

Niterói

2019

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

G731a Gracioli Neto, Carlos
UMA ABORDAGEM BASEADA EM DEFINIÇÃO DE ESCOPO E ELEMENTOS ESPERADOS PARA APOIAR INSPEÇÕES DE MODELOS DE SOFTWARE / Carlos Gracioli Neto ; Daniel Cardoso Moraes de Oliveira, orientador. Niterói, 2019.
110 f. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2019.

DOI: http://dx.doi.org/10.22409/PGC.2019.m.71902422104

1. Inspeção de Software. 2. Modelos UML. 3. Garantia de Qualidade de Modelo. 4. Definição de Escopo. 5. Produção intelectual. I. Oliveira, Daniel Cardoso Moraes de, orientador. II. Universidade Federal Fluminense. Instituto de Computação. III. Título.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

CARLOS GRACIOLI NETO

UMA ABORDAGEM BASEADA EM DEFINIÇÃO DE ESCOPO E ELEMENTOS ESPERADOS PARA APOIAR INSPEÇÕES DE MODELOS DE *SOFTWARE*

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: ENGENHARIA DE SISTEMAS E INFORMAÇÃO.

Aprovada em março de 2019.

BANCA EXAMINADORA

Prof. D.Sc. DANIEL CARDOSO MORAES DE OLIVEIRA – Orientador
Universidade Federal Fluminense

Prof. D.Sc. JOSÉ VITERBO FILHO
Universidade Federal Fluminense

Prof. D.Sc. MARCOS KALINOWSKI
Pontifícia Universidade Católica do Rio de Janeiro

Niterói
2019

A Deus, a minha amada esposa Elen, por seu apoio incondicional em todos os momentos, a minha filha Maria Eduarda pela alegria trazida em nosso lar e à minha família.

AGRADECIMENTOS

Agradeço a Deus por me guiar, iluminar e me dar tranquilidade para seguir em frente com os meus objetivos e não desanimar com as dificuldades.

Agradeço ao professor Marcos Kalinowski por me orientar durante essa jornada. Sou grato pelos ensinamentos preciosos, inclusive, por ter me ajudado em todo o experimento e mapeamento sistemático, assim como ter me apresentado a Engenharia de *Software* Experimental.

Agradeço ao professor Daniel Cardoso Moraes de Oliveira por me orientar após o professor Marcos sair da instituição. Sou grato pela paciência e pela excelência na condução da orientação dessa pesquisa e, também, por ter me ensinado Banco de Dados Distribuídos, quando fui seu aluno.

Graças a eles eu posso dizer que sou um pesquisador.

À UFF pela oportunidade de cursar o mestrado em Computação e pela sala cedida a mim para que eu pudesse trabalhar em um ambiente adequado.

À todos os professores do Instituto de Computação com quem pude conviver e, em especial, ao professor José Viterbo Filho, por ter acreditado no potencial de um artigo elaborado em sua disciplina Aprendizado de Máquina e ter publicado em um congresso nacional.

Ao Amadeu Anderlin Neto, doutorando em Informática na Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), pelo valioso trabalho e auxílio na revisão/confecção do artigo publicado no ICEIS.

À minha família (minha esposa Elen e minha filha Maria Eduarda) por ter me apoiado em todo o trajeto e entendido o esforço empreendido.

Se não for o Senhor o construtor da casa, será inútil trabalhar na construção. Se não é o Senhor que vigia a cidade, será inútil a sentinela montar guarda.
Será inútil levantar cedo e dormir tarde, trabalhando arduamente por alimento. O Senhor concede o sono àqueles a quem ama.
Os filhos são herança do Senhor, uma recompensa que ele dá.
Como flechas nas mãos do guerreiro são os filhos nascidos na juventude.
Como é feliz o homem cuja aljava está cheia deles! Não será humilhado quando enfrentar seus inimigos no tribunal.

Salmos 127

RESUMO

A inspeção de *software* representa uma maneira eficaz de identificar defeitos em artefatos de *software* em sua fase inicial, como modelos. Infelizmente, modelos de maior escala e documentos de referência associados não podem ser completamente inspecionados em uma sessão de inspeção, que comumente dura até duas horas. Sessões consideravelmente maiores mostraram uma eficiência de detecção de defeitos muito menor devido à fadiga cognitiva. O objetivo desta dissertação é propor e avaliar uma abordagem de definição de escopo para permitir a inspeção de partes específicas de interesse em modelos de maior escala. Primeiramente projetou-se a abordagem, que envolve a identificação dos Elementos de Modelo Esperados (EMEs) em partes selecionadas do documento de referência e, em seguida, o uso desses EMEs para o escopo do modelo. Foi realizado um experimento controlado usando artefatos industriais. Os participantes foram solicitados a realizar inspeções em diagramas de classe da UML com base em partes selecionadas de especificações funcionais. No tratamento experimental, a definição de escopo foi aplicada e foi fornecido aos inspetores o escopo do modelo e os EMEs. O grupo de controle usou o modelo original diretamente, sem EMEs. Foram medidas a eficácia e a eficiência da detecção de defeitos dos inspetores e coletados os dados qualitativos sobre a complexidade percebida. Os resultados demonstraram que a aplicação da abordagem de definição de escopo antes da inspeção aumentou a eficácia e a eficiência da detecção de defeitos do inspetor, com grandes tamanhos de efeito. Os dados qualitativos foram obtidos observando uma percepção de reduzida complexidade durante a inspeção.

Palavras-chave: Inspeção de *software*, modelos UML, garantia de qualidade do modelo, definição de escopo, estudo empírico.

ABSTRACT

Software inspection represents an effective way to identify defects in early phase *software* artifacts, such as models. Unfortunately, larger scale models and associated reference documents cannot be thoroughly inspected in one inspection session, which usually lasts up to two hours. Considerably longer sessions have shown a much lower defect detection efficiency due to cognitive fatigue. The goal of this dissertation is to propose and evaluate a model scope definition approach to allow inspecting specific parts of interest in larger scale models. First, we designed the approach, which involves identifying Expected Model Elements (EMEs) in selected parts of the reference document and then using these EMEs to scope the model. We conducted a controlled experiment using industrial artifacts. Subjects were asked to conduct UML class diagram inspections based on selected parts of functional specifications. In the experimental treatment, model scope definition was applied and the inspectors were given the scope model and the EMEs. The control group used the original model directly, without EMEs. We measured the inspectors defect detection effectiveness and efficiency and collected qualitative data on the perceived complexity. The results showed that applying model scope definition approach prior to the inspection significantly increased the inspector defect detection effectiveness and efficiency, with large effect sizes. Qualitative data allowed observing a perception of reduced complexity during the inspection.

Keywords: *software* inspection, UML models, model quality assurance, model scope definition, empirical study.

LISTA DE ILUSTRAÇÕES

Figura 1: Processo de inspeção de *software adaptado de* (FAGAN; E., 1976). 22
Figura 2: Uma família de técnicas para leitura de projeto orientado a objeto *adaptado de* (TRAVASSOS *et al.*, 1999a). 25
Figura 3: Estrutura da UML. 27
Figura 4: Exemplo de um Diagrama de Casos de Uso. 28
Figura 5: Exemplo de um Diagrama de Classes. 29
Figura 6: Processo do Mapeamento Sistemático. 33
Figura 7: *String* de busca do mapeamento sistemático. 36
Figura 8: O processo de filtragem dos artigos. 37
Figura 9: Número de artigos por biblioteca digital. 38
Figura 10: Estudos por técnicas de revisão. 41
Figura 11: Diagramas UML identificados nas abordagem. 42
Figura 12: Abordagem de definição de escopo de modelo com EMEs. 47
Figura 13: Diagrama de Classes do Módulo de Gestão de Usuários. 49
Figura 14: Lista de EMEs do Módulo de Gestão de Usuários. 49
Figura 15: Diagrama de Classes (Recortado) do Módulo de Gestão de Usuários. 50
Figura 16: Configuração do desenho cruzado no experimento. 57
Figura 17: Recortes durante a definição de escopo (retângulos tracejados) para o módulo de faturamento. 58
Figura 18: Indicador de eficácia (defeitos encontrados / defeitos semeados) para detecção de defeitos. 62
Figura 19: Indicador de eficiência (defeitos encontrados por hora) para detecção de defeitos. 63

LISTA DE TABELAS

Tabela 1: Tipos de Defeitos 20
Tabela 2: Esquema de classificação 39
Tabela 3: Técnicas/Ferramentas/Abordagens identificadas ordenadas por ano 40
Tabela 4: Objetivo do experimento controlado 52
Tabela 5: Experiência por participante no primeiro ensaio 55
Tabela 6: Experiência por participante no segundo ensaio 56
Tabela 7: Tipos de defeitos semeados no diagrama 59
Tabela 8: Nível de dificuldade dos defeitos semeados no diagrama 59
Tabela 9: Resultados quantitativos por sujeito e tratamento. 61
Tabela 10: Complexidade das tarefas experimentais percebidas pelos participantes. 64

LISTA DE ABREVIATURAS E SIGLAS

A	:	Ambiguidade (tipo de defeito)
CBR	:	Checklist-Based Reading (Leitura Baseada em Lista de Verificação)
CE	:	Critérios de Exclusão
CI	:	Critérios de Inclusão
DP	:	Desvio Padrão
EER	:	Extended Entity Relationship (Modelo Entidade Relacionamento Estendido)
EMEs	:	Elementos do Modelo Esperados
ES	:	Engenharia de *Software*
FI	:	Fato Incorreto (tipo de defeito)
GQM	:	Goal-Question-Metric (Objetivos-Questões-Métricas)
IE	:	Informação Estranha (tipo de defeito)
MS	:	Mapeamento Sistemático da Literatura
O	:	Omissão (tipo de defeito)
OORTs	:	Object-Oriented Reading Techniques (Técnicas de Leitura Orientada a Objetos)
PBR	:	Perspective-Based Reading (Leitura Baseada em Perspectiva)
QPs	:	Questões de Pesquisa
UML	:	Unified Modeling Language (Linguagem de Modelagem Unificada)
V&V	:	Verificação e Validação

SUMÁRIO

Capítulo 1 – INTRODUÇÃO 15
1.1 CONTEXTO E MOTIVAÇÃO 15
1.2 OBJETIVOS 16
1.3 METODOLOGIA DE PESQUISA 17
1.4 ORGANIZAÇÃO DA DISSERTAÇÃO 18
Capítulo 2 – REFERENCIAL TEÓRICO 19
2.1 INTRODUÇÃO 19
2.2 INSPEÇÃO DE *SOFTWARE* 19
2.2.1 TIPOS DE DEFEITOS 20
2.2.2 MÉTRICAS NA INSPEÇÃO 21
2.2.3 PROCESSO DE INSPEÇÃO 21
2.2.4 TÉCNICAS DE LEITURA 23
2.2.5 BENEFÍCIOS DA INSPEÇÃO DE SOFTWARE 26
2.3 UML 26
2.3.1 DIAGRAMA DE CASOS DE USO 28
2.3.2 DIAGRAMA DE CLASSES 28
2.4 ELEMENTOS DO MODELO ESPERADOS 30
2.5 CONSIDERAÇÕES FINAIS 31
Capítulo 3 – MAPEAMENTO SISTEMÁTICO DE MÉTODOS DE REVISÃO DE MODELOS UML 32
3.1 INTRODUÇÃO 32
3.2 METODOLOGIA 33
3.3 IMPLEMENTAÇÃO 34
3.3.1 QUESTÕES DE PESQUISA 34
3.3.2 ESTRATÉGIA DE PESQUISA 35
3.3.3 FONTES DE DADOS 36

3.3.4 SELEÇÃO DE ESTUDO 36
3.3.5 ETAPAS DO ESTUDO 37
3.3.6 ESQUEMA DE CLASSIFICAÇÃO 39
3.4 RESULTADOS 39
3.4.1 MÉTODOS DE REVISÃO (QUESTÃO PRIMÁRIA) 39
3.4.2 DETALHES DOS NÍVEIS DOS MODELOS 41
3.4.3 VANTAGENS OU DESVANTAGENS RELATADAS 42
3.5 DISCUSSÃO DOS RESULTADOS 44
3.6 AMEAÇAS À VALIDADE 45
3.7 CONSIDERAÇÕES FINAIS 45
Capítulo 4 – ABORDAGEM PROPOSTA PARA DEFINIÇÃO DE ESCOPO PARA REVISÃO DE MODELOS UML 46
4.1 INTRODUÇÃO 46
4.2 ABORDAGEM DE DEFINIÇÃO DE ESCOPO DE MODELOS E EMES 47
4.2.1 EXEMPLO DE APLICAÇÃO DA ABORDAGEM 48
4.3 CONSIDERAÇÕES FINAIS 50
Capítulo 5 – ESTUDO DE VIABILIDADE DA ABORDAGEM DE DEFINIÇÃO DE ESCOPO PARA REVISÃO DE MODELOS UML 51
5.1 INTRODUÇÃO 51
5.2 PLANEJAMENTO DO EXPERIMENTO 51
5.2.1 OBJETIVOS 52
5.2.2 CONTEXTO DO EXPERIMENTO 53
5.2.3 SELEÇÃO DE VARIÁVEIS 53
5.2.4 HIPÓTESES 54
5.2.5 PARTICIPANTES 54
5.2.6 DESENHO DO EXPERIMENTO 57
5.2.7 MATERIAIS UTILIZADOS 57
5.2.8 OPERAÇÃO 60

5.3 RESULTADOS 60
5.3.1 ANÁLISE QUANTITATIVA 61
5.3.2 ANÁLISE QUALITATIVA 64
5.4 DISCUSSÃO DOS RESULTADOS 65
5.5 AMEAÇAS À VALIDADE 66
5.5.1 VALIDADE INTERNA 66
5.5.2 VALIDADE EXTERNA 66
5.5.3 VALIDADE DE CONSTRUÇÃO 67
5.5.4 VALIDADE DE CONCLUSÃO 67
5.6 CONSIDERAÇÕES FINAIS 67
Capítulo 6 – CONCLUSÕES 68
6.1 CONTRIBUIÇÕES 68
6.2 LIMITAÇÕES 68
6.3 TRABALHOS FUTUROS 69
REFERÊNCIAS 70
APÊNDICE A – LISTA DE ESTUDOS SELECIONADOS PARA O MAPEAMENTO SISTEMÁTICO 75
APÊNDICE B – MATERIAIS UTILIZADOS NO EXPERIMENTO 79
B.1 Descrição da Tarefa (Abordagem) 79
B.2 Descrição da Tarefa (*Ad-Hoc*) 80
B.3 Formulário de Caracterização 81
B.4 Formulário de Notificação de Defeitos 82
B.5 Questionário de Acompanhamento (Abordagem) 83
B.6 Questionário de Acompanhamento (*ad-hoc*) 85
APÊNDICE C – DOCUMENTAÇÃO DE REQUISITOS E DIAGRAMAS UTILIZADOS NO EXPERIMENTO 86
C.1 Subconjunto da Documentação real de requisitos - Módulo Administração (MADM) 86
C.1.1 Escopo do Módulo 86

C.1.2 Requisitos Funcionais (Recorte Real) 86
C.1.3 Diagrama de Casos de Uso (Recorte Real) 87
C.1.4 Descrição De Alguns Casos De Uso (Recorte Real) 87
C.1.5 Defeitos Semeados no Diagrama de Classes 91
C.1.6 Diagramas 92
C.1.6.1 Diagrama de Classes Sem Definição de Escopo 92
C.1.6.2 Diagrama de Classes Com Definição de Escopo 93
C.1.7 Elementos do Modelo Esperados (EMEs) 93
C.2 Subconjunto da Documentação real de requisitos - Módulo de Faturamento (MFAT) 95
C.2.1 Escopo do Módulo 95
C.2.2 Requisitos Funcionais (Recorte Real) 95
C.2.3 Diagrama de Casos de Uso (Recorte Real) 95
C.2.4 Descrição De Alguns Casos De Uso (Recorte Real) 95
C.2.5 Defeitos Semeados no Diagrama de Classes 99
C.2.6 Diagramas 100
C.2.6.1 Diagrama de Classes Sem Definição de Escopo 100
C.2.6.2 Diagrama de Classes Com Definição de Escopo 101
C.2.7 Elementos do Modelo Esperados (EMEs) 101
APÊNDICE D – DOCUMENTAÇÃO DE REQUISITOS E DIAGRAMAS UTILIZADOS NO CAPÍTULO 4 103
D.1 Escopo do Módulo de Gestão de Usuários (MGU) 103
D.2 Requisitos Funcionais (Recorte Real) 103
D.2.1 Diagrama de Casos de Uso 103
D.2.2 Descrição de Alguns Casos de Uso 103
D.2.3 Diagrama de Classes com Defeitos 109
D.2.4 Elementos do Modelo Esperados 109

CAPÍTULO 1 – INTRODUÇÃO

1.1 CONTEXTO E MOTIVAÇÃO

Modelos de engenharia de *software* representam abstrações para diferentes aspectos de um sistema de *software* (por exemplo, estrutura, comportamento ou interação). A qualidade de tais modelos pode ser de importância fundamental para a conclusão de projetos com sucesso (LANGE; CHAUDRON, 2005). Assim, a verificação de modelos antes da criação de *software* é de particular relevância para a análise de sistemas de informação de alta qualidade.

Verificar a representação correta dos conceitos de domínio em modelos de *software* requer conhecimento humano do domínio. Métodos de inspeção de *software* (THELIN *et al.*, 2003; TRAVASSOS *et al.*, 1999) foram encontrados eficazes para detectar defeitos em modelos de *software* e requisitos em estudos empíricos (ELBERZHAGER *et al.*, 2012).

Por exemplo, é a partir dos requisitos que muitos outros modelos serão produzidos. Em Linguagem de Modelagem Unificada (UML) (OMG, 2017), esses modelos são os artefatos que permitem explicitar as propriedades estruturais, interativas e comportamentais do *software*.

Desta forma, é alta a probabilidade de introduzir defeitos durante essa transformação de requisitos para os modelos (por exemplo, diagramas de casos de uso e classe) (TRAVASSOS; SHULL; CARVER, 2002) e se torna imprescindível a revisão desses modelos logo após a sua elaboração com o intuito de garantir sempre a consistência entre os mesmos. Por isso, a inspeção de *software* é um tipo particular de revisão com um rigoroso e bem definido processo (FAGAN; E., 1976) e a sua importância tem sido bem documentada na literatura (AURUM; PETERSSON; WOHLIN, 2002).

O conceito de inspeção de *software* foi introduzido (FAGAN; E., 1976) em 1976, relacionando-a à prática da revisão dos artefatos para a garantia da qualidade. A inspeção do modelo de *software* geralmente requer a verificação se um modelo conceitual representa corretamente e completamente o conteúdo de documentos de referência adequados, como especificações de sistemas. Na prática, os modelos que representam abstrações de grandes sistemas de informações corporativas também tendem a ser grandes (por exemplo, diagramas de classe da UML para grandes sistemas de informações podem ter centenas de classes de domínio). Infelizmente, os estudos de inspeção de modelos se concentraram apenas na inspeção de modelos de pequeno a médio porte até o momento.

Portanto, uma questão importante é como abordar casos em que modelos de larga escala precisam ser inspecionados em relação a seus documentos de referência associados, ou seja,

envolvendo materiais de inspeção além do tamanho que um inspetor pode cobrir dentro das limitações de um processo tradicional de inspeção de passagem única (LAITENBERGER; DEBAUD, 2000), que normalmente dura até duas horas.

Usualmente a inspeção dos modelos de projeto pode ser apoiada por 3 diferentes tipos de técnicas: *ad-hoc*, *checklists* e técnicas de leitura. Técnicas de leitura representam o estado da arte em tecnologias de apoio à inspeção do *software*. Diferentes técnicas de leitura vêm sendo desenvolvidas para a inspeção de artefatos distintos, tais como requisitos (MAFRA, 2006), casos de uso (DOS SANTOS, 2010), diagramas orientados a objetos (TRAVASSOS *et al.*, 1999b), dentre outras.

Sendo assim, foi realizado um mapeamento sistemático da literatura (MS) sobre métodos de revisão para modelos UML, que revelou que existe uma carência nesta área, principalmente no que diz respeito à inspeção de modelos UML (por exemplo, grandes diagramas contendo várias classes) envolvendo sistemas de grande porte, tendo como base os requisitos. Foram identificados 33 trabalhos relacionados, porém nenhum deles se aplica diretamente ao problema de definição de escopo de grandes modelos UML, embora tratem de conceitos semelhantes. A partir dessas motivações, iniciou-se o estudo apresentado neste trabalho, cujos objetivos serão brevemente descritos a seguir.

1.2 OBJETIVOS

O objetivo principal dessa dissertação é investigar a eficácia e a eficiência de uma proposta de abordagem de revisão e definição de escopo de modelos UML com EMEs, com base em evidência (WINKLER *et al.*, 2017a), com orientação específica e prática para identificar defeitos, concentrando os revisores em algum aspecto do projeto, com o objetivo de garantir que será atingido um alto grau de cobertura dos defeitos de projeto, bem como a divisão dos diagramas de grande porte para um melhor entendimento. Visando esse objetivo, foi proposta uma questão de pesquisa: “Qual é o impacto da abordagem de definição de escopo com EMEs na eficácia e eficiência da inspeção?”.

Para responder a pergunta proposta, foi conduzido um experimento baseado em um subconjunto de uma documentação de requisitos real e em diagramas baseados nesses requisitos, os quais foram inspecionados em dois testes de sala de aula com alunos de graduação, representando replicações internas exatas. Dois pacotes de inspeção foram criados contendo material do treinamento aplicado (*ad-hoc* com ou sem abordagem de definição de escopo de modelo com UML), especificação de requisitos, diagrama de casos de uso, cenários dos casos de uso e diagrama de classes, documentos com as regras de negócio, descrição da

tarefa, formulário de notificação de defeitos, questionário de acompanhamento e abordagem proposta apenas para um dos grupos. Os pacotes se diferenciavam apenas na definição de escopo.

1.3 METODOLOGIA DE PESQUISA

O interesse em estudos experimentais tem crescido ao longo dos anos, evidenciado pelo aumento do número de publicações envolvendo estudos experimentais (ZELKOWITZ; WALLACE, 1998). Ao serem aplicados em condições realistas no desenvolvimento de *software*, os estudos experimentais podem propiciar uma avaliação adequada para ajudar a validar tecnologias maduras - como por exemplo, uma avaliação da eficácia de métodos propostos em vários ambientes - e identificar os problemas em tecnologias menos maduras (SHULL; CARVER; TRAVASSOS, 2001).

A estratégia investigada nesse trabalho para lidar com esse problema envolve o escopo do modelo para partes selecionadas dos documentos de referência. Vale ressaltar que hoje em dia o *software* é tipicamente desenvolvido seguindo processos iterativos ou ágeis (THEOCHARIS *et al.*, 2015), onde novas especificações (por exemplo, histórias de usuários ou casos de uso e suas descrições) são adicionadas. Portanto, ser capaz de inspecionar de maneira efetiva e eficiente grandes modelos contra partes selecionadas (ou incrementais) de documentos de referência é uma necessidade prática.

Foi introduzido o conceito de escopo de modelo como uma peça de modelo bem definida que atua como um filtro ou visualização que mostra apenas elementos de modelo relevantes. Essa proposta de abordagem de definição de escopo com os Elementos de Modelo Esperados consiste em identificar os Elementos do Modelo Esperados (EMEs) nas partes selecionadas do documento de referência e usar esses EMEs para: (a) escopo do modelo (remover partes não relacionadas) e (b) guiar os inspetores durante a detecção de defeitos. Na subseção 1.4, é apresentada a organização dessa dissertação, como um resumo do que será abordado em cada capítulo.

Foi conduzido um experimento controlado com estudantes usando artefatos industriais reais (diagramas de classes UML e partes selecionadas de especificações funcionais) com o objetivo de entender como a abordagem de definição de escopo influenciaria a eficácia e eficiência da inspeção do modelo. Os participantes foram solicitados a conduzir inspeções de diagramas de classes UML com base em partes selecionadas de especificações funcionais para dois módulos diferentes. No tratamento experimental, o Modelo de Escopo com EMEs foi

aplicado e os inspetores receberam os EMEs e o modelo de escopo. O grupo de controle usou o modelo original diretamente, sem EMEs.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho está organizado da seguinte maneira: O capítulo 2, Referencial Teórico, apresenta o processo de inspeção de *software* e define termos relacionados a ele, além de apresentar a Linguagem de Modelagem Unificada (UML – *Unified Modeling Language)*, assim como os diagramas utilizados nesse trabalho e, por fim, apresenta os conceitos dos Elementos do Modelo Esperados (EMEs). O capítulo 3, Mapeamento Sistemático de Métodos de Revisão de Modelos UML, apresenta o mapeamento sistemático (MS) que sustenta esse trabalho, com o protocolo e resultados obtidos. O capítulo 4, Abordagem proposta para Definição de Escopo para Revisão de Modelos UML, apresenta a abordagem de definição de escopo de modelo com EMEs, detalhando uma aplicação prática da abordagem. No capítulo 5, Estudo de Viabilidade da Abordagem de Definição de Escopo para Revisão de Modelos UML, o estudo experimental é descrito, os resultados são analisados e são discutidas as ameaças à validade. Enfim, no capítulo 6, Conclusões, é apresentada a conclusão deste trabalho, explicitando as contribuições, limitações e trabalhos futuros.

Além dos capítulos citados anteriormente, esta dissertação também é composta por apêndices: O Apêndice A apresenta a lista completa de artigos selecionados do Mapeamento Sistemático da Literatura do Capítulo 3. O Apêndice B apresenta os materiais utilizados no estudo experimental em sua apresentação original, ou seja, como eles foram apresentados aos participantes do experimento em ambas as rodadas. Os materiais incluem o formulário de caracterização, as duas versões de descrição da tarefa, o formulário de notificação de defeitos e as duas versões do questionário de acompanhamento (*follow-up*). O Apêndice C apresenta o subconjunto da documentação real de requisitos, os defeitos semeados, os diagramas com defeitos que foram inspecionados durante as rodadas do estudo experimental e a abordagem proposta. O Apêndice D apresenta o subconjunto da documentação real de requisitos, os defeitos semeados e os diagramas que foram utilizados para exemplificar na prática a aplicação da abordagem proposta.

CAPÍTULO 2 – REFERENCIAL TEÓRICO

2.1 INTRODUÇÃO

Torna-se imprescindível o esforço na melhoria de atividades que garantam a qualidade do *software*, tendo em vista a sua construção envolver a elaboração e utilização de diversos artefatos em diferentes níveis de abstração e perspectivas. Esses vários níveis de abstração e perspectivas são importantes para apoiar os projetistas a representar e compreender os aspectos do *software* (LANGE; CHAUDRON, 2005).

Diante desse fato, esse capítulo está organizado da seguinte forma: A seção 2.2 apresenta algumas definições relacionadas com inspeção de *software* para uma melhor compreensão do assunto abordado no capítulo, enquanto que a seção 2.3 apresenta a Linguagem de Modelagem Unificada (UML – *Unified Modeling Language)*, assim como os diagramas utilizados nesse trabalho (Diagramas de Casos de Uso e de Classes); na seção 2.4 são apresentados os conceitos dos Elementos do Modelo Esperados (EMEs) e; enfim, na Seção 2.5 são apresentadas algumas considerações finais a respeito de inspeção de *software*.

2.2 INSPEÇÃO DE *SOFTWARE*

O *software* deve ser construído atendendo às especificações do projeto e o produto final deve servir às necessidades reais do usuário. A Verificação e Validação (V&V) têm, respectivamente, os interesses citados anteriormente (OBERKAMPF; TRUCANO, 2008).

A verificação tem o propósito de garantir que o componente de *software* ou sistemas baseados nele cumpra suas especificações, ou seja, verifica se está construindo certo o produto (BOEHM, 1981), enquanto que a validação busca garantir que o componente de *software* ou sistemas baseados nele cumpra suas especificações (WALLIN, 2002).

A inspeção de *software* é uma importante técnica de V&V, utilizada para verificar se um *software* está de acordo com seus requisitos (SAUER *et al.*, 2000), podendo ser aplicado a todos os artefatos de *software*, por exemplo, documento de requisitos, projeto de alto-nível, casos de testes e código fonte, e possui um processo de detecção de defeitos rigoroso e bem definido, tendo como ideal assegurar que o produto produzido, seja ele um documento, artefato ou outro resultado a ser fornecido, possui qualidade suficiente para ser utilizado por seu usuário (TRAVASSOS, 2001). Quando aplicada ao modelo conceitual, a inspeção é considerada como uma técnica de validação. Mas, se a técnica é aplicada sob o código fonte, ela toma o foco de verificação.

Inspeção de *software* é uma atividade recomendada para a melhoria da qualidade e traz diversos benefícios para um projeto, tais como: redução de esforço (CONRADI *et al.*, 1999), redução do tempo, redução de custo (GILB; GRAHAM; FINZI, 1993) e aumento da produtividade (CONRADI *et al.*, 1999), por isso, quanto antes a presença do defeito for revelada, menor o custo de correção do defeito e maior a probabilidade de corrigi-lo corretamente.

A próxima subseção 2.2.1 apresentará os tipos de defeitos, imprescindível para ajudar os inspetores a manter o foco nos tipos de problemas; em seguida, a Seção 2.2.2 descreverá alguns benefícios obtidos ao se realizar a inspeção de *software*; posteriormente, a Seção 2.2.3 abordará o processo de inspeção de *software*; e a Seção 2.2.4 apresentará algumas técnicas para detectar os defeitos; e, enfim, a Seção 2.2.5 descreverá alguns benefícios obtidos ao se realizar a inspeção de *software*.

2.2.1 TIPOS DE DEFEITOS

A inspeção tem como objetivo detectar os defeitos em um ou mais artefatos de *software*, portanto, conhecer os tipos de defeitos torna-se imprescindível para ajudar os inspetores a manter o foco nos tipos de problemas que eles devem procurar durante a inspeção.

Na literatura técnica existem diversas taxonomias para classificação de defeitos, diferindo entre si em relação à quantidade de categorias e seus detalhamentos (TRAVASSOS, 2001). A taxonomia utilizada neste trabalho segue, por conveniência, a classificação adotada em (TRAVASSOS, 2001), que apresenta um conjunto de classes genéricas de defeitos não mutuamente exclusivas, ou seja, um defeito pode pertencer a mais de uma categoria, e está apresentada na Tabela 1. Na prática, de acordo com (KALINOWSKI; CARD; TRAVASSOS, 2012) esses tipos mais genéricos podem ser complementados com tipos mais específicos que refletem características do artefato específico sendo revisado.

Tabela 1: Tipos de Defeitos

Tipos de Defeitos	Descrição
Omissão	Um ou mais diagramas, não contêm uma representação para um conceito apresentado nos documentos de requisitos ou nos requisitos gerais do projeto
Fato Incorreto	Um diagrama contém uma deturpação na representação de um conceito apresentado nos requisitos
Inconsistência	A representação de um conceito em um diagrama difere da representação do mesmo conceito em outros diagramas ou no mesmo diagrama.
Ambiguidade	A representação de um conceito em um diagrama não está representada de maneira clara, podendo ocasionar uma má interpretação no usuário que irá utilizar a documentação, como por exemplo os desenvolvedores.
Informação Estranha	Os diagramas contêm informações, que apesar de corretas, não se aplicam ao domínio e por isso não deveriam ser inseridas nos diagramas.

2.2.2 MÉTRICAS NA INSPEÇÃO

A métrica é um item fundamental para a garantia de qualidade, afinal só se é possível avaliar o que se pode medir. Por isso, é importante definir quais métricas serão utilizadas neste trabalho. Tais métricas são esforço, falsos positivos, eficácia, eficiência e precisão, que podem ser entendidas no contexto de inspeção de *software* de acordo com a definição de (WINKLER *et al.*, 2017b):

- *Esforço:* é a diferença entre os tempos de início e término relatados na inspeção, desconsiderando as pausas efetuadas.
- *Eficácia:* é a razão entre o número de defeitos corretos e o total de defeitos no documento.
- *Eficiência:* é o número de defeitos corretos em um determinado intervalo de tempo, por exemplo, defeitos encontrados por hora.
- *Falsos Positivos:* é o número de defeitos candidatos que não correspondem a defeitos semeados.
- *Precisão:* é a razão entre o número de defeitos corretos e o total de defeitos reportados (BRIAND *et al.*, 2014).

2.2.3 PROCESSO DE INSPEÇÃO

O processo de inspeção foi descrito primeiramente por Michael Fagan (FAGAN; E., 1976) e é largamente utilizado, servindo como base para vários outros processos de inspeções propostos. Um grupo de inspeção envolve vários participantes, por exemplo, desenvolvedores de *software*, em um processo formal de investigação. Esse processo que atualmente é conhecido como o processo tradicional de inspeção, possui papéis e atividades bem definidos (FAGAN; E., 1976):

- *Moderador*: responsável por gerenciar a equipe de inspeção e o processo de inspeção;
- *Autor*: produz o artefato que será inspecionado;
- *Redator*: relata os defeitos encontrados e as soluções sugeridas durante a inspeção;
- *Inspetor*: é o papel assumido por cada integrante da equipe que tenta encontrar defeitos no artefato, sendo que todos podem atuar como inspetores além de executar outras atribuições.

Entretanto, os pesquisadores concordam que o moderador não deve estar envolvido com a inspeção dos artefatos, limitando-se a organizar o processo de inspeção, isto é, planejando-o, distribuindo os papéis e os recursos, além de realizar outras funções gerenciais do processo (FUSARO; LANUBILE; VISAGGIO, 1997).

As atividades (Figura 1) que compõem o processo de inspeção de *software* são (FAGAN; E., 1976):

1. Planejamento – Um usuário, desempenhando o papel de moderador da inspeção, define o contexto da inspeção (descrição da inspeção, técnica a ser utilizada na detecção de defeitos, documentos a serem inspecionados, autor do documento, entre outros), seleciona os inspetores e distribui o material a ser inspecionado.
2. Preparação Individual – Os inspetores estudam os artefatos individualmente, e eventualmente, fazem anotações sobre estes, produzindo uma lista de discrepâncias. O fornecimento de técnicas de leitura pode facilitar a execução desta tarefa.
3. Reunião de Inspeção – Uma reunião em equipe ocorre, envolvendo o moderador, os inspetores e os autores do documento. Discrepâncias são discutidas e classificadas como defeito ou falso positivo. A decisão final sobre a classificação de uma discrepância sendo discutida é do moderador.
4. Retrabalho - O autor corrige os defeitos apontados pelo relatório que foi entregue pelo moderador.
5. Acompanhamento - O moderador verifica as correções feitas pelo autor nos artefatos e define se será necessário realizar uma nova inspeção.

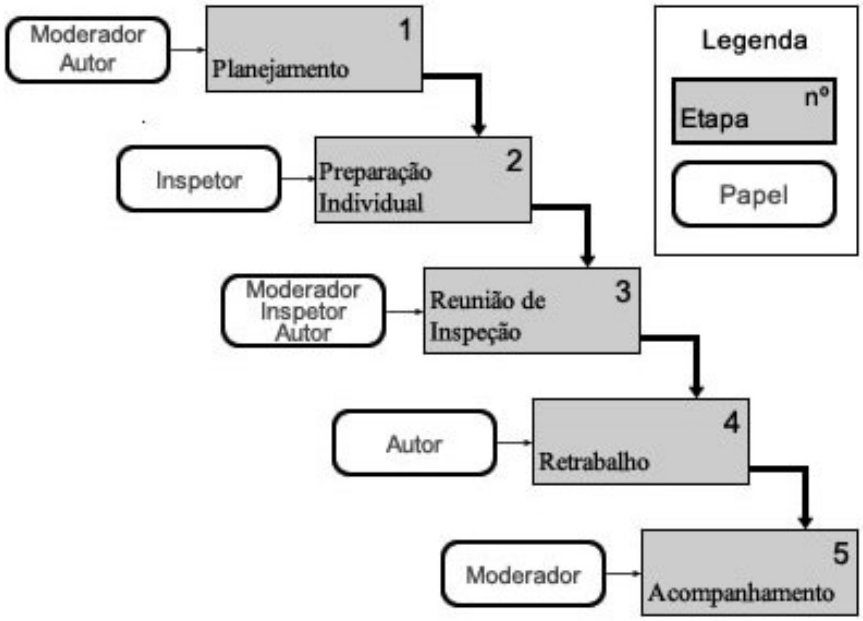


Figura 1: Processo de inspeção de *software adaptado de* (FAGAN; E., 1976)**.**

A literatura apresenta diversos outros processos de inspeção com diferentes etapas. Um exemplo de outro processo de inspeção é o proposto por (SAUER *et al.*, 2000), no qual as atividades de Preparação e a Reunião de Inspeção são substituídas por três atividades sequenciais: Detecção de Defeitos, Coleção de Defeitos e Discriminação de Defeitos.

Dentre as tarefas realizadas pelo moderador na fase de planejamento, está a escolha da técnica de revisão que será aplicada pelos inspetores na fase de preparação. Exemplos de técnicas que podem ser utilizadas são as técnicas de leitura, que são discutidas na próxima seção.

2.2.4 TÉCNICAS DE LEITURA

A inspeção é uma técnica de revisão baseada na leitura e entendimento de um documento a fim de encontrar defeitos. A solução, portanto, é empregar técnicas de leitura, que são um conjunto concreto de instruções dadas ao leitor de como ler e o que olhar em um produto de *software*. Há evidências empíricas de que as técnicas de leitura podem aumentar consideravelmente a eficácia da inspeção em diferentes tipos de artefatos de *software* (FUSARO; LANUBILE; VISAGGIO, 1997).

Leitura de *software* é uma análise individual de um produto textual de *software* que pode ser uma especificação de requisitos, código fonte, planos de teste, com objetivo de obter entendimento necessário para realizar uma tarefa como detectar defeitos por exemplo.

Existem diversas técnicas de leitura, sendo divididas em sistemática e não sistemática. A primeira aplica uma abordagem explícita e estrutural ao processo, fornecendo um conjunto de instruções aos inspetores e explicando como eles devem ler o documento e o que devem procurar. A segunda, aplica uma abordagem intuitiva e oferece o mínimo ou nenhum apoio aos inspetores, como é o caso do *checklist* e do *ad-hoc* (FUSARO; LANUBILE; VISAGGIO, 1997). As principais técnicas de leitura serão apresentadas nas subseções a seguir.

2.2.4.1 *AD-HOC*

Como discutido na seção anterior, a técnica *ad-hoc* é uma técnica não sistemática, popular e básica, pois não utiliza nenhuma técnica formal de leitura. Nessa técnica cada leitor lê o documento do seu modo. Por esse motivo, ela é dependente da experiência do leitor, e apresenta um grande defeito que é o fato de não ser repetível nem passível de melhoria, pois não existe um procedimento a ser seguido (FUSARO; LANUBILE; VISAGGIO, 1997).

2.2.4.2 BASEADA EM *CHECKLIST*

A técnica de *checklist*, também conhecida como *Checklist-Based Reading* (CBR) é baseada em uma série de questões específicas com o objetivo de guiar o inspetor em busca de fontes comuns de defeitos (FUSARO; LANUBILE; VISAGGIO, 1997). (LAITENBERGER; DEBAUD, 1997) apresentam alguns pontos negativos em relação ao *checklist*: as questões são genéricas e não suficientemente adaptadas para um ambiente de desenvolvimento particular; instruções concretas de como usar um *checklist* geralmente não são fornecidas e; as questões do *checklist* são limitadas a identificação de defeitos que pertencem a tipos de defeitos específicos.

2.2.4.3 BASEADA EM PERSPECTIVA

Foi desenvolvida por Victor Basili, em 1996, e já foi objeto de estudos (LAITENBERGER; EL EMAM; HARBICH, 2001), especialmente relacionados à inspeção de documento de requisitos (BASILI, 1996).

A técnica, também conhecida como *Perspective-Based Reading* (PBR) define cenários operacionais em que o leitor do documento a ser inspecionado deve seguir e pode ser caracterizada como uma fusão de questões e modelos (BASILI, 1996).

A PBR representa uma família de técnicas de leitura elaborada inicialmente para a detecção de defeitos em documentos de requisitos escritos em linguagem natural. Entretanto, sua utilização foi estendida para inspeções de projeto e código-fonte (SHULL, 1999).

2.2.4.4 BASEADA NO USUÁRIO

Conhecida como *Usage-Based Reading* (UBR) é uma técnica de leitura que auxilia a detecção de defeitos severos do ponto de vista do usuário (BERLING; THELIN, 2004).

A UBR foca na leitura orientada por um modelo de casos de uso priorizados em ordem de importância para o usuário do sistema, durante a fase de preparação de um processo de inspeção de *software*. A premissa básica é permitir que as expectativas do usuário governem a inspeção. Os modelos de caso de uso poderiam ser utilizados para inspeções em todas as fases de desenvolvimento (requisitos, projeto, código, *etc.*) de um projeto específico. Durante a inspeção, os revisores leriam o documento executando manualmente os casos de uso e tentando detectar defeitos que são mais importantes de acordo com a prioridade estabelecida e, por conseguinte, para o usuário. Conceitualmente UBR está relacionada com a perspectiva de usuário de PBR (BERLING; THELIN, 2004).

2.2.4.5 BASEADA EM DEFEITOS

A técnica conhecida como *Defect-Based Reading* (DBR) representa uma família de técnicas de leitura para a detecção de defeitos em documentos de requisitos descritos em SCR, uma notação formal baseada em máquinas de estado. Cada técnica DBR pode ser projetada para focar uma determinada classe de defeitos específicos à notação SCR (omissão de funcionalidade, inconsistência de tipos, etc.) (PORTER; VOTTA, 1994).

2.2.4.6 BASEADA EM CASOS DE USO

A técnica de caso de uso utiliza casos de uso da UML e tem como objetivo checar se cada objeto é capaz de responder corretamente a todos possíveis modos aos quais ele deve ser usado. Mais especificamente, a técnica verifica se os métodos corretos estão sendo chamados pelos usuários da classe e se as decisões e as mudanças de estados feitas em cada método são corretas e consistentes (DUNSMORE; ROPER; WOOD, 2003a).

2.2.4.7 ORIENTADA A OBJETOS

A também conhecida como *Object-Oriented Reading Techniques* (OORTs), representa uma família de sete técnicas de leitura que fornecem um procedimento para as revisões individuais dos diferentes diagramas e documentos de projeto orientados a objetos.

O processo de leitura utilizando OORTs precisa ser em duas dimensões: leitura horizontal e vertical. Na leitura horizontal, diferentes diagramas de projeto são verificados para assegurar que estejam consistentes entre si. Na leitura vertical, é necessária a validação dos modelos de projeto, para assegurar que o projeto do sistema está correto com relação aos requisitos. O conjunto completo de técnicas é ilustrado na Figura 2. Cada linha entre os artefatos representa uma técnica de leitura para ler um documento comparando-o com outro.

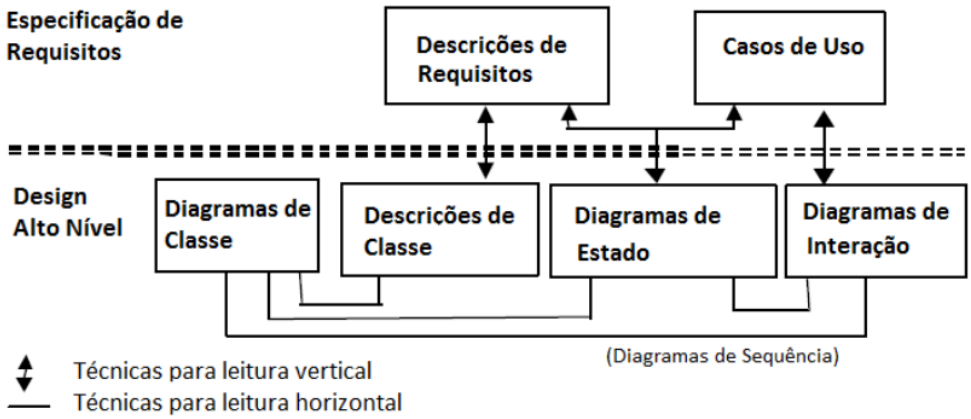


Figura 2: Uma família de técnicas para leitura de projeto orientado a objeto *adaptado de* (TRAVASSOS *et al.*, 1999a)**.**

2.2.5 BENEFÍCIOS DA INSPEÇÃO DE SOFTWARE

Um dos maiores benefícios de se utilizar inspeções de *software* é a detecção de defeitos nas fases iniciais do processo de desenvolvimento de *software*, facilitando a correção destes defeitos com menor esforço e custo. Desta forma, o esforço com retrabalho é reduzido em média para 10% a 20% do esforço total de desenvolvimento (BOEHM; BASILI, 2001). Essa redução no retrabalho pode implicar em melhorias significativas para a produtividade de *software*.

Estudos afirmam em suas pesquisas que além da redução do esforço com retrabalho, a produtividade, o tempo de projeto e o custo são reduzidos significantemente (LAITENBERGER; ATKINSON, 1999).

A aplicação de inspeções de forma bem planejada pode trazer diversos outros benefícios. Dentre eles se encontram (KALINOWSKI; SPINOLA; TRAVASSOS, 2004):

1. *Aprendizado:* Inspetores experientes podem tentar detectar padrões de como os defeitos ocorrem e definir diretrizes que ajudem na detecção destes. Além disto, bons padrões de desenvolvimento podem ser observados durante a inspeção, sendo possível sua descrição como melhores práticas para a organização.
2. *Integração entre processos de detecção e de prevenção de defeitos:* Saber onde e quando os defeitos ocorrem pode ajudar a estabelecer planos de contingência que evitem a sua ocorrência.
3. *Produtos mais inteligíveis:* Os autores dos diversos artefatos, sabendo que estes serão inspecionados, passarão a produzir artefatos de uma forma que sua compreensão seja facilitada. A produção de artefatos mais inteligíveis, além de facilitar a inspeção, trará benefícios para as fases seguintes do processo de desenvolvimento, incluindo principalmente a fase de manutenção.

Dados defeituosos ajudam a melhorar o processo de *software* do projeto: Analisando a causa dos defeitos encontrados é possível fazer ajustes no processo para evitar a ocorrência de defeitos deste mesmo tipo.

2.3 UML

A UML é uma linguagem-padrão para a elaboração de projetos de *software*. Ela poderá ser empregada para visualizações, especificações, construção e documentação de artefatos que façam uso de sistemas complexos de *software* (BOOCH; RUMBAUGH; JACOBSON, 2006).

Essa linguagem tem como objetivo criar uma apresentação gráfica de uma coleção de elementos de modelo, frequentemente mostrado como um gráfico conectado de arcos (relacionamentos) e vértices (os outros elementos) (FURLAN, 1998).

Em UML, existem duas categorias básicas de diagramas: diagramas de estrutura e diagramas de comportamento, essa ainda subdividida em diagramas de interação, conforme Figura 3 abaixo, representando a versão 2.5 (OMG, 2017).

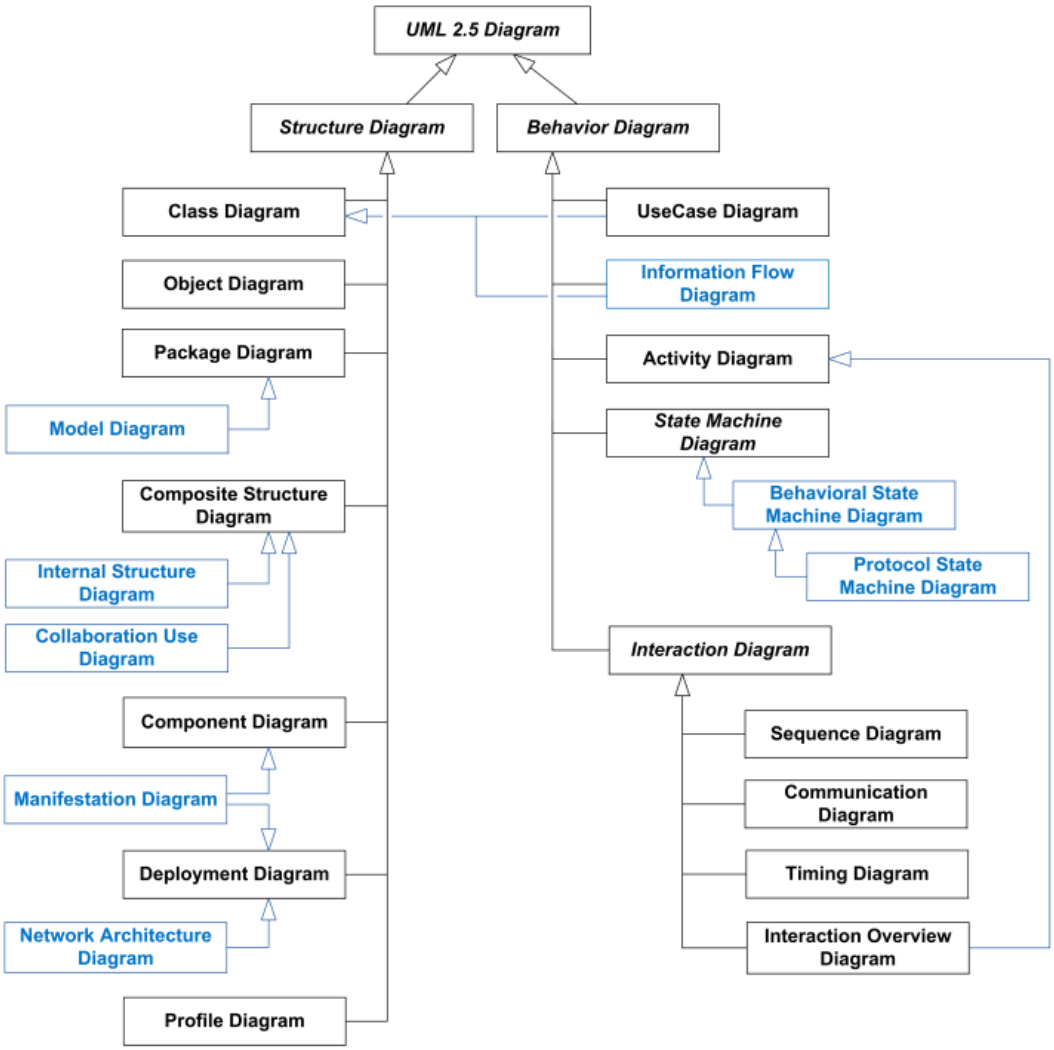


Figura 3: Estrutura da UML.

Cada diagrama UML pertence a uma dessas duas categorias de diagrama. O propósito dos diagramas de estrutura é mostrar a estrutura estática do sistema que está sendo modelado. Eles incluem a classe, o componente e/ou os diagramas de objeto. Os diagramas comportamentais, por outro lado, mostram o comportamento dinâmico entre os objetos no sistema, incluindo itens como métodos, colaborações, atividades e casos de uso (BOOCH; RUMBAUGH; JACOBSON, 2006).

Nas próximas Subseções são apresentados os diagramas utilizados nesse trabalho (Diagrama de Casos de Uso e Diagrama de Classes).

2.3.1 DIAGRAMA DE CASOS DE USO

Casos de uso são utilizados para entender melhor e com precisão os requisitos do *software*. No Diagrama de Casos de Uso e, consequentemente, em seus cenários, são descritas as principais funcionalidades do *software* e a interação dessas funcionalidades com os usuários do mesmo *software*.

Diagrama de Casos de Uso é um conjunto de ações que um *software* desempenha para produzir um resultado observável de valor a um ator específico. Serve para visualizar os relacionamentos entre os atores e os casos de uso do *software* (cenários), numa visão geral (FURLAN, 1998), conforme o exemplo da Figura 4 (BOOCH; RUMBAUGH; JACOBSON, 2006) abaixo.

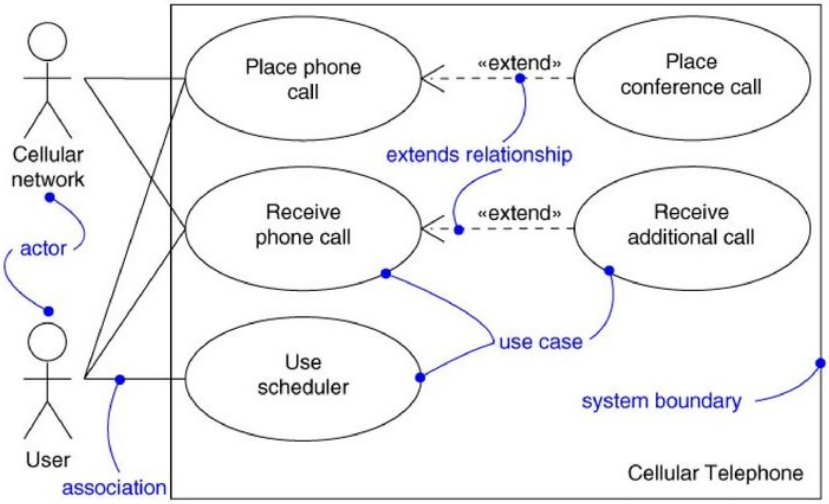


Figura 4: Exemplo de um Diagrama de Casos de Uso.

Esses diagramas são importantes, principalmente, para a organização e modelagem dos comportamentos de um sistema. É uma descrição de um conjunto de sequências de ações, inclusive variantes, que um sistema executa para produzir um resultado de valor observável por um ator (BOOCH; RUMBAUGH; JACOBSON, 2006).

2.3.2 DIAGRAMA DE CLASSES

O diagrama de classes é muito utilizado na modelagem de sistemas orientados a objetos, pois possibilita a visão de todo o relacionamento e colaboração entre as classes do sistema.

Conforme destacado, o diagrama de classe é um tipo de diagrama estruturado, desta forma, mostra a estrutura estática do sistema que está sendo modelado, concentrando-se nos elementos de um sistema, sem restrição de tempo. A estrutura estática é transmitida mostrando os tipos e suas instâncias no sistema. Além de mostrar os tipos de sistema e suas instâncias, os

diagramas de estrutura também mostram pelo menos alguns dos relacionamentos entre esses elementos e até mesmo podem mostrar suas estruturas internas. Em geral, esses diagramas permitem a validação do design e a comunicação do design entre os indivíduos e as equipes. (BOOCH; RUMBAUGH; JACOBSON, 2006). A Figura 5 apresenta um exemplo de diagrama de classes (BOOCH; RUMBAUGH; JACOBSON, 2006), que, conforme já definido, é utilizado para visualizar os aspectos estáticos desses blocos de construção e seus relacionamentos e para especificar detalhes da construção.

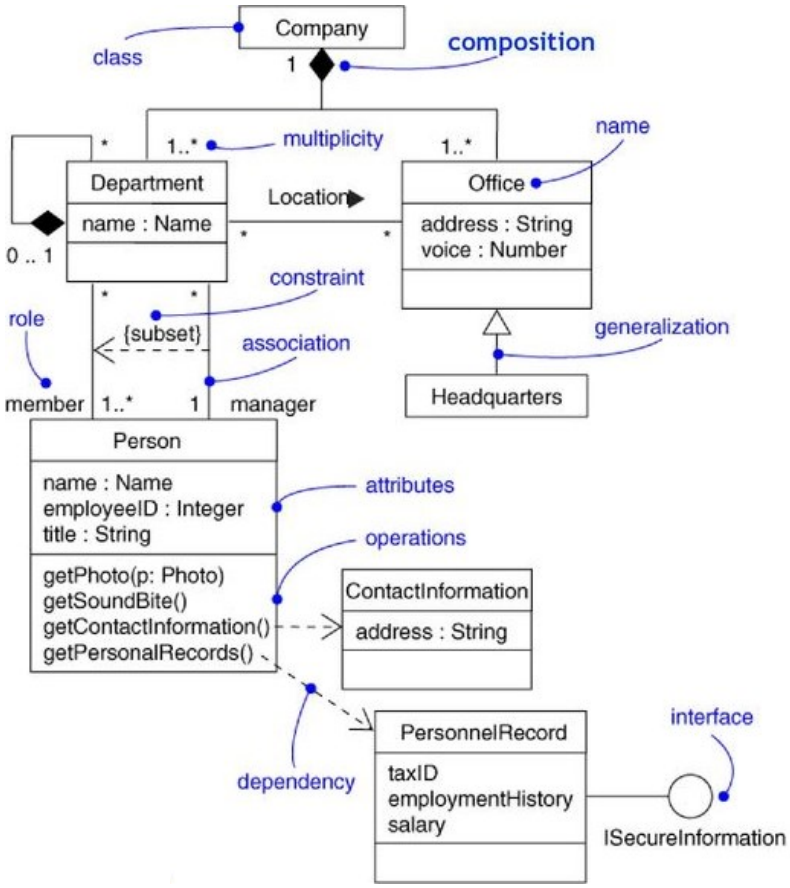


Figura 5: Exemplo de um Diagrama de Classes.

Diagramas de classes servem de base para outros tipos de diagramas existentes no projeto, importante para visualização dos modelos de estrutura do sistema. Trata-se de uma estrutura lógica estética em uma superfície de duas dimensões mostrando uma coleção de elementos declarativos de modelo, como classes, tipos e seus respectivos conteúdos e relações (FURLAN, 1998).

O propósito do diagrama de classes é mostrar os tipos que estão sendo modelados no sistema. Na maioria dos modelos UML, esses tipos incluem classe, atributos, métodos e associações.

2.4 ELEMENTOS DO MODELO ESPERADOS

Winkler *et al.* (2017a) investigaram distribuir o esforço de inspeção de modelos em um grupo de inspetores usando o *crowdsourcing* e, para isso, definiram que na fase de preparação da inspeção o gerente de inspeção deveria dividir a especificação do sistema, em um conjunto de entidades pequenas, ou seja, fragmentos de texto e sentenças. No trabalho desenvolvido, a análise foi realizada em um Modelo Entidade Relacionamento Estendido (EER - *Extended Entity Relationship*) e, baseando-se nesse modelo, esses conceitos (fragmentos) foram representados como entidades, atributos ou relacionamentos (WINKLER *et al.*, 2017a).

Dessa forma, fica claro que esses conceitos (elementos) são muito importantes, porém geralmente não são coletados explicitamente pelos inspetores e, por isso, definiram na pesquisa que os elementos encontrados deveriam se tornar explícitos. Eles nomearam esses conceitos como Elementos do Modelo Esperados (EMEs), pois representavam o resultado da análise de documentos de requisitos e eram insumos promissores para a inspeção de modelos (WINKLER *et al.*, 2017a).

Com base em Winkler (2017a, 2017b e 2018), essa dissertação fez o uso dos EMEs e aplicou em Modelos UML (apresentados nas seções anteriores), tendo em vista que nas pesquisas realizadas os resultados foram promissores.

Diferentemente de Winkler (2017a), os modelos utilizados nessa dissertação são da UML (Diagramas de Classes e Casos de Uso), por se basearem em *softwares* orientados a objetos e pela UML ser uma linguagem padrão para a elaboração da arquitetura de projetos de *software* (LARMAN, 2004).

Um elemento de modelo esperado (EME) da UML, na verdade, é uma abstração de um recurso estrutural ou comportamental do sistema que está sendo modelado e que inclui conteúdo semântico em um modelo (LARMAN, 2004). Todos os EMEs da UML possuem propriedades, por exemplo, um nome.

Levando em consideração os dois modelos apresentados (Diagrama de Casos de Uso e Diagramas de Classe), os EMEs dos Diagramas de Casos de Uso poderiam ser atores, associações, casos de uso e estereótipos, conforme demonstrada na Figura 4, enquanto que os EMEs dos Diagramas de Classes poderiam ser classes, atributos, operações/métodos e associações, conforme apresentou a Figura 5.

Para exemplificar a explicitação dos EMEs de um diagrama de classes, na seção 4.2.1 será apresentada uma aplicação prática da abordagem com a definição dos EMEs.

2.5 CONSIDERAÇÕES FINAIS

A inspeção do modelo de *software* geralmente requer a verificação se um modelo conceitual representa corretamente e completamente o conteúdo de documentos de referência adequados, como especificações de sistemas. Na prática, os modelos que representam abstrações de grandes sistemas de informações corporativas também tendem a ser grandes (por exemplo, diagramas de classe UML para grandes sistemas de informações podem ter centenas de classes de domínio). Infelizmente, os estudos de inspeção de modelos se concentraram apenas na inspeção de modelos de pequeno a médio porte até o momento.

Portanto, uma questão importante é como abordar casos em que grandes modelos precisam ser inspecionados em relação a seus documentos de referência associados, ou seja, envolvendo materiais de inspeção além do tamanho que um inspetor pode cobrir dentro das limitações de um processo tradicional de inspeção de passagem única (LAITENBERGER; DEBAUD, 2000), normalmente de até duas horas.

Na seção a seguir é apresentado um mapeamento sistemático da literatura com as técnicas de inspeção para modelos UML mais utilizadas.

CAPÍTULO 3 – MAPEAMENTO SISTEMÁTICO DE MÉTODOS DE REVISÃO DE MODELOS UML

3.1 INTRODUÇÃO

Os métodos ou técnicas de revisão provaram ser um método prático para garantir que os artefatos de *softwares* criados durante o ciclo de vida do *software* tenham as características de qualidade necessárias (TRAVASSOS *et al.*, 1999a). Existem atualmente vários métodos de revisão de *software*, mais precisamente de modelos UML, sendo utilizados e testados, entre eles: revisão por pares, inspeção de *software*, técnicas de leitura, PBR - Leitura Baseada em Perspectiva, CBR - Leitura Baseada em Lista de Verificação, OORTs - Técnicas de Leitura Orientada a Objetos, etc.

Neste contexto, essa dissertação apresenta um mapeamento sistemático da literatura, conduzido para abordar a seguinte questão de pesquisa de alto nível: **"Que métodos de revisão foram propostas para os modelos de UML?".** As seguintes questões de pesquisa complementares foram derivadas da questão principal:

Q1. Quais diagramas UML são o foco dos métodos identificadas?

Q2. Quais tipos de modelos (estruturais, comportamentais, interação) são o foco dos métodos identificadas?

Q3. Qual o nível de detalhamento dos modelos (conceitual, lógico ou de implementação) que estão sendo considerados nas abordagens identificados?

Q4. Quais tipos de controle de qualidade são o foco dos métodos identificados (validação e / ou verificação)?

Q5. Em quais contextos de modelagem (Atividades de Engenharia de Software e Domínios de Aplicação) as abordagens identificados foram aplicadas?

Q6. Quais são os tipos de pesquisa (por exemplo, artigo de avaliação, proposta de solução) em que as técnicas são apresentadas?

Q7. Quais são os tipos de estudos empíricos (por exemplo, experimento, estudo de caso, pesquisa) usados para avaliar os métodos?

Q8. Quais são as vantagens / benefícios relatados dos métodos identificados?

Q9. Quais são as desvantagens / deficiências relatadas dos métodos identificados?

As pesquisas foram realizadas por trabalhos publicados até o ano de 2016 e ao responder a essas perguntas, neste estudo, identificaram-se os métodos de revisão mais utilizados, expondo suas vantagens e desvantagens, assim como o estado atual das técnicas de revisão dos

modelos UML, identificando os tópicos nos quais novos esforços de pesquisa podem ser investidos, como a abordagem de definição de escopo que será apresentada na Seção 4.

Na Seção 3.2, é apresentada a metodologia utilizada para realizar o mapeamento. A Seção 3.3 apresenta a implementação da metodologia de pesquisa, incluindo o processo de definição das questões de pesquisa abordadas, o processo de seleção do estudo e o esquema de classificação utilizados. Os resultados do mapeamento sistemático são mostrados na Seção 3.4. A Seção 3.5 discute os resultados, enquanto que as ameaças à validade do estudo são apresentadas na Seção 3.6. Por fim, na seção 3.7, são apresentadas algumas conclusões.

3.2 METODOLOGIA

Mapeamento sistemático (MS) é um tipo de estudo secundário que pretende caracterizar uma determinada área de pesquisa através de um procedimento sistemático cujo objetivo é o de identificar a extensão e a natureza dos estudos preliminares disponíveis na área (BUDGEN *et al.*, 2008). Portanto, nesta pesquisa o MS será utilizado, pois pretende mapear a investigação em vez de responder a questão de pesquisa em detalhes, fornecendo uma estrutura do tipo de relatórios de pesquisa e resultados que foram publicados por categorizá-los e muitas vezes dá um visual resumo, mapa, de seus resultados.

O objetivo principal é pesquisar a literatura para determinar quais métodos de revisão têm sido propostas para modelos UML e que tipos de modelos (estrutural, comportamental, interação) são o foco das técnicas identificadas, onde são publicados, em quais bancos de dados foram indexados, quais os tipos de pesquisa e quais são as vantagens ou desvantagens relatadas das abordagens identificadas.

Um conjunto bem organizado de orientações e procedimentos para a realização de MS no contexto da engenharia de *software* é definido (PETERSEN *et al.*, 2008), o que constitui a base para o estudo apresentado nesta seção. A comunidade de pesquisa empírica engenharia de *software* definiu um processo padrão para a realização deste tipo de estudo. A Figura 6 mostra as fases de MS utilizados. A execução de cada fase será explicado em detalhes nas seções.

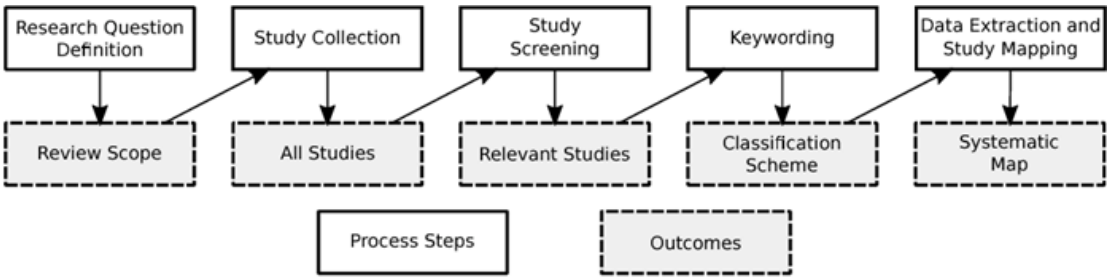


Figura 6: Processo do Mapeamento Sistemático.

3.3 IMPLEMENTAÇÃO

Nesta seção, será explicado como foi implementado o processo para o estudo de mapeamento sistemático (Figura 6), incluindo as perguntas de pesquisa, a estratégia e os termos de busca específicos, os critérios e processos de seleção e o esquema de classificação.

3.3.1 QUESTÕES DE PESQUISA

Para este estudo, uma questão de pesquisa primária foi definida:

QP: Que métodos de inspeção foram propostos para os modelos de UML?

As seguintes questões complementares de pesquisa foram derivadas da questão principal. Ao responder a essas perguntas, teremos uma caracterização detalhada dos estudos identificados:

Q1. Quais diagramas UML são o foco dos métodos identificados?

Descobrir os diagramas UML nos quais a pesquisa se concentrou, revelar as partes da UML que são consideradas mais importantes do que outras, bem como identificar oportunidades para futuras pesquisas.

Q2. Quais tipos de modelos (estruturais, comportamentais, interação) são o foco dos métodos identificados?

Os modelos UML são diagramas de três tipos principais: estrutural, comportamental e interação. Os modelos UML descrevem como as classes e objetos em um sistema interagem uns com os outros. Geralmente, os diagramas estruturais definem o sistema de *software* subjacente (o código), os diagramas comportamentais descrevem o que acontece no sistema sob certas condições e os diagramas de interação explicam o fluxo de controle. Esta questão poderia ser uma subquestão da Q2.

Q3. Qual o nível de detalhamento dos modelos (conceitual, lógico ou de implementação) que estão sendo considerados nas técnicas identificados?

O objetivo desta questão é identificar quais níveis de detalhes estão sendo considerados nas abordagens identificadas. Esta questão também poderia ser uma subquestão da Q2.

Q4. Quais tipos de controle de qualidade são o foco dos métodos identificados (validação e/ou verificação)?

Descobrir os diferentes tipos de qualidade de modelo e características específicas de qualidade que foram abordados pela pesquisa.

Q5. Em quais contextos de modelagem (atividade de Engenharia de Software - ES e domínios de aplicação) as técnicas identificadas foram aplicadas?

O objetivo desta questão é identificar os contextos de modelagem.

Q6. Quais são os tipos de pesquisa (por exemplo, artigo de avaliação, proposta de solução) em que as técnicas são apresentadas?

O objetivo desta questão é identificar que tipos de pesquisa, de acordo com a classificação proposta (WIERINGA *et al.*, 2006):

a) Pesquisa de avaliação: Implementada na prática, avaliação da implementação realizada; requer mais do que apenas um estudo de caso demonstrativo.

b) Proposta de solução: a solução para um problema é proposta, os benefícios/aplicações são demonstrados por exemplos, experimentos ou laboratórios de estudantes, também inclui propostas complementadas por um estudo de caso demonstrativo para o qual nenhum plano de avaliação/disseminação a longo prazo é óbvio.

c) Trabalhos filosóficos: Novo modo de pensar, estruturando um campo em forma de taxonomia ou quadro, estudos secundários.

d): Opinião pessoal, não fundamentada em trabalho relacionado e metodologia de pesquisa.

e) Artigo de experiência pessoal: Experiência pessoal, como as coisas são feitas na prática.

Q7. Quais são os tipos de estudos empíricos (por exemplo, experimento, estudo de caso, etc.) usados para avaliar os métodos identificados?

O objetivo desta questão é identificar quais tipos de estudos empíricos foram utilizados nas abordagens de revisão/inspeção propostas para os modelos de UML.

Q8. Quais são as vantagens/benefícios relatadas dos métodos identificados?

O objetivo desta questão é identificar, caso tenha sido relatado, as vantagens/benefícios dos métodos identificadas.

Q9. Quais são as desvantagens/deficiências relatadas dos métodos identificadas?

O objetivo desta questão é identificar, caso tenha sido relatado, as desvantagens/deficiências dos métodos identificadas.

3.3.2 ESTRATÉGIA DE PESQUISA

Primeiramente foram definidas as seguintes palavras-chave para realizar a pesquisa nas bibliotecas digitais:

População: a) UML; b) Orientado a Objetos;

Intervenção: a) Método, Técnica, Abordagem; b) Revisão, leitura, inspeção.

A Figura 7 apresenta a *string* de pesquisa derivada dessas palavras-chave.

"UML" OR "Unified Modeling Language" OR "Software Diagram" OR "Object Oriented"

AND

"Reading Technique" OR "Inspection Technique" OR "Review Technique" OR "Reading Approach" OR "Inspection Approach" OR "Review Approach" OR "Reading Method" OR "Inspection Method" OR "Review Method" OR "Software Inspection" OR "Software Review"

Figura 7: *String* de busca do mapeamento sistemático.

Foi aplicada essa *string* de busca a títulos e resumos. Optou-se por não fazer a pesquisa de texto completo porque descobriu-se que a pesquisa de texto completo resultou em um grande número de estudos de outros domínios, além do desenvolvimento de *software*.

Um parâmetro importante para avaliar a eficácia da busca é a definição de artigos de controle. Estes devem ser artigos reconhecidos na área pesquisada, e servem para avaliar se a busca está refletindo corretamente os objetivos propostos. Três artigos de controle (TRAVASSOS *et al.*, 1999a), (HUNGERFORD; HEVNER; COLLINS, 2004) e (SABALIAUSKAITE *et al.*, 2002) foram definidos para este mapeamento.

3.3.3 FONTES DE DADOS

Ao escolher as fontes de dados, pretendeu-se incluir importantes revistas e conferências sobre o tema da pesquisa. Serão incluídas publicações recuperadas de várias bibliotecas digitais e mecanismos de pesquisa da Web: Biblioteca Digital ACM, IEEE Xplorer e Scopus, pois são fontes de dados recomendadas para pesquisadores de engenharia de *software*, fornecendo acesso a importantes periódicos e conferências na área.

3.3.4 SELEÇÃO DE ESTUDO

Qualquer sequência de pesquisa requer ajustes, portanto, é necessário filtrar as funções irrelevantes, que exigem um conjunto de critérios.

Os critérios de inclusão (CI) foram:

CI-1: Artigos publicados que descrevem métodos de revisão para modelos UML;

CI-2: Quando vários estudos são relatados no mesmo artigo, cada estudo relevante é considerado separadamente.

Os critérios de exclusão (CE) foram:

CE-1: O artigo não está em inglês;

CE-2: Artigos que não possuem informações sobre os métodos de revisão para modelos UML;

CE-3: O artigo é apenas um resumo do tutorial ou workshop;

CE-4: Artigos que não foram avaliados pelos pares;

CE-5: O artigo ocorreu várias vezes (duplicados);

CE-6: O texto completo do artigo não está disponível para download.

A identificação e filtragem dos artigos foi dividida em cinco etapas, como mostra a Figura 8.

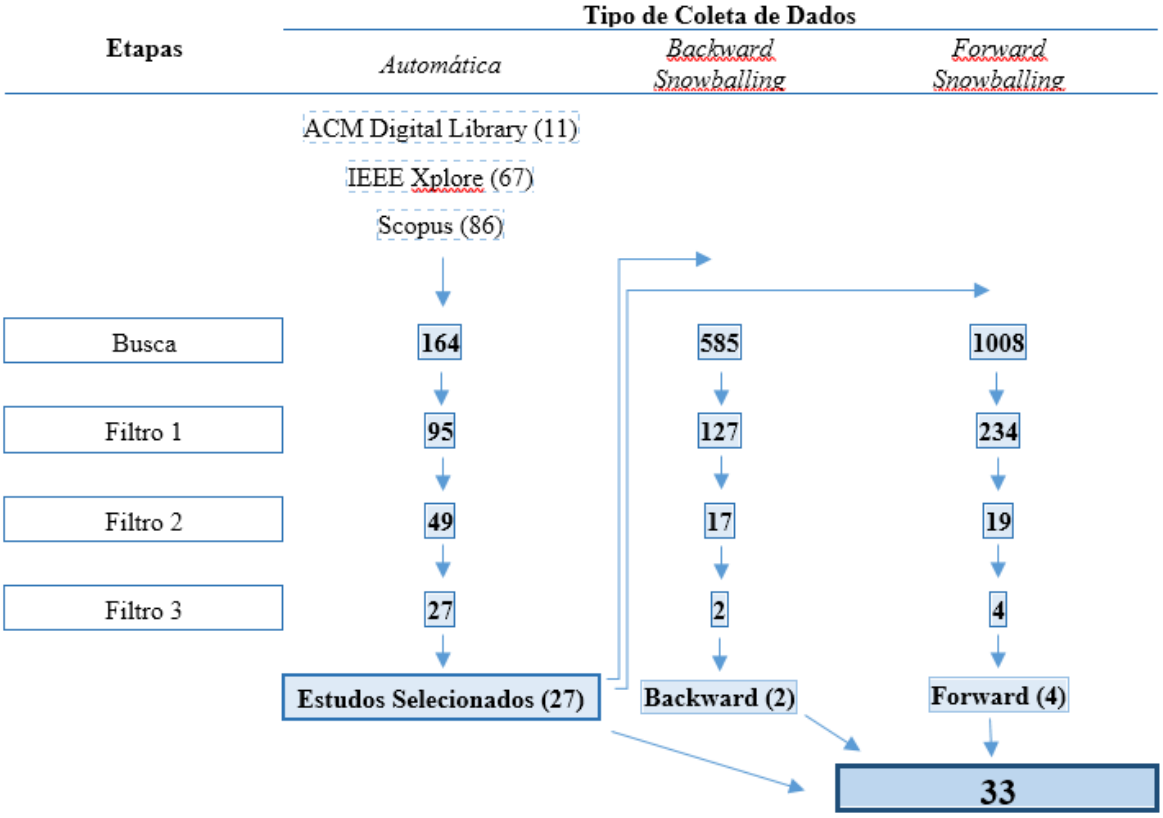


Figura 8: O processo de filtragem dos artigos.

3.3.5 ETAPAS DO ESTUDO

A primeira etapa consistiu na busca de artigos utilizando a *string* de busca em cada uma das bibliotecas digitais selecionadas para este estudo (Biblioteca Digital ACM, IEEEXplore e Scopus).

Na segunda etapa, a primeira filtragem (Figura 8 - Filtro 1) ocorreu e foi realizada por apenas um pesquisador. Neste processo, são utilizados os critérios de exclusão, retirando todos

os resumos dos trabalhos, apresentações em Power Point, que não tenham sido avaliados pelos pares, que não estejam em inglês e em duplicidades. Após essa etapa, 95 artigos foram selecionados para passar para o próximo estágio de filtragem.

Na terceira etapa, um segundo filtro (Figura 8 - Filtro 2) foi aplicado. Cada artigo foi analisado. A filtragem foi realizada através da leitura dos títulos, resumos e introduções (se necessário),utilizando os critérios de inclusão e exclusão. Quando uma decisão ainda não foi possível, o processo prosseguiu para a próxima etapa, onde foi lido na íntegra. Após essa etapa, 49 artigos foram selecionados para passar para o próximo estágio de filtragem.

Na quarta etapa, o último filtro (Figura 8 - Filtro 3) foi aplicado. Desta vez, os artigos selecionados foram lidos na íntegra. No final desta etapa, mais 22 artigos foram excluídos.

Por fim, na quinta etapa foi aplicada a técnica de bola de neve (*Forward*, através do Google Scholar e *Backward*), verificando as referências de cada estudo selecionado (27), a fim de não perder nenhum estudo potencialmente relevante (BOEHM; BASILI, 2001). No final, os três filtros foram aplicados nos estudos selecionados do processo de bola de neve, e finalmente foram combinados nos resultados finais da seleção do estudo. Os 33 artigos restantes foram usados para extrair as informações para responder às questões de pesquisa.

O número final de artigos selecionados de cada biblioteca digital é mostrado na Figura 9. Claramente, a maioria dos trabalhos selecionados veio das bibliotecas *on-line* Scopus e IEEEXplore.

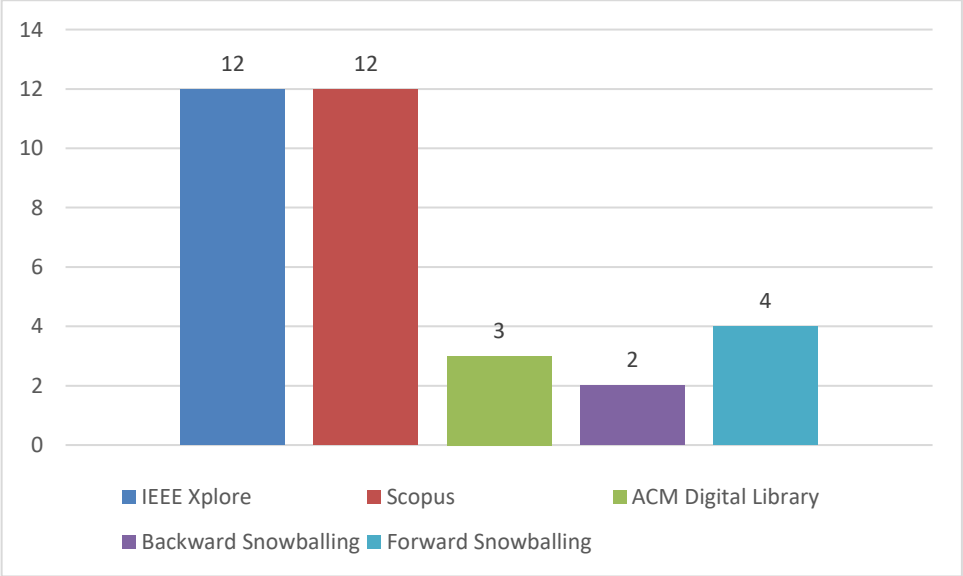


Figura 9: Número de artigos por biblioteca digital.

3.3.6 ESQUEMA DE CLASSIFICAÇÃO

Os atributos do esquema de classificação foram estruturados em seis categorias para permitir uma melhor análise da informação: meta dados sobre os estudos selecionados, métodos de revisão, nível de detalhamento dos modelos, tipos de controle de qualidade, contexto de modelagem e vantagens ou desvantagens do abordagens identificadas. Cada categoria (exibida na Tabela 2) está relacionada a uma ou mais questões de pesquisa (conforme definido na Subseção 3.3.1).

Tabela 2: Esquema de classificação

Categoria	Definição
Meta dados dos estudos (Q6 e Q7)	A primeira categoria contém dados sobre os estudos selecionados, tais como: locais onde os estudos foram publicados, autores e afiliações dos trabalhos, tipos de pesquisa (por exemplo, artigo de avaliação, proposta de solução), tipos de estudo empírico (por exemplo, experimento, estudo de caso) e ano de publicação.
Métodos de Revisão (Questão Primária)	Informações sobre métodos de revisão para modelos UML. Podemos separar as técnicas de inspeção em dois grupos principais: técnicas de leitura, que buscam orientar o inspetor na leitura do artefato para determinados aspectos e técnicas de verificação, que propõem um conjunto de procedimentos de verificação e validação.
Nível de detalhes dos modelos (Q1, Q2 e Q3)	Os modelos UML são divididos em três tipos principais: estrutural, comportamental e interação. Após analisar quais tipos de modelos foram utilizados e quais diagramas, é possível verificar o nível de detalhamento dos modelos que foram considerados.
Vantagens ou Desvantagens reportadas (Q8 e Q9).	Definidas as vantagens/benefícios e desvantagens/deficiências das abordagens identificadas.

3.4 RESULTADOS

Esta seção apresenta os resultados do estudo de mapeamento. Primeiro, é fornecida uma visão geral dos artigos incluídos e das técnicas identificadas. As subseções a seguir apresentam as respostas às nossas perguntas de pesquisa com base nas informações extraídas dos estudos incluídos.

3.4.1 MÉTODOS DE REVISÃO (QUESTÃO PRIMÁRIA)

No geral, foram identificados 33 trabalhos envolvendo 66 diferentes autores. As técnicas identificadas e uma breve descrição podem ser encontradas na Tabela 3. As referências completas da lista de artigos selecionados são apresentadas no Apêndice A. Em relação aos anos de publicação, os artigos que apresentam as técnicas de inspeção variam de 1999 a 2015.

A maioria das publicações (18 – 55%) são conferências e seminários e outros 15 (45%) trabalhos foram publicados em periódicos.

Tabela 3: Técnicas/Ferramentas/Abordagens identificadas ordenadas por ano

Id	Ano	Técnicas
P11	1999	PBR
P26		OORTs
P12	2000	PBR e CBR
P32	2001	PBR e CBR
P22	2002	PBR e CBR
P5		UBR e CBR
P28		CBR
P4	2003	PBR
P6		UBR, CBR e PBR
P7		CBR UBR e Abstraction-Driven Technique.
P25		UBR e CBR
P2		OORTs
P23		CBR e PBR
P9	2004	CBR
P20		CBR e PBR
P21		CBR e PBR
P29		Técnica com o uso de estereótipos
P24	2005	CBR e PBR
P1		PBR
P18	2006	Web-IPSE, AISA e sistema próprio
P3	2007	UBR, CBR e SBR
P10		OORTs
P15	2009	UCR, UBR e CBR
P13	2010	Ferramenta de detecção automática de defeitos
P16		LEAN
P14	2011	ActCheck
P31		Uso de estereótipos.
P8	2013	Técnica de Revisão de Código Baseado em Diagrama de Classes.
P17		CBR
P30	2014	Técnica de leitura baseada em PBR, CBR e DBR
P33		CBR; SBR; PBR; UBR; OORTs.
P19	2015	AGI
P27		SMartyCheck

A Figura 10 mostra o número de artigos que analisaram ou mencionaram cada técnica de revisão identificada neste mapeamento. Além dessas técnicas, existem ferramentas, representadas no gráfico pelo termo "Ferramentas". Finalmente, há também alguns trabalhos

que não definiram nomes para os métodos e, portanto, são representados no gráfico pelo termo "Outros".

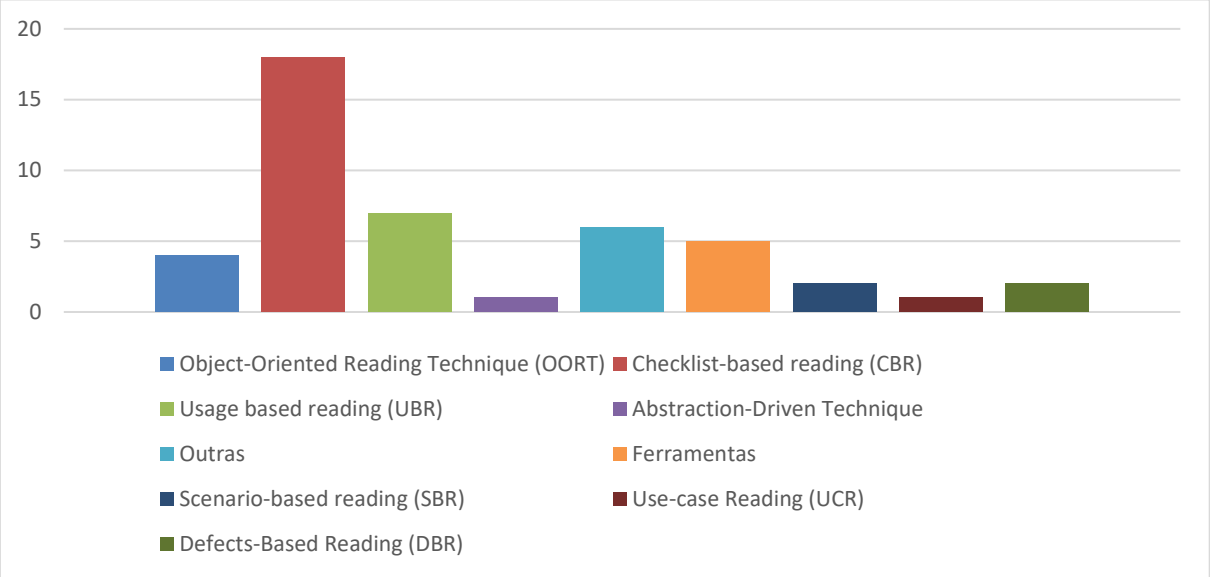


Figura 10: Estudos por técnicas de revisão.

Observa-se, ainda, no que se refere aos tipos de tipos de estudo empíricos, que o experimento e/ou quase-experimento foi o mais difundido (21 artigos - 64%), enquanto que o estudo de caso foi utilizado por um total de 12 artigos, representando 36% do total.

3.4.2 DETALHES DOS NÍVEIS DOS MODELOS

Os diagramas UML mais analisados nos documentos estão descritos na Figura 11. Pode-se observar que, destacam-se, entre os modelos estruturais, o Diagrama de Classes (23 - 69,70%) e, entre os modelos comportamentais, o Diagrama de Casos de Uso (20 - 60,61%). Por fim, dentre os modelos de interação, destacam-se o Diagrama de Sequência (15 - 45,45%). Observa-se que os artigos utilizaram como artefatos para revisar documentos especificando requisitos (18 - 54,55%) e Diagrama de Atividades (7 - 21,21%).

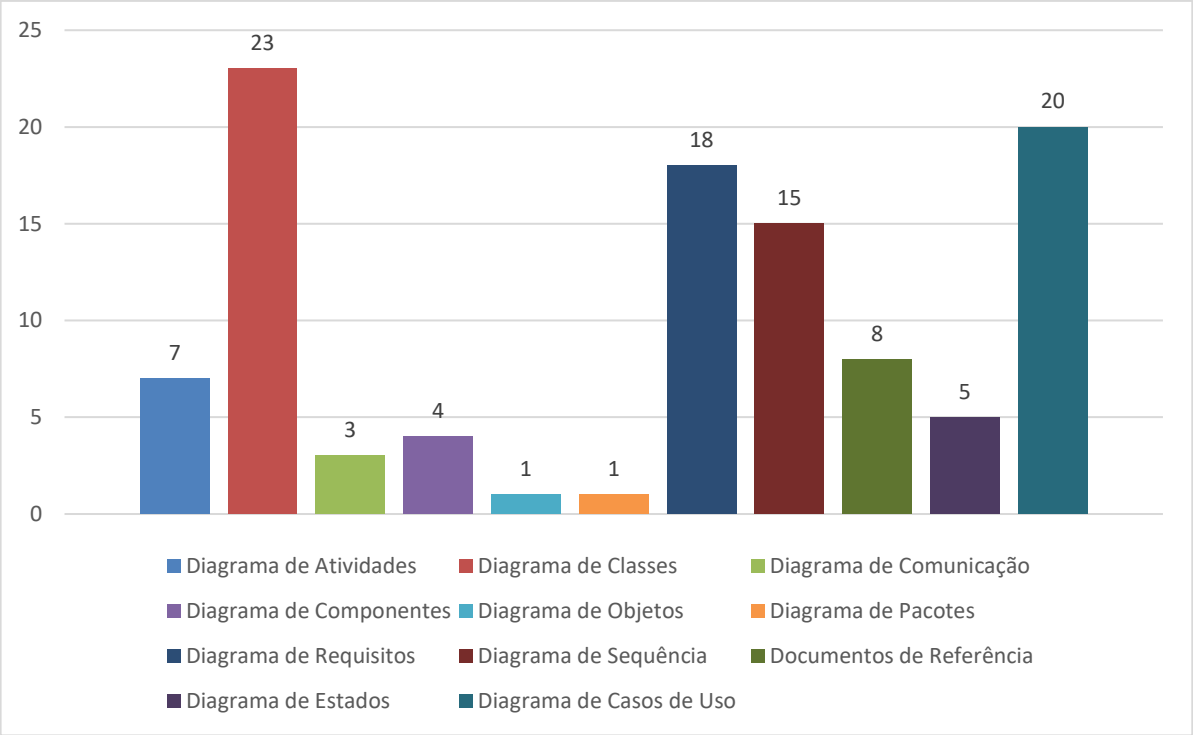


Figura 11: Diagramas UML identificados nas abordagem.

Em relação aos tipos de modelos (estrutural, comportamental, interação) especificamente, os artigos apresentaram o resultado: estrutural (25 - 76%), comportamental (32 - 97%) e interação (18 - 55%).

3.4.3 VANTAGENS OU DESVANTAGENS RELATADAS

Observa-se que a maioria dos trabalhos apresentou as vantagens dos métodos utilizados, mas apenas uma pequena parcela apresentou as desvantagens, no entanto, foi possível abordar as lacunas de alguns.

É importante destacar o estudo realizado (STARON; KUZNIARZ; THURN, 2005) que forneceu evidências, baseadas em vários experimentos, de que os estereótipos presentes nos modelos UML contribuem para garantir a qualidade do processo de inspeção de *software*, bem como a redução de defeitos pela aplicação de técnicas de CBR e PBR. Assim, os resultados obtidos evidenciaram que a técnica de CBR é mais eficiente e a PBR é mais efetiva. Portanto, os estereótipos são essenciais para garantir a qualidade das inspeções de *software* realizadas. Em outras palavras, o experimento fornece evidências de que a presença de estereótipos nos projetos de UML melhora a correção dos modelos após o processo de inspeção, uma vez que os modelos estereotipados continham menos falhas (mais falhas foram encontradas nos modelos durante as inspeções).

(SABALIAUSKAITE *et al.*, 2002) avaliaram o desempenho de dois grupos de inspeção ao longo de experimentos usando duas técnicas de leitura: CBR e PBR. Nesse cenário, os resultados obtidos pela comparação dessas técnicas não revelaram diferenças significativas entre eles.

Outro estudo realizado (SABALIAUSKAITE; KUSUMOTO; INOUE, 2004) também comparou as técnicas de CBR e PBR, mas em um contexto diferente do seu estudo anterior. Nesse, apresentaram uma avaliação do projeto de documentos UML inspecionados. Os seguintes resultados foram obtidos: eficácia semelhante na detecção de defeitos em ambas as técnicas de inspeção; os revisores que usaram o PBR passaram menos tempo inspecionando artefatos do que os revisores de CBR; e o custo por defeitos encontrados usando o CBR é menor que aquele usando PBR.

(DUNSMORE; ROPER; WOOD, 2003b) avaliaram as técnicas *Checklist*, *Abstraction-Driven* e *Use Case*, e os resultados mostraram que os participantes usando a técnica de leitura *Checklist* estavam encontrando a maioria dos defeitos, comparados àqueles usando as abordagens orientadas a abstração ou caso de uso. Eles citam as fraquezas (desvantagens) das técnicas estudadas: quanto à Lista de Verificação, citaram seu baixo desempenho ao lidar com defeitos relacionados a linhas de código ausentes e sua incapacidade de levar os inspetores a compreender o código sob inspeção; e quanto ao Caso de Uso, um elemento preocupante é que, novamente a partir da análise dos relatos, mais participantes utilizando essa técnica se desviaram da aplicação recomendada, possivelmente sugerindo uma falta de confiança na técnica.

(ROCHA; RAMALHO; MACHADO, 2015) propõem uma inspeção baseada em testes de automação, denominada Técnica de Inspeção Orientada Automatizada (AGI), adotando a Arquitetura Orientada a Modelos (Model-Driven Architecture - MDA) para modelos UML. Os autores desse estudo conduziram três estudos de caso para observar a eficácia do AGI. Os resultados obtidos revelam uma boa cobertura de diferentes tipos de defeitos e a taxa de detecção de defeitos por tempo é semelhante às outras técnicas: PBR; Inspeção Guiada (MGI) e OORTs. Estes fatos são devidos a AGI não dependerem de intervenção humana. As principais desvantagens são as seguintes: (1) o tempo de configuração é maior para o AGI; e (2) o fato de a AGI explorar apenas as regras de inspeção definidas na versão atual de suas ferramentas de apoio, enquanto a MGI pode identificar outros tipos de defeitos ainda não considerados, dependendo da experiência do inspetor.

3.5 DISCUSSÃO DOS RESULTADOS

Nesta seção, é apresentada uma discussão dos resultados. O objetivo deste mapeamento sistemático foi caracterizar as técnicas de inspeção propostas para os modelos UML, identificar os diagramas UML, os tipos de modelos e o nível de detalhamento dos modelos que são o foco das abordagens identificadas e quais são os relatos vantagens e desvantagens das abordagens identificadas.

(TRAVASSOS, 2014) discute os benefícios das inspeções de *software* para avaliar a viabilidade e a eficácia para suportar a identificação de defeitos existentes na maioria dos projetos de *software*. Ele apresenta os principais tipos de defeitos e técnicas de inspeção publicados na literatura.

(GERALDI *et al.*, 2015) propõe o SMartyCheck, uma técnica de inspeção de *software* baseada em lista de verificação para modelos de uso de SPL e modelos de variabilidade de classes de acordo com a abordagem do Smarty. O estudo empírico realizado em tal trabalho fornece evidências incipientes da viabilidade do SMartyCheck e melhora essa técnica por meio de feedback obtido de vários especialistas.

Os quase experimentos realizados por (LAITENBERGER; EL EMAM; HARBICH, 2001) também estabeleceram comparações para identificar a eficácia e o custo da detecção de defeitos na técnica PBR em relação à CBR. Para avaliar este cenário, foram realizadas duas réplicas levando em consideração os profissionais da Bosch Telecom GmbH. Então, os resultados gerais obtidos forneceram evidências de que o PBR é mais efetivo que o CBR. A técnica de PBR reduziu o custo por defeito encontrado e contribuiu para a detecção de uma quantidade maior de defeitos durante as reuniões de inspeção. Além disso, alcançou menores custos para identificação de defeitos com base no esforço dos participantes.

Outro estudo experimental importante conduzido por (THELIN; RUNESON; WOHLIN, 2003) apresenta a técnica de Leitura Baseada no Uso (UBR), que foi avaliada experimentalmente comparando-a com a técnica de CBR. Foi analisado por meio de três testes, que avaliaram a eficiência (defeitos encontrados por hora), eficácia (porcentagem de defeitos encontrados) e preparo (tempo de inspeção em minutos). Assim, o experimento indicou que a técnica UBR é significativamente mais eficiente e eficaz que a técnica CBR.

Em resumo, os estudos apresentados nesta seção responderam às questões de pesquisa e forneceram evidências das técnicas ou abordagens de inspeção existentes, abrangendo estudos, como vários estudos de validação acadêmica que comparam e avaliam a eficácia das

principais técnicas de inspeção (por exemplo, OORT, CBR , PBR, SBR, entre outros) em contextos distintos.

3.6 AMEAÇAS À VALIDADE

Os resultados deste mapeamento sistemático podem ter sido afetados por algumas ameaças à validade, tais como:

Questão de investigação: As questões de pesquisa definidas neste estudo podem não cobrir toda a área de métodos de revisão para modelos UML. Para lidar com este risco, as perguntas definidas foram analisados por pelo menos dois pesquisadores, um dos quais atuaram como consultor e revisor externo do protocolo, assim como foram propostos 3 (três) artigos de controle.

Viés de publicação: É difícil garantir que todos os trabalhos relevantes sejam retornados como resultados nas pesquisas realizadas. Para minimizar essa ameaça, foram considerados as principais bibliotecas digitais em computação, bem como as técnicas *snowballing*.

Cadeia de pesquisa: Há preocupação com a cadeia de pesquisa. Ela está relacionada com o uso dos termos “método de revisão” e "modelos UML" como parte da cadeia de pesquisa. Para isso, foram utilizados vários sinónimos, tal como abordagem, técnica, inspeção e orientação a objetos.

Extração de dados: É difícil assegurar que todos os estudos pertinentes primários foram selecionados para este mapeamento ou que os estudos foram analisados adequadamente. Para reduzir este risco, a classificação e a extração de informação foi realizada por um investigador e um revisor.

3.7 CONSIDERAÇÕES FINAIS

Através do mapeamento sistemático descrito nessa seção foi possível fornecer uma visão abrangente sobre o estado atual das pesquisas de abordagens de revisão. Foram identificados 33 estudos primários, cobrindo abordagens, técnicas e/ou ferramentas para inspecionar modelos UML de projetos de *software*. Essas abordagens variam em aplicabilidade, descrições, artefatos, vantagens, desvantagens, tipos de modelos, nível de detalhes de aplicabilidade, tipo de controle de qualidade e tipos de estudos empíricos.

Além disso, foi caracterizado o estado da arte dos métodos de revisão para os modelos UML, identificando possíveis lacunas e tópicos nos quais novos esforços de pesquisa podem ser investidos, assim como a abordagem proposta na próxima seção.

CAPÍTULO 4 – ABORDAGEM PROPOSTA PARA DEFINIÇÃO DE ESCOPO PARA REVISÃO DE MODELOS UML

4.1 INTRODUÇÃO

Ao longo dos anos, algumas pesquisas foram relatadas em relação a inspeções de modelos. Travassos *et al.* (1999), por exemplo, introduziram uma técnica de leitura para inspecionar a estrutura UML orientada a objeto e os modelos comportamentais em relação à consistência entre os modelos e as informações de referência. Sabaliauskaite *et al.* (2002; 2003; 2004) realizaram experimentos controlados com documentos da UML para comparar e avaliar a eficácia e eficiência das técnicas de leitura com diferentes níveis de orientação. Thelin *et al.* (2003) introduziram a leitura baseada em uso para orientar os inspetores priorizando primeiro os cenários de negócios e, em seguida, verificando se um modelo de projeto representava corretamente as informações dos cenários de negócios mais importantes.

Esses estudos se concentraram em um escopo completo de trabalho, geralmente envolvendo modelos de pequeno a médio porte e seus documentos de referência relacionados, que um inspetor médio pode abordar em duas horas para reduzir os riscos da fadiga. No entanto, os estudos não consideraram como abordar objetos de inspeção maiores (por exemplo, modelos maiores e/ou documentos de referência maiores). Portanto, não está claro até que ponto suas descobertas são válidas para (partes de) artefatos de *software* maiores. Winkler *et al.* (2017a; 2017b; 2018) investigaram distribuir o esforço de inspeção de modelos em um grupo de inspetores usando o *crowdsourcing*. Embora os inspetores tenham que se concentrar em partes específicas de um modelo, o escopo do modelo não foi considerado diretamente nessas investigações.

A ideia de usar a abordagem de definição de escopo para suportar inspeções de *software* também foi explorada por Briand *et al.* (2014). Seus resultados mostram uma diminuição significativa no esforço e um aumento na precisão das decisões quando os modelos são filtrados antes da inspeção.

No entanto, eles se concentraram em um problema específico, extraindo fatias de projeto (fragmentos de modelo) para suportar a inspeção de segurança. Em contraste, a abordagem proposta nesta pesquisa, detalhada nas próximas seções, é genérica e não restrita a um tipo específico de requisitos.

4.2 ABORDAGEM DE DEFINIÇÃO DE ESCOPO DE MODELOS E EMES

A ideia central da abordagem de Definição de Escopo de Modelo com EMEs é definir um escopo de modelo com base em uma parte selecionada dos documentos de referência (por exemplo, uma parte selecionada de uma grande especificação funcional). O escopo é conduzido com base nos elementos do modelo esperados (EMEs). Os EMEs para a parte selecionada dos documentos de referência devem ser identificadas e usados para: (a) escopo do modelo (remover partes não relacionadas aos EMEs) e (b) orientar os inspetores durante a detecção de defeitos.

Para ajudar a identificar os EMEs nos documentos de referência, as diretrizes comumente aplicadas ao projetar o modelo poderiam ser usadas. Por exemplo, Larman (2004) apresenta diretrizes para identificar classes, atributos, operações e relacionamentos para diagramas de classes UML com base em requisitos. Alternativamente, Sabou *et al.* (2018) argumentam sobre a viabilidade do uso de fontes especializadas para tal fim. Embora tais alternativas pudessem ser aplicadas, para fins de simplificação, no escopo desta pesquisa, considera-se que a abordagem de definição de escopo com EMEs é aplicada por uma função específica, que normalmente poderia ser conduzida por alguém com habilidades semelhantes àquelas necessárias para identificar esses elementos quando projetam modelos conceituais (por exemplo, um analista de requisitos)

A Figura 12 descreve o contexto no qual a abordagem de Definição de Escopo de Modelo com EMEs é aplicada. As entradas são a parte selecionada dos documentos de referência e o modelo a ser revisado, as saídas são a lista de EMEs e o modelo com escopo definido.

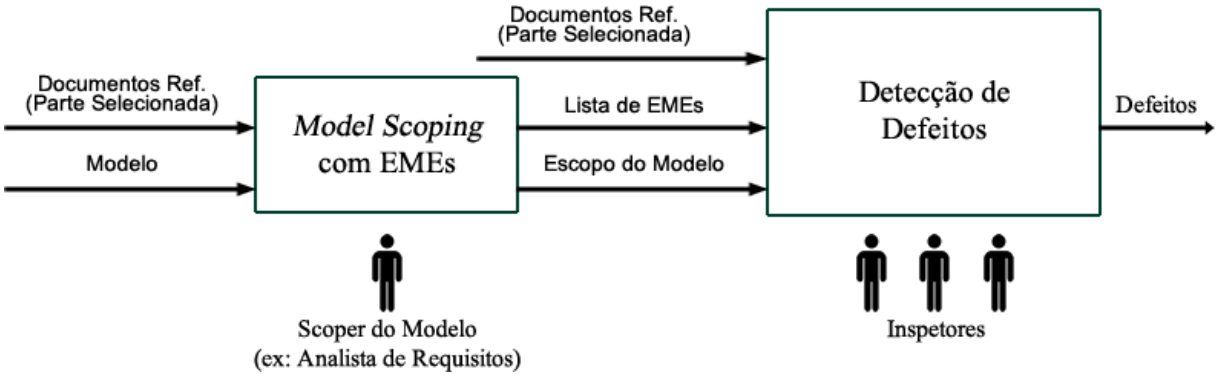


Figura 12: Abordagem de definição de escopo de modelo com EMEs.

A abordagem de definição de escopo de Modelo com EMEs é conduzido seguindo 3 etapas:

1. Com base no tipo de modelo a ser inspecionado, defina os tipos de EMEs a serem identificados (por exemplo, para diagramas de classe UML, tipos de EMEs são classes, atributos, métodos e relacionamentos; para diagramas de estado UML, tipos de EMEs são estados e eventos de transição).

2. Leia a parte selecionada dos documentos de referência e identifique a lista de EMEs (as diretrizes existentes para identificar EMEs podem ser usadas para apoiar esta etapa).

3. Escopo do modelo, removendo os elementos do modelo que não estão na lista de EMEs. Ao fazer isso, para evitar a remoção de informações contextuais relevantes (intimamente relacionadas), os seguintes elementos não devem ser removidos: (a) elementos que representam relacionamentos entre os elementos incluídos na lista de EMEs (por exemplo, uma associação entre classes em diagrama de classes da UML ou uma transição entre dois estados em um diagrama de estado da UML); e (b) elementos que estão contidos em um elemento incluído na lista de EMEs (por exemplo, atributos de uma classe em um diagrama de classes da UML).

Durante a detecção de defeitos (não faz parte da abordagem baseada em definição de escopo com EMEs), os inspetores podem usar a lista de EMEs e os documentos de referência para verificar se os EMEs estão corretamente representados no modelo com escopo como base para relatar desfechos.

Dessa forma, para exemplificar a aplicação da abordagem, na próxima subseção é apresentado um exemplo de aplicação da abordagem na prática, utilizando um modelo real de *software*.

4.2.1 EXEMPLO DE APLICAÇÃO DA ABORDAGEM

Após apresentar a metodologia da abordagem proposta, nessa seção será apresentada a aplicação prática da mesma, sendo utilizados artefatos de um projeto de *software* industrial real, basicamente um módulo do mesmo projeto utilizado no estudo de viabilidade, apresentado na próxima Seção 5.

O módulo utilizado para apresentar a abordagem foi o módulo de Gestão de Usuários, que objetiva controlar o acesso dos funcionários de uma empresa às funcionalidades do sistema. O Apêndice D apresenta o subconjunto da documentação real de requisitos, os defeitos semeados, os diagramas com defeitos que foram inspecionados durante essa apresentação da aplicação prática da abordagem proposta.

O Diagrama de Classes original, com base no Documento de Requisitos (Apêndice D) é apresentado na Figura 13.

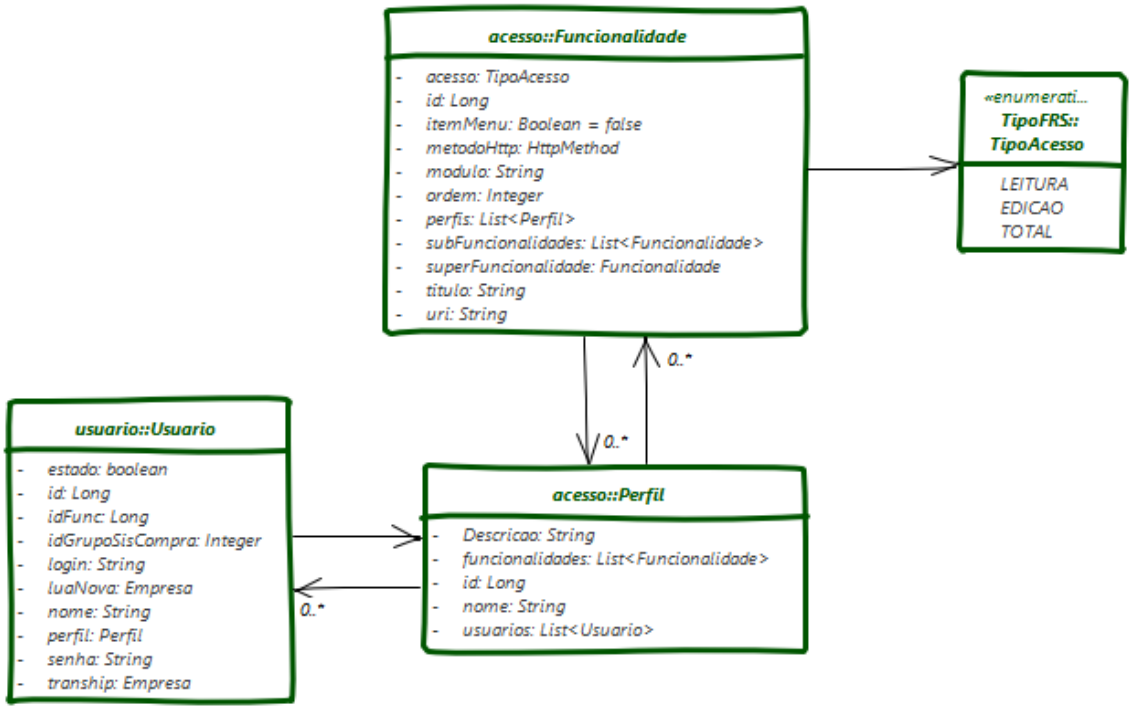


Figura 13: Diagrama de Classes do Módulo de Gestão de Usuários.

Primeiramente, para que seja possível aplicar a abordagem proposta, é necessário entender o tipo de modelo a ser inspecionado, e definir quais os tipos de EMEs a serem identificados, nesse caso, o diagrama de classe e, por isso, os EMEs são: classes, atributos, métodos e associações. Na Figura 14 é apresentada a etapa 2 da abordagem proposta, ou seja, a lista de EMEs com base no recorte do documento de requisitos.

CLASSES ESPERADAS

1. Funcionalidade
2. Perfil
3. Usuario

RELACIONAMENTOS ESPERADOS

1. Perfil ► Funcionalidade
2. Perfil ► Usuario

ATRIBUTOS ESPERADOS

1. Funcionalidade.id
2. Funcionalidade.itemMenu
3. Funcionalidade.metodoHttp
4. Funcionalidade.modulo
5. Funcionalidade.ordem
6. Funcionalidade.subFuncionalidades
7. Funcionalidade.superFuncionalidades
8. Funcionalidade.titulo
9. Funcionalidade.uri
10. Usuario.estado
11. Usuario.id
12. Usuario.idFunc
13. Usuario.idGrupoSisCompra
14. Usuario.luaNova
15. Usuario.nome
16. Usuario.perfil
17. Usuario.senha
18. Usuario.transShip
19. Perfil.decricao
20. Perfil.funcionalidades
21. Perfil.id
22. Perfil.nome

Figura 14: Lista de EMEs do Módulo de Gestão de Usuários.

Na etapa 3 (última) da abordagem, é realizada a definição do escopo (recorte do diagrama) com base nos EMEs definidos. O diagrama recortado é apresentado na Figura 15 a seguir.

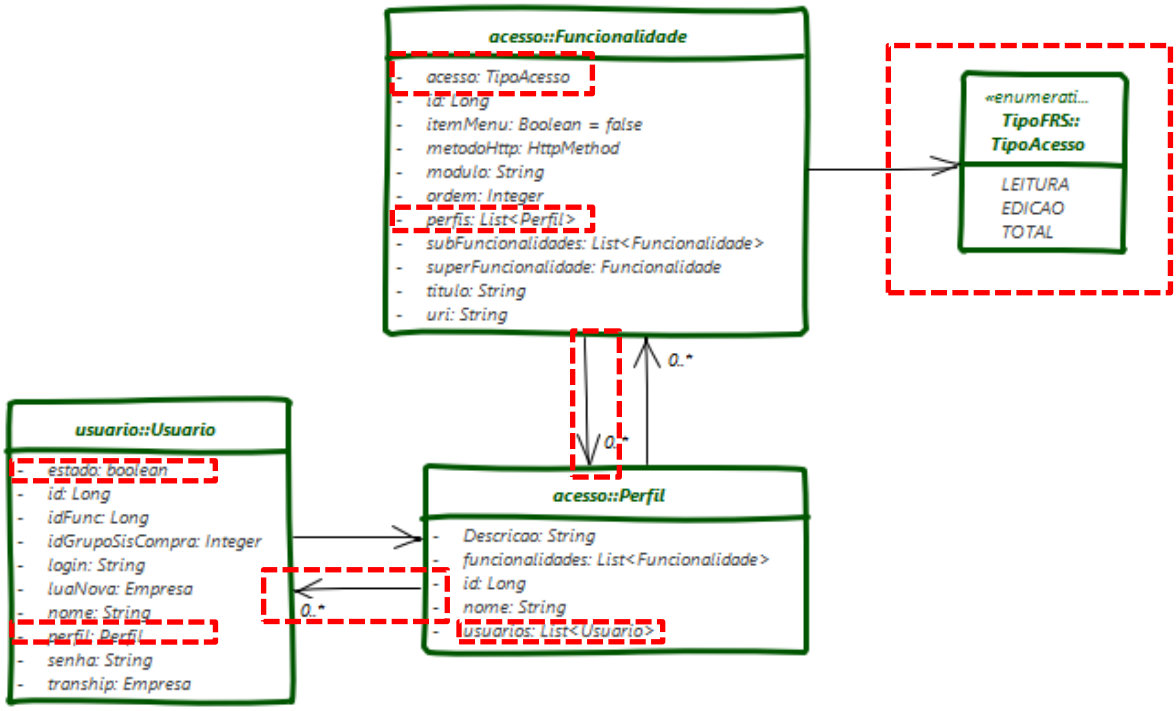


Figura 15: Diagrama de Classes (Recortado) do Módulo de Gestão de Usuários.

Preferiu-se apresentar o diagrama com os recortes a serem realizados (em vermelho) para representar o foco que os inspetores que utilizam a abordagem terão na hora de encontrar os defeitos no diagrama. Por fim, como foi apresentado na seção anterior, durante a detecção de defeitos, etapa que não faz parte da abordagem baseada em definição de escopo com EMEs, os inspetores podem usar a lista de EMEs e os documentos de referência para verificar se os EMEs estão corretamente representados no modelo com escopo como base para relatar desfechos.

4.3 CONSIDERAÇÕES FINAIS

Primeiro, foi projetada a abordagem, que envolve a identificação dos Elementos do Modelo Esperados (EMEs) em partes selecionadas do documento de referência e, em seguida, o uso desses EMEs para o escopo do modelo (ou seja, remover partes não relacionadas) e, por fim, a inspeção é realizada. Esses EMEs também podem ser usados para dar suporte a inspetores durante a detecção de defeitos. Na próxima seção será apresentado o experimento.

CAPÍTULO 5 – ESTUDO DE VIABILIDADE DA ABORDAGEM DE DEFINIÇÃO DE ESCOPO PARA REVISÃO DE MODELOS UML

5.1 INTRODUÇÃO

Atualmente, o *software* é tipicamente desenvolvido seguindo processos iterativos ou ágeis (THEOCHARIS *et al.*, 2015), onde novas especificações (por exemplo, histórias de usuários ou casos de uso e suas descrições) são adicionadas. Portanto, ser capaz de inspecionar de maneira efetiva e eficiente grandes modelos contra partes selecionadas (ou incrementais) de documentos de referência é uma necessidade prática.

Uma vez apresentada a abordagem proposta no capítulo anterior, fazia-se necessário avalia-la por meio de estudos experimentais, tendo em vista que tais estudos têm sido de grande importância, pois como afirmam (BASILI, 1996) os novos métodos, técnicas, linguagens e ferramentas não deveriam ser apresentados sem passar primeiro por um processo de experimentação a fim de serem validados, principalmente na área de engenharia de *software*, em que tem sido dada grande ênfase, pois vêm se mostrando um meio efetivo (LOTT; ROMBACH, 1996).

Um processo sistemático e incremental para validação de tecnologias maduras, e um guia para a melhoria de novas tecnologias, composto por 4 (quatro) etapas (Estudo de viabilidade, Estudo observacional, Estudo de caso em um ciclo de vida real, e Estudo de caso na indústria) foi proposto (SHULL; CARVER; TRAVASSOS, 2001) e seguindo esse processo, nossos primeiros estudos contemplam a etapa de Estudo de viabilidade.

Sendo assim, nesse capítulo será descrito o *planejamento do experimento* (Seção 5.2), ou seja, o contexto, as variáveis, as hipóteses, a seleção de participantes, o desenho e os materiais utilizados e a operação será apresentada de forma que o leitor possa compreender os passos executados no experimento. Os *resultados* são apresentados na Seção 5.3 e discutidos na Seção 5.4. Na Seção 5.5 são abordadas as *ameaças à validade*. E, por fim, na Seção 5.6 são realizadas as considerações finais.

5.2 PLANEJAMENTO DO EXPERIMENTO

Com o objetivo de entender como a Definição de Escopo com os EMEs influenciaria a eficácia e eficiência da inspeção do modelo, foi projetado e conduzido um estudo experimental. O planejamento, primeiro passo a ser executado em um experimento (WOHLIN *et al.*, 2000), foi feito com base na técnica *Goal-Question-Metric* (GQM) (VAN SOLINGEN *et al.*, 2002),

por isso essa seção se inicia abordando o objetivo e a questão de pesquisa na Subseção 5.2.1, o contexto do experimento na Subseção 5.2.2, a seleção de variáveis na Subseção 5.2.3, as hipóteses na Subseção 4.2.2, apresentará os participantes na Subseção 5.2.5, descreverá o desenho do experimento na Subseção 5.2.6 e os materiais utilizados na Subseção 5.2.7 e terminará descrevendo a operação na Subseção 5.2.8.

5.2.1 OBJETIVOS

É digno de nota que, embora a abordagem seja considerada genérica (ou seja, aplicável a qualquer tipo de modelo), foi instanciado o experimento usando diagramas de classe UML para serem inspecionados com relação às especificações do sistema de informações funcionais.

Com base em nossa meta, foi extraída a seguinte pergunta de nova pesquisa: **"Qual é o impacto do Escopo de Modelo com EMEs na eficácia e eficiência da inspeção?"** As variáveis usadas para responder a esta pergunta de pesquisa são descritas em detalhes na subseção 5.2.3.

Assim, sendo, o objetivo era investigar a eficácia e a eficiência da abordagem de definição de escopo para revisão de modelos UML, verificando sua viabilidade.

A Tabela 4 descreveu esse objetivo utilizando o modelo GQM (VAN SOLINGEN *et al.*, 2002). Medidas relacionadas com a quantidade de defeitos detectados e tempo dedicado à inspeção foram coletadas para apoiar a resposta às questões.

Tabela 4: Objetivo do experimento controlado

Analisar	a inspeção de diagramas de classes UML usando abordagem de definição de escopo com EMEs
Com o propósito de	Caracterizar
Em relação a	eficácia e eficiência de inspeção
Do ponto de vista de	Pesquisador de sistemas de informação
No contexto de	Inspeção de diagrama de classes UML baseado em uma especificação funcional válida, conduzida por inspetores iniciantes, quando comparada com a não utilização de abordagem de definição de escopo com EMEs.

Conforme definido na Tabela 4, esses estudos procuram investigar a eficácia e eficiência da abordagem proposta, e conforme destacado na Seção 2.2, eficácia é a razão entre o número de defeitos encontrados e o total de defeitos no documento enquanto que eficiência é o número de defeitos encontrados em um determinado intervalo de tempo, nesse caso, defeitos encontrados por hora (BIFFL; GUTJAHR, 2001).

5.2.2 CONTEXTO DO EXPERIMENTO

O experimento foi conduzido em dois testes de sala de aula de graduação, representando replicações internas exatas, envolvendo alunos matriculados em aulas de Engenharia de *Software* na Universidade Federal Fluminense (UFF). Esses alunos foram solicitados a revisar os diagramas de classes UML com base nas especificações funcionais corretas.

Para tornar o contexto mais representativo, foram selecionados artefatos de um projeto de *software* industrial real. O projeto dizia respeito ao desenvolvimento de um sistema integrado de gestão, com vários módulos. Foram selecionados os diagramas de especificação funcional e de classe de dois módulos, sendo um módulo referente às funções administrativas mais simples e o outro, aos serviços de faturamento financeiro mais complexos. Cada especificação funcional continha uma descrição geral, uma lista de requisitos funcionais, diagramas de casos de uso e descrições de casos de uso. Vale ressaltar que as especificações foram revisadas por profissionais e validadas por especialistas industriais.

Para o experimento, foram selecionados trechos de cada especificação funcional relacionados a casos de uso específicos, que eram bons representantes de casos de uso a serem implementados em um próximo ciclo de desenvolvimento e contra os quais o modelo deveria ser verificado antes da implementação. O fragmento do módulo administrativo compreendia as informações contextuais (ou seja, visão geral, requisitos funcionais, diagrama de caso de uso e descrições de casos de uso) para quatro casos de uso pequenos, enquanto o trecho do módulo de faturamento financeiro compreendia as informações contextuais para dois casos de uso mais complexos.

Essas especificações foram consideradas corretas. Foram semeados cada diagrama de classe com 28 defeitos artificiais relacionados aos respectivos trechos de especificação funcional. Para a distribuição dos tipos de defeitos, considerou-se a taxonomia de defeitos proposta por Shull (1998), contendo os tipos: ambiguidade, inconsistência, fato incorreto, omissão e informação irrelevante. Foram semeados 7 defeitos de cada tipo (exceto inconsistência, já que cada tarefa envolvia a inspeção de um modelo único que, portanto, não poderia ser inconsistente com os outros). A semeadura também considerou a inclusão de defeitos de diferentes níveis de dificuldade (fácil, médio e difícil) para cada tipo. Por fim, a atividade foi aplicada e, posteriormente, revisada.

5.2.3 SELEÇÃO DE VARIÁVEIS

Antes de iniciar o experimento, se faz necessário escolher as variáveis independentes e dependentes. As variáveis independentes são aquelas que é possível controlar e alterar na

experiência. As variáveis dependentes são os elementos que são necessários medir ao longo do experimento em um contexto particular. Mudando as variáveis independentes detecta-se se as variáveis dependentes são afetadas, ou não.

A variável independente no experimento de inspeção é o tratamento aplicado pelos grupos para encontrar defeitos no modelo UML. Embora ambos os grupos usassem uma técnica de inspeção *ad-hoc* (ou seja, nenhuma técnica de leitura específica), o grupo experimental recebeu a Taxonomia de Defeitos, a lista de EMEs e o escopo do modelo, e o grupo de controle recebeu apenas a Taxonomia de Defeitos e o modelo completo.

Em relação às variáveis dependentes, foram utilizadas a eficácia e a eficiência do experimento de inspeção, definido da seguinte forma:

- Eficácia é a razão entre o número de defeitos reais encontrados e o número total de defeitos conhecidos.
- Eficiência é a razão entre o número de defeitos reais encontrados e o tempo gasto.

Além de medir essas variáveis, foram coletados *feedbacks* qualitativos em um questionário de acompanhamento, inspirado no questionário Modelo de Aceitação de Tecnologia (TAM, *Technology Acceptance Model*) (DAVIS, 1989) sobre a utilidade percebida, a facilidade de uso e a intenção comportamental de adoção. O TAM fornece construções teóricas adequadas e tem sido extensivamente usado e validado para esse propósito (Turner *et al.*, 2010).

5.2.4 HIPÓTESES

Para investigar a questão de pesquisa e utilizando as variáveis descritas na subseção anterior, as seguintes hipóteses foram formuladas:

- **H01:** não há diferença na eficácia ao inspecionar diagramas de classes UML com ou sem o uso da abordagem de definição de escopo com EMEs.
- **H02:** não há diferença na eficiência ao inspecionar diagramas de classes UML com ou sem o uso da abordagem de definição de escopo com EMEs.

Na próxima subseção serão apresentados os participantes da pesquisa.

5.2.5 PARTICIPANTES

Os participantes deveriam representar inspetores novatos, para evitar conhecimentos específicos sobre técnicas de inspeção como um fator de confusão significativo, dessa forma foram selecionados alunos de duas turmas de graduação em Engenharia de *Software* na Universidade Federal Fluminense (UFF), envolvendo 44 (turnos diurnos) e 10 (noturnos).

Todos os alunos preencheram um formulário de caracterização com perguntas objetivas: (a) sua experiência com desenvolvimento de *software*; (b) sua experiência em modelagem orientada a objetos (UML); e (c) sua experiência em inspeção de *software*. Coletou-se o formulário de caracterização de dados de cada aluno e classificou em: nenhuma (N), baixa (B), média (M), média-alta (MA) e alta (A) experiência para cada tópico especializado.

A Tabela 5 apresenta os resultados da caracterização dos participantes do primeiro estudo e da divisão do grupo.

Tabela 5: Experiência por participante no primeiro ensaio

Grupo	ID Part.	Desenvolvimento de *Software*	Modelos UML	Inspeção de *Software*
1	P1	MA	M	M
	P2	B	MA	B
	P3	M	MA	M
	P4	MA	A	B
	P5	M	M	B
	P6	B	M	B
	P7	MA	B	B
	P8	A	A	M
	P9	MA	M	B
	P10	B	M	B
	P11	A	A	B
	P12	B	M	B
	P13	B	MA	M
	P14	B	B	B
	P15	M	M	M
	P16	MA	M	B
	P17	MA	MA	M
	P18	M	B	B
2	P19	A	MA	B
	P20	B	MA	B
	P21	B	B	B
	P22	MA	MA	B
	P23	MA	MA	B
	P24	MA	MA	M
	P25	B	B	M
	P26	B	B	B
	P27	B	B	B
	P28	B	B	B
	P29	MA	MA	B
	P30	M	M	MA
	P31	MA	M	B
	P32	MA	B	B

A Tabela 6 apresenta os resultados da caracterização dos participantes do segundo estudo e da divisão do grupo. Para facilitar a compreensão do bloqueio, destacaram-se os participantes mais experientes nas Tabelas (5 e 6) com preenchimento de fundo em tons de cinza.

Tabela 6: Experiência por participante no segundo ensaio

Grupo	ID Part.	Desenvolvimento de *Software*	Modelos UML	Inspeção de *Software*
1	P33	A	A	B
	P34	B	A	B
	P35	M	M	B
	P36	A	A	M
2	P37	A	A	M
	P38	A	A	B
	P39	B	A	B
	P40	M	A	B

Observa-se nas Tabelas 5 e 6 que, em relação à experiência com desenvolvimento de *software*, o assunto foi caracterizado como tendo: (a) Sem experiência, se ele nunca teve contato com o desenvolvimento de *software*; (b) Baixa experiência, se ele teve contato com desenvolvimento de *software* apenas nas aulas, leitura de material de apoio ou para uso próprio; (c) Experiência mediana, se tivesse contato com programação em um projeto acadêmico; (d) Experiência média-alta, se tivesse contato com programação em um projeto na indústria; ou (e) alta experiência, se tivesse experiência na indústria, com programação em vários projetos. Da mesma forma, os conhecimentos para experiência em projeto de *software* foram atribuídos.

Após caracterizar a experiência dos participantes, para mitigar as ameaças à validade quanto à distribuição dos participantes entre os grupos, foram aplicados os princípios de balanceamento, bloqueio e atribuição aleatória (WOHLIN *et al.*, 2012). Para o balanceamento, tentou-se criar grupos de tamanho igual, porém devido a algumas ausências no dia da execução do experimento, um grupo de cada rodada ficou com um número maior de integrantes. Em relação ao bloqueio, evitou-se que uma equipe tivesse participantes mais experientes do que a outra, a fim de evitar resultados tendenciosos de uma equipe que tivesse um desempenho melhor nas tarefas designadas. Finalmente, participantes de igual experiência foram aleatoriamente designados para os grupos.

O experimento compreendeu dois exercícios aplicados em dois dias (um exercício por dia). Os alunos que participaram de apenas um exercício foram removidos da análise. Além disso, os participantes que encontraram menos de 10% dos defeitos foram descartados como

outliers, pois como eles utilizaram a média de horas do restante para a realização da inspeção, seus resultados foram entendidos como os de alunos com dificuldade na compreensão da tarefa ou que simplesmente não se empenharam na atividade por alguma outra razão. Assim, dos 44 participantes do primeiro ensaio, os dados de 32 participantes foram considerados para a análise dos dados. Em relação ao segundo estudo, de 10 participantes, os dados de 8 participantes foram considerados.

5.2.6 DESENHO DO EXPERIMENTO

O desenho do experimento foi definido para incluir um fator com dois tratamentos (*ad-hoc* com ou sem a abordagem de definição de escopo com EMEs) e duas tarefas diferentes (exercícios de inspeção de artefatos, objetos do estudo). Adotou-se um desenho cruzado para mitigar as ameaças à validade do experimento no que diz respeito a: (i) diferenças entre tarefas experimentais (a influência do exercício proporcionado pelos materiais nos resultados); e (ii) o efeito de aprendizado (a influência da ordem em que os tratamentos são aplicados nos desfechos). É digno de nota que os princípios aplicados para distribuir os participantes entre os grupos ainda permitem comparar os resultados para cada exercício individual. O desenho cruzado utilizado é mostrado na Figura 16.

	Exercício A - passo um -	**Exercício B - passo dois -**
Grupo 1	*Ad-hoc*	*Definição de Escopo*
Grupo 2	*Definição de Escopo*	*Ad-hoc*

Figura 16: Configuração do desenho cruzado no experimento.

5.2.7 MATERIAIS UTILIZADOS

Os instrumentos utilizados neste experimento (apresentados no Apêndice B) foram: formulário de caracterização; material de treinamento em inspeção de modelos; descrição da tarefa, com instruções para realizar a inspeção, incluindo a taxonomia do defeito; um formulário de notificação de defeitos; um trecho de uma especificação funcional industrial real (a visão geral do módulo do sistema a ser inspecionado, os requisitos funcionais, os diagramas de casos de uso e suas descrições); um diagrama de classes UML industrial com defeitos semeados pelos autores; e um questionário de acompanhamento (ver Apêndice A). Quando a abordagem de

definição de escopo com EMEs foi aplicada, em vez do diagrama de classes UML, os participantes receberam o recorte (escopo) do diagrama de classes da UML e a lista de EMEs.

Os escopos dos modelos definidos foram preparados aplicando-se a abordagem de definição de escopo com EMEs nos diagramas de classe UML semeados com base nos trechos de especificação funcional. Para o módulo administrativo, foi identificada uma lista de EMEs nas informações contextuais (ou seja, visão geral, requisitos funcionais, diagrama de casos de uso e descrições de casos de uso) para quatro casos de uso pequenos (manutenção de dados da empresa, manutenção de dados do cliente, manutenção de informações fiscais e manutenção de centros de custo). Em seguida, usando essa lista de EMEs, foi definido o diagrama de classes da UML aplicando a abordagem de definição de escopo com EMEs, conforme descrito no capítulo anterior. Esse processo reduziu o diagrama de classes de domínio deste módulo de 19 para 12 classes.

Da mesma forma, para o módulo de faturamento, com base nos dois casos de uso selecionados (mais complexos) (registrando faturas de serviços prestados e registrando pagamentos de faturas), o diagrama de classes foi reduzido de 22 para 16 classes. A Figura 17 mostra os recortes realizados durante o escopo do módulo de faturamento.

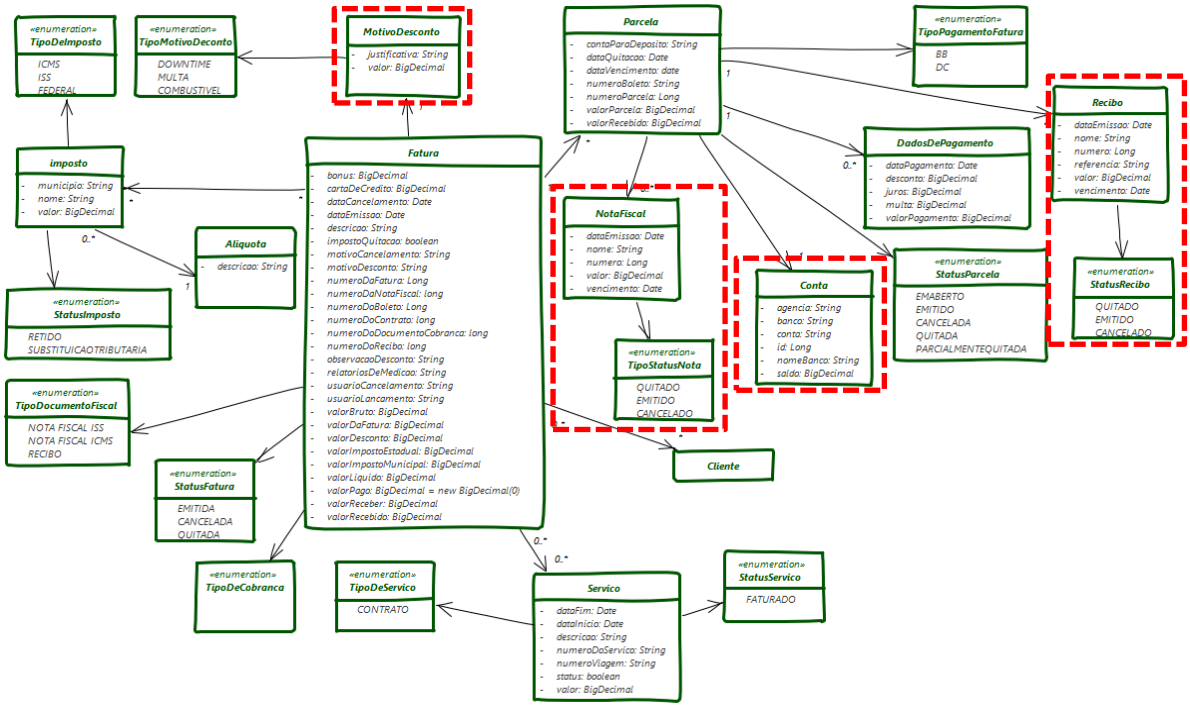


Figura 17: Recortes durante a definição de escopo (retângulos tracejados) para o módulo de faturamento.

Esses recortes estão relacionados a entidades referentes à emissão física de faturas (em comunicação com o sistema de emissão de faturas federal) e de outras receitas, que são detalhadas em dois outros casos de uso que não fizeram parte da seleção. Assim, essas partes do modelo apenas adicionariam complexidade à inspeção, já que não estavam relacionadas à seleção contra a qual o modelo deveria ser verificado.

Os defeitos, os tipos de defeitos (baseados na definição vista na Seção 2.2) e o nível de dificuldade para detectá-los foram definidos pela equipe de pesquisadores. Os defeitos, os diagramas inspecionados e os requisitos estão apresentados no Apêndice C. A distribuição dos tipos de defeitos nos diagramas está apresentada na Tabela 7.

Tabela 7: Tipos de defeitos semeados no diagrama

Tipo	Quantidade	Distribuição
Omissão	7	25%
Ambiguidade	7	25%
Fato Incorreto	7	25%
Informação Estranha	7	25%
Total	28	100%

A distribuição dos defeitos segundo a dificuldade de detecção atribuída pelos pesquisadores está apresentada na Tabela 8.

Tabela 8: Nível de dificuldade dos defeitos semeados no diagrama

Dificuldade	Quantidade	Distribuição
Fácil	9	32%
Médio	11	39%
Difícil	8	29%
Total	28	100%

Vale ressaltar que, enquanto o escopo permitia cortar algumas partes irrelevantes para o escopo de inspeção, muitas classes ainda permaneciam, dado que os casos de uso selecionados eram muito centrais para cada módulo. O processo, também, não foi polarizado, pois esses artefatos foram utilizados como fornecidos pelo parceiro industrial. O trecho da especificação funcional para a tarefa de inspeção foi planejado para uma inspeção de até 75 minutos (inspeções industriais são recomendadas para não levar mais que 2 horas). Por fim, ressalta-se que, se um desses módulos ainda tivesse centenas de casos de uso e centenas de classes de domínio, o corte para os casos de uso selecionados ainda selecionaria a mesma quantidade relevante de classes, relacionadas aos EMEs contidos no trecho.

Ao finalizar a inspeção, os participantes receberam um questionário para relatar a experiência durante o experimento, no qual deviam explicitar se o tempo foi suficiente, se precisaram de algum orientação adicional, quais dificuldades encontraram e o que poderia ter sido feito para tornar tarefa mais agradável, além de definir a complexidade da tarefa executada. Os participantes que executaram a tarefa utilizando a abordagem, receberam, ainda, um documento extra, que questionava a abordagem utilizada em relação a facilidade de uso, utilidade, intenção de adotá-la, etc., para realizar trabalhos de inspeção. Esse formulário permitiu maior compreensão sobre a inspeção realizada por ele.

5.2.8 OPERAÇÃO

O experimento foi conduzido em três dias. No primeiro dia, os participantes responderam ao formulário de caracterização para permitir dividi-los em grupos experimentais. Antes da execução do experimento, no segundo dia, conceitos básicos dos diagramas e tipos relevantes de defeitos foram revisados com os participantes em uma sessão de treinamento de 15 minutos. Depois disso, a inspeção foi conduzida da seguinte maneira.

Na primeira rodada (Exercício A), o Grupo 1 inspecionou o diagrama de classes UML completo com 19 classes, com base no documento de requisitos usando o tratamento *ad-hoc*. Por outro lado, os participantes do Grupo 2 (Escopo de Modelo com EMEs) inspecionaram o recorte do diagrama de classes da UML com 12 classes. Além do mesmo trecho exato de especificação funcional, os participantes do Grupo 2 usaram a lista de EMEs. Todos os participantes tiveram que relatar os defeitos encontrados no diagrama. No total, os inspetores tiveram 75 minutos para realizar a inspeção, inclusive relatando os defeitos encontrados. É importante mencionar que a comunicação entre os participantes não foi permitida. Após a inspeção, os participantes responderam ao questionário de acompanhamento (definido na subseção anterior).

No último dia, os procedimentos realizados para o Exercício A (módulo administrativo) foram repetidos para o Exercício B (módulo de faturamento). No entanto, o grupo que foi anteriormente atribuído ao tratamento *ad-hoc* foi agora atribuído ao tratamento de definição de escopo e vice-versa. Foram utilizados, exatamente, os mesmos procedimentos e materiais para ambos os ensaios. Na próxima seção, são apresentados os resultados do experimento.

5.3 RESULTADOS

Esta seção relata a análise de dados quantitativos e qualitativos coletados no experimento.

5.3.1 ANÁLISE QUANTITATIVA

Com a obtenção dos dados quantitativos do formulário de defeitos, contou-se o número de defeitos reais encontrados, os falsos positivos e tempo usado por cada participante. Seguindo o desenho da Figura 10, a Tabela 9 apresenta os resultados por indivíduo e o resultado geral.

Tabela 9: Resultados quantitativos por sujeito e tratamento.

		Exercício A					Exercício B				
	ID	**Duração (min)**	**Defeitos Encont.**	**Falsos Positivos**	**Eficácia**	**Eficiência**	**Duração (min)**	**Defeitos Encont.**	**Falsos Positivos**	**Eficácia**	**Eficiência**
Grupo 1	P1	71	8	4	29%	6,76	73	13	14	46%	10,68
	P2	72	10	4	36%	8,33	75	13	4	46%	10,40
	P3	72	15	8	54%	12,50	75	17	7	61%	13,60
	P4	69	13	6	46%	11,30	73	14	15	50%	11,51
	P5	68	7	11	25%	6,18	66	12	0	43%	10,91
	P6	71	9	4	32%	7,61	78	12	14	43%	9,23
	P7	68	10	6	36%	8,82	75	14	5	50%	11,20
	P8	66	7	4	25%	6,36	72	11	6	39%	9,17
	P9	70	9	5	32%	7,71	75	15	10	54%	12,00
	P10	69	10	4	36%	8,70	75	12	0	43%	9,60
	P11	72	17	9	61%	14,17	76	17	5	61%	13,42
	P12	70	9	12	32%	7,71	70	14	7	50%	12,00
	P13	72	6	13	21%	5,00	63	15	5	54%	14,29
	P14	69	4	8	14%	3,48	72	13	11	46%	10,83
	P15	70	12	9	43%	10,29	75	18	23	64%	14,40
	P16	72	7	7	25%	5,83	75	12	3	43%	9,60
	P17	69	15	13	54%	13,04	73	13	20	46%	10,68
	P18	69	11	9	39%	9,57	70	15	12	54%	12,86
	P33	68	18	4	64%	15,88	75	16	18	57%	12,80
	P34	75	9	7	32%	7,20	75	13	4	46%	10,40
	P35	71	8	4	29%	6,76	75	12	3	43%	9,60
	P36	75	7	13	25%	5,60	75	18	15	64%	14,40
Grupo 2	P19	60	11	4	39%	11,00	50	7	11	25%	8,40
	P20	67	12	4	43%	10,75	63	4	0	14%	3,81
	P21	51	14	5	50%	16,47	50	8	12	29%	9,60
	P22	71	9	3	32%	7,61	66	10	2	36%	9,09
	P23	72	16	10	57%	13,33	73	12	8	43%	9,86
	P24	72	9	9	32%	7,50	75	8	4	29%	6,40
	P25	71	11	6	39%	9,30	66	4	14	14%	3,64
	P26	57	10	4	36%	10,53	74	4	7	14%	3,24
	P27	63	15	10	54%	14,29	75	12	3	43%	9,60
	P28	60	9	3	32%	9,00	60	10	5	36%	10,00
	P29	67	9	15	32%	8,06	52	13	6	46%	15,00
	P30	72	6	14	21%	5,00	75	7	4	25%	5,60
	P31	61	12	4	43%	11,80	65	6	5	21%	5,54
	P32	62	14	7	50%	13,55	55	9	6	32%	9,82
	P37	75	16	2	57%	12,80	75	8	2	29%	6,40
	P38	68	13	4	46%	11,47	75	3	17	11%	2,40
	P39	75	12	3	43%	9,60	59	8	18	29%	8,14
	P40	75	11	0	39%	8.80	75	9	3	32%	7.20

Realizaram-se, posteriormente, análises estatísticas utilizando a ferramenta estatística SPSS v 25.0.0. Para o teste de hipóteses, utilizou-se o teste não paramétrico de Mann-Whitney com $\alpha = 0{,}10$. Este nível de significância estatística foi achado aceitável para experimentos de engenharia de *software*, que tipicamente envolvem amostras pequenas (Dybå *et al.*, 2006). Além disso, para aumentar o tamanho da amostra, decidiu-se agregar os resultados de ambos os ensaios, o que não expõe à ameaças adicionais à validade, uma vez que os ensaios foram replicações internas exatas.

A Figura 18 mostra os resultados descritivos como diagramas de caixa (*boxplots*) que comparam o indicador de eficácia de ambos os exercícios (A e B). É possível observar que a mediana do tratamento da abordagem de definição de escopo com EMEs em ambos os exercícios (mediana de A 0,41, média de A 0,41, mediana de B 0,48, média de B 0,49) é superior à mediana do tratamento *ad-hoc* (mediana de A 0,32, média de A 0,36, mediana de B 0,28, média de B 0,29). Os tamanhos de efeito (diferença média padronizada entre duas populações) para os exercícios A e B são, respectivamente, 0,47 (médio) e 1,57 (muito grande). Assim, o grupo que utilizou a abordagem de definição de escopo com EMEs foi mais eficaz que o grupo de controle. Além disso, o teste de Mann-Whitney mostrou uma diferença significativa entre os grupos ($p = 0{,}075$ para o exercício A e $p = 0{,}001$ para o exercício B).

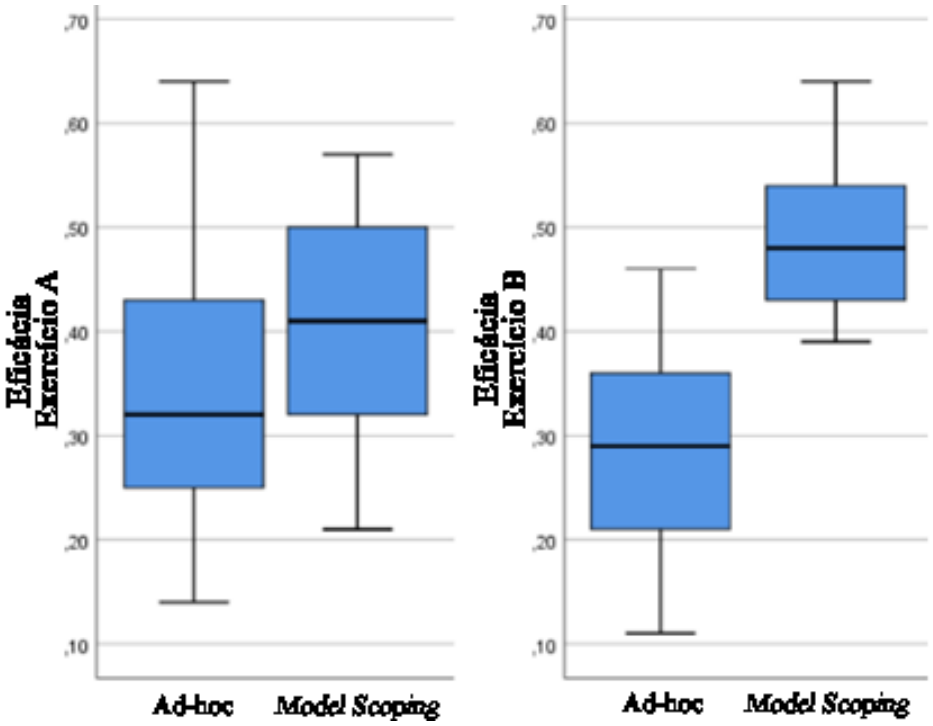


Figura 18: Indicador de eficácia (defeitos encontrados / defeitos semeados) para detecção de defeitos.

Esses resultados sugerem que o uso da abordagem de definição de escopo com EMEs permitiu que os inspetores alcançassem maior eficácia na detecção de defeitos. Esses resultados permitem rejeitar a hipótese nula H_{01}.

A Figura 19 mostra *boxplots* comparando a eficiência dos tratamentos.

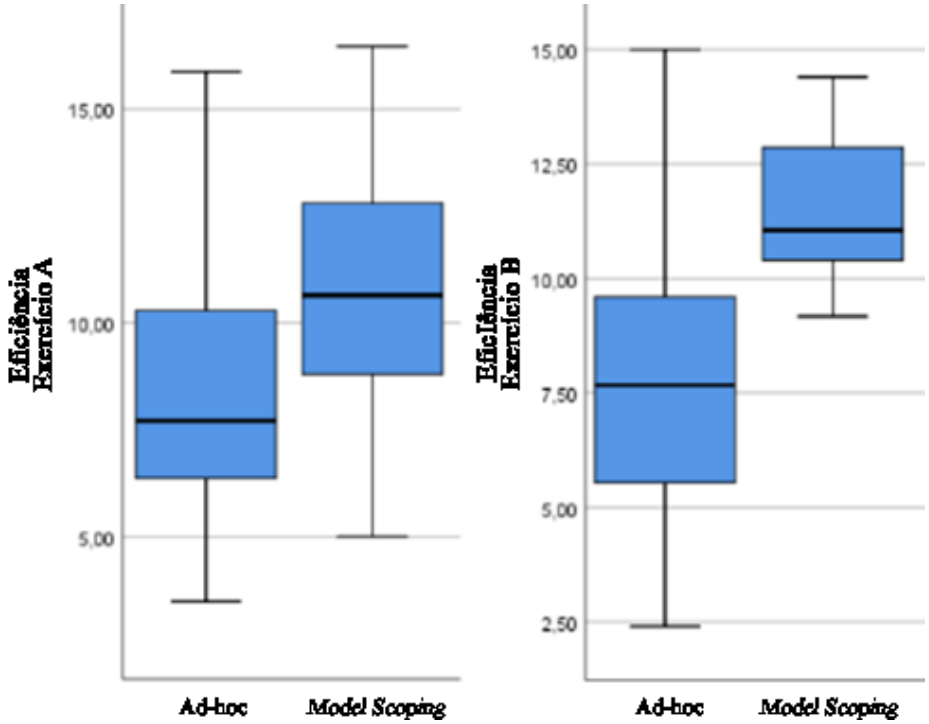


Figura 19: Indicador de eficiência (defeitos encontrados por hora) para detecção de defeitos.

Para eficiência também é possível observar que a mediana do tratamento de definição de escopo em ambos os exercícios (mediana de A10,64, média de A 10,60, mediana de B 11,05, média de B 11,53) é maior que a mediana do tratamento *ad-hoc* (mediana de A 7,71, média de A 8,58, mediana de B 7,67, média de B 7,43). Os tamanhos de efeito para os exercícios A e B são, respectivamente, 0,65 (médio a grande) e 1,30 (muito grande).

Assim, o grupo que utilizou a abordagem de definição de escopo com EMEs foi mais eficiente quando comparado ao grupo controle. Novamente, o teste de Mann-Whitney mostra que a diferença entre os grupos é estatisticamente significativa ($p = 0,024$ para o exercício A e $p = 0,001$ para o exercício B). Assim, o uso da abordagem de definição de escopo com EMEs permitiu que os inspetores atingissem maior eficiência. Esses resultados permitem rejeitar a hipótese nula H_{02}.

5.3.2 ANÁLISE QUALITATIVA

Foram coletados dados qualitativos dos questionários de acompanhamento. Os participantes avaliaram sua percepção sobre a complexidade da tarefa (como muito complexa, complexa, simples ou muito simples) e notaram dificuldades que tiveram durante o experimento. Os participantes que utilizaram o tratamento de definição de escopo com EMEs (que recebeu uma lista de EMEs para apoiar a inspeção) preencheram o questionário TAM com uma escala Likert de cinco pontos (discordo completamente, discordo, neutro, concordo e concordo completamente) para as perguntas sobre utilidade percebida, facilidade de uso e intenção de adoção.

Em relação à complexidade percebida da tarefa, mostrada na Tabela 10, houve uma diferença sutil entre os tratamentos. Embora globalmente as tarefas fossem consideradas complexas pelos participantes, o que era esperado, já que lidavam com artefatos industriais reais, a quantidade de participantes que percebia a complexidade como simples era maior para a abordagem de definição de escopo com tratamento de EMEs. Em particular, é fácil observar que o exercício B (o módulo de faturamento, que tinha regras de negócios mais complexas) foi percebido como mais complexo pelos participantes ao usar o tratamento *ad-hoc*. Assim, a abordagem de definição de escopo e os EMEs parecem ter reduzido a complexidade percebida. Por exemplo, o participante P34 mencionou *"dificuldades para entender a descrição dos casos de uso e depois comparar com o diagrama de classes"* e sugeriu que *"o diagrama poderia ser [...] menor"*. O participante P1 mencionou que *"navegar pelas inúmeras descrições ao pesquisar defeitos é uma tarefa complexa [...], é necessário algum tipo de orientação"*.

Tabela 10: Complexidade das tarefas experimentais percebidas pelos participantes.

Tratamento	Módulo	Complexidade			
		Muito Simples	Simples	Complexo	Muito Complexo
Ad-hoc	Administrativo	0%	5%	90%	5%
	Faturamento	0%	12,5%	50%	37,5%
Definição de Escopo	Administrativo	0%	15%	75%	10%
	Faturamento	0%	22,5%	72,5%	5%

O questionário TAM foi aplicado apenas com o tratamento de definição de escopo com EMEs. Em geral, os inspetores perceberam receber as EMEs para apoiar a inspeção como útil, fácil de usar e gostaria de receber tal apoio. Observe que os participantes não sabiam que estavam inspecionando um modelo com escopo definido. De fato, todos os participantes

concordaram ou concordaram completamente com as perguntas do TAM. Além disso, notou-se que muitos inspetores, que usaram o tratamento de definição de escopo com EMEs no primeiro exercício, mencionaram que perderam esse tipo de apoio no segundo exercício. Por exemplo, o participante P32 mencionou que o exercício continha *"muita informação"* e que era *"difícil entender o que fazer sem um guia"*. Ele também pediu *"a página extra (EMEs) do exercício anterior"*. O participante P28 mencionou dificuldades em *"identificar defeitos nas relações entre classes"* e informou que *"o uso de EMEs"* poderia tornar a tarefa mais agradável.

5.4 DISCUSSÃO DOS RESULTADOS

Nesta seção, serão discutidos os resultados das análises quantitativa e qualitativa dos dados dos experimentos.

O principal objetivo da análise quantitativa foi investigar como a abordagem de definição de escopo com EMEs afetaria a eficácia e a eficiência dos inspetores ao revisar modelos baseados em documentos de referência. Foram selecionados diagramas de classes industriais reais e suas especificações funcionais relacionadas como documentos de referência para dois módulos de um sistema de gerenciamento integrado. Como entrada para o experimento, foram definidos um conjunto de casos de uso (e suas informações contextuais) para cada módulo e, assim, foram conduzidas a atividade de definição de escopo com EMEs, conforme descrito na Seção 4.

Os resultados do experimento indicam que a aplicação da abordagem de definição de escopo com EMEs antes da inspeção melhorou a eficácia e a eficiência dos inspetores ao revisar os diagramas de classe UML em relação aos trechos de especificação funcional. Além disso, os resultados foram estatisticamente significativos e tiveram grandes tamanhos de efeito.

Como complemento, a análise qualitativa indicou que os inspetores percebem suas tarefas de inspeção como menos complexas se a abordagem de definição de escopo com EMEs fosse aplicada antes da inspeção (ou seja, eles inspecionam um modelo com definição de escopo e recebem uma lista de EMEs). Eles também perceberam ser útil receber EMEs candidatos para apoiar a verificação do modelo em relação aos documentos de referência.

Esses resultados indicam a aplicação de definição de escopo com EMEs antes de inspeções em situações nas quais grandes diagramas de classes UML devem ser inspecionados contra trechos (ou incrementos) de especificações funcionais. Isso está de acordo com os resultados de Briand *et al.* (2014), que também observaram reduções de esforço quando modelos de escopo para suas finalidades específicas (abordando requisitos de segurança). Embora possa ser assumido que a abordagem de definição de escopo com EMEs é aplicável a

qualquer tipo de modelo, limita-se o aconselhamento e as recomendações às descobertas desse ambiente experimental específico.

Também é digno de nota que aplicar adequadamente a abordagem de definição de escopo com EMEs requer algum esforço e ser capaz de identificar corretamente os EMEs em partes selecionadas do documento de referência. O autor levou 1 hora para identificar os EMEs dentro de cada trecho de especificação funcional e, em seguida, usou os EMEs para escopo do modelo. Ainda assim, acredita-se que o esforço vale a pena em muitos casos (em especial para grandes modelos), considerando que as equipes de inspeção geralmente envolvem de três a cinco inspetores, que percebem sua tarefa como menos complexa e mais eficaz e eficiente. Abordagens para identificar os elementos do modelo esperado no texto natural foram investigadas (SABOU *et al.*, 2018) e poderiam ser usadas neste contexto.

5.5 AMEAÇAS À VALIDADE

Nessa seção são apresentadas e discutidas as ameaças à validade dos experimentos realizados neste trabalho, organizado por categorias descritas por Wohlin *et al.* (2012).

5.5.1 VALIDADE INTERNA

As tarefas do experimento foram realizadas pelos participantes individualmente e sob a supervisão de um dos pesquisadores. A comunicação entre os participantes não foi permitida.

Depois de caracterizar a experiência dos participantes, os princípios de balanceamento, bloqueio e atribuição aleatória (WOHLIN *et al.*, 2012) foram aplicados para mitigar ameaças à validade em relação à distribuição de participantes entre grupos.

5.5.2 VALIDADE EXTERNA

Quanto à representatividade de artefatos, foram utilizados diagramas de classes UML industriais reais e especificações funcionais. Ainda assim, eles representam artefatos de uma organização específica. Por representatividade de assunto, foram utilizados estudantes para representar inspetores novatos. O uso de estudantes para este propósito é uma abstração válida, em particular considerando que eles foram adequadamente caracterizados (FALESSI *et al.*, 2018).

Ainda assim, os artefatos são de uma organização específica e os assuntos vêm de um contexto específico. Portanto, não há alegações de validade externa ao longo do trabalho e a validade do estudo é específica para o contexto em que foi realizada. De fato, são exigidas

replicações envolvendo uma variedade de artefatos (por exemplo, modelos e documentos de referência) e contextos para reforçar a evidência experimental.

5.5.3 VALIDADE DE CONSTRUÇÃO

Em relação ao tratamento, a tarefa experimental envolveu inspeções usando artefatos industriais reais. Foi empregado um desenho cruzado para isolar os fatores de confusão relacionados às tarefas experimentais (ou seja, os exercícios) e o efeito de aprendizagem. Além disso, é importante notar que as métricas usadas para medir eficácia e eficiência têm sido amplamente utilizadas em estudos empíricos sobre inspeção de *software*.

Uma ameaça potencial à validade de construção refere-se aos defeitos semeados nos diagramas. Para mitigar essa ameaça, os defeitos foram distribuídos de forma harmônica quanto aos seus tipos, bem como os níveis de dificuldade de detecção.

5.5.4 VALIDADE DE CONCLUSÃO

Para melhorar a validade de conclusão, agregou-se os participantes de ambos os estudos para aumentar o tamanho da amostra para análise de dados. Isso foi possível, dado que haviam duas replicações internas exatas seguindo o mesmo projeto de desenho cruzado. *Outliers* foram cuidadosamente removidos para evitar influência nos resultados. Foram aplicados testes estatísticos apropriados e os resultados foram estatisticamente significativos com grandes tamanhos de efeito. Com base nesses resultados, as conclusões tiradas são válidas para o cenário experimental relatado.

5.6 CONSIDERAÇÕES FINAIS

A pesquisa consistiu em planejar, conduzir e analisar os resultados de estudos experimentais. Para a avaliação, foi conduzida uma experiência controlada com estudantes usando artefatos industriais reais com o objetivo de entender como a abordagem baseada em definição de escopo com EMEs influenciaria a eficácia e a eficiência da inspeção do modelo. Os resultados do experimento indicam, com significância estatística, que a utilização da abordagem de definição de escopo de modelo UML com EMEs, antes da inspeção, melhorou a eficácia e eficiência dos inspetores.

CAPÍTULO 6 – CONCLUSÕES

6.1 CONTRIBUIÇÕES

Nesta dissertação foi apresentado o conceito de definição de escopo como uma peça de modelo bem definida que atua como um filtro ou visualização que mostra apenas elementos relevantes do modelo. Foi proposta e avaliada uma abordagem baseada em definição de escopo com os elementos do modelo esperado. A abordagem consiste em identificar os Elementos do Modelo Esperados (EMEs) nas partes selecionadas do documento de referência e, em seguida, usar esses EMEs para definir o modelo e orientar os inspetores durante a detecção de defeitos.

Para a avaliação, foi conduzida uma experiência controlada com estudantes usando artefatos industriais reais com o objetivo de entender como a abordagem baseada em definição de escopo com EMEs influenciaria a eficácia e a eficiência da inspeção do modelo. Os resultados do experimento indicam, com significância estatística, que a aplicação da abordagem de Escopo com EMEs antes da inspeção melhorou tanto a eficácia quanto a eficiência dos inspetores ao revisar os diagramas de classe UML contra os trechos de especificação funcional. Além disso, dados qualitativos indicaram que os inspetores percebem suas tarefas de inspeção menos complexas quando a abordagem de definição de escopo com EMEs foi aplicada antes da inspeção.

Embora este documento apresente a primeira abordagem de Definição de Escopo de Modelo baseado em documentos de referência e influencie positivamente a eficácia e eficiência da inspeção de modelos, aplicá-la adequadamente requer algum esforço, assim como ser capaz de identificar corretamente EMEs em partes selecionadas do documento de referência.

6.2 LIMITAÇÕES

Muito embora a solução proposta seja eficiente e eficaz o suficiente para dar suporte à inspeção de modelos de *softwares* de larga escala, a abordagem de definição de escopo com EMEs possui algumas limitações e aspectos que podem ser melhorados. A saber:

- Os resultados do mapeamento sistemático apresentaram poucos trabalhos relacionados e os mesmos não puderam ser utilizados para a elaboração e validação da abordagem proposta, visto que nenhum dos trabalhos retornados apresentava a definição de escopo e a lista de EMEs aplicadas a modelos UML de larga escala.
- Os estudos de viabilidade conduzidos para avaliar a eficiência e eficácia da abordagem de definição de escopo com EMEs utilizaram modelos de apenas um

projeto. O projeto utilizado é de larga escala, dividido em diversos módulos, porém foram utilizados alguns diagramas de classe de apenas dois módulo deste projeto.

- A abordagem foi aplicada apenas para avaliar defeitos em um modelo da UML, Diagramas de Classes e, além disso, não foi possível verificar a eficiência e eficácia de detecção de todos os tipos de defeitos, por exemplo, a inconsistência.
- Além disso, a abordagem foi aplicada e avaliada por estudantes da academia, ou seja, ela não foi avaliada (nem refinada) em ambientes industriais reais.

6.3 TRABALHOS FUTUROS

Este trabalho possui diversas oportunidades para trabalhos futuros, em particular, para tratar suas limitações e explorar outros contextos de avaliação. A saber:

- Expandir o experimento relatado utilizando a Abordagem baseada em Definição de Escopo com EMEs, incluindo o uso com todos os diagramas, em outros contextos, para reforçar a evidência experimental e melhorar a validade externa.
- Expandir o experimento relatado utilizando a Abordagem baseada em Definição de Escopo com EMEs em ambientes industriais reais com especialistas, para comparar com os resultados do experimento aplicado na academia.
- Implementar uma abordagem para a definição dos Elementos do Modelo Esperados (EMEs), reduzindo o esforço considerável na definição e validação dos mesmos.

REFERÊNCIAS

ALSHAZLY, A. A.; ELFATATRY, A. M.; ABOUGABAL, M. S. *Original Article: Detecting defects in software requirements specification.* Alexandria Engineering Journal, v. 53, p. 513–527, 2014.

ANDA, B.; SJØBERG, D. I. K. *Towards an inspection technique for use case models.* Proceedings of the 14th international conference on *Software* engineering and knowledge engineering SEKE 02, n. 1325, p. 127, 2002.

AURUM, A.; PETERSSON, H.; WOHLIN, C. *State-of-the-Art: Software* Inspections after 25 Years. Verification and Reliability, v. 12, n. 3, p. 133–154, 2002.

BASILI, V. R. *Packaging Researcher Experience to Assist Replication of Experiments.* Proc ISERN Meeting. Anais...1996

BERLING, T.; THELIN, T. *A case study of reading techniques in a software company.* Proceedings - 2004 International Symposium on Empirical *Software* Engineering, ISESE 2004. Anais...Redondo Beach, CA: 2004. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-11244309672&doi=10.1109%2FISESE.2004.1334910&partnerID=40&md5=83aec9829058dd2de872914030aa4cbc>

BIFFL, S.; GUTJAHR, W. *Influence of team size and defect detection technique on inspection effectiveness.* Proceedings Seventh International *Software* Metrics Symposium. Anais...IEEE Comput. Soc, 2001Disponível em: <http://ieeexplore.ieee.org/document/915516/>. Acesso em: 16 jul. 2018

BOEHM, B.; BASILI, V. R. *Software Defect Reduction.* Top 10 List Computer, 2001.

BOEHM, B. W. *Software Engineering Economics.* [s.l: s.n.].

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *UML: Guia do usuário.* Rio de Janeiro: Elsevier Brasil, 2006.

BRIAND, L. *et al. Traceability and SysML design slices to support safety inspections.* ACM Transactions on *Software* Engineering and Methodology, 2014.

BUDGEN, D. *et al. Using Mapping Studies in Software Engineering.* Proceedings of PPIG. Anais...2008Disponível em: <http://www.ppig.org/papers/20th-budgen.pdf%5Cnhttp://www.inf.puc-rio.br/~inf2921/2014_2/docs/artigos/Using Mapping Studies in *Software* Engineering.pdf>

CONRADI, R. *et al. A study of inspections and testing at Ericsson*, Norway. Proceedings of the International Conference on Product Focused *Software* Process Improvement (PROFES´99). Anais...Oulu, Finland: 1999Disponível em: <https://pdfs.semanticscholar.org/3f50/ae5daea81bdde16a99fc3def40ec56b83b92.pdf>. Acesso em: 16 jul. 2018.

DAVIS, F.D., 1989. P*erceived Usefulness*, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly, vol. 13.

DENGER, C.; CIOLKOWSKI, M. *High quality statecharts through tailored, perspective-based inspections*. 2003 Proceedings 29th Euromicro Conference, 2003.

DUNSMORE, A.; ROPER, M.; WOOD, M. *Practical code inspection techniques for object-oriented systems: an experimental comparison*. IEEE *Software*, 2003a.

DUNSMORE, A.; ROPER, M.; WOOD, M. T*he development and evaluation of three diverse techniques for object-oriented code inspection*. IEEE Transactions on *Software* Engineering, 2003b.

DYBÅ, T., KAMPENES, V. B., SJ Berg, D. I. K., 2006. *A systematic review of statistical power in Software Engineering experiments.* Information and *Software* Technology 48 (8):745-755.

ELBERZHAGER, F., MÜNCH, J., NHA, V.T.N., 2012. *A systematic mapping study on the combination of static and dynamic quality assurance techniques.* In: Information and *Software* Technology, 54(1):1-15

FAGAN, M. E.; E., M. *Design and code inspections to reduce errors in program development.* IBM Systems Journal, v. 15, n. 3, p. 182–211, 1976.

FURLAN, J. D. *Modelagem de objetos através da UML - the unified modeling language.* São Paulo: Makron books, 1998.

FUSARO, P.; LANUBILE, F.; VISAGGIO, G. *A Replicated Experiment to Assess Requirements Inspection Techniques.* Empirical *Software* Engineering, v. 2, n. 1, p. 39–57, 1997.

GENERO, M. *et al. Assessing the Influence of Stereotypes on the Comprehension of UML: A Controlled Experiment.* MODEL DRIVEN ENGINEERING LANGUAGES AND SYSTEMS, PROCEEDINGS — 2008, Volume 5301, p. 280–294, 2008.

GERALDI, R. T. *et al. Checklist-based inspection of SMarty variability models proposal and empirical feasibility study.* (T. E. M. L. Hammoudi S. Maciaszek L., Ed.)ICEIS 2015 - 17th International Conference on Enterprise Information Systems, Proceedings. Anais...SciTePress, 2015Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84939537537&partnerID=40&md5=253f060f7bbeb052f7d7a9f5082df5b0>

GILB, T.; GRAHAM, D.; FINZI, S. *Software inspection.* [s.l.] Addison-Wesley, 1993.

HE, J.; WANG, L.; ZHAO, J. *Supporting Automatic Code Review via Design*. 2013 IEEE Seventh International Conference on *Software* Security and Reliability Companion, 2013.

HUNGERFORD, B. C.; HEVNER, A. R.; COLLINS, R. W. *Reviewing software diagrams: a cognitive study*. IEEE Transactions on *Software* Engineering, v. 30, n. 2, p. 82–96, fev. 2004.

KALINOWSKI, M.; CARD, D. N.; TRAVASSOS, G. H. *Evidence-Based Guidelines to Defect Causal Analysis.* IEEE *Software*, v. 29, n. 4, p. 16–18, jul. 2012.

KALINOWSKI, M.; SPINOLA, R. O.; TRAVASSOS, G. H. *Infra-Estrutura Computacional para Apoio ao Processo de Inspeção de Software.* III Simpósio Brasileiro de Qualidade de *Software*. Anais...Brasília: 2004. Disponível em: <https://www.researchgate.net/publication/242703915>. Acesso em: 25 jul. 2018

KROLL, P.; KRUCHTEN, P. *The rational unified process made easy : a practitioner's guide to the RUP.* [s.l.] Addison-Wesley, 2003.

LAITENBERGER, O. *et al. Experimental comparison of reading techniques for defect detection in UML design documents. Journal of Systems and Software*, v. 53, n. 2, p. 183–204, 2000.

LAITENBERGER, O.; ATKINSON, C. *Generalizing perspective-based inspection to handle object-oriented development artifacts.* Proceedings of the 21st international conference on *Software* engineering - ICSE '99. Anais...New York, New York, USA: ACM Press, 1999Disponível em: <http://portal.acm.org/citation.cfm?doid=302405.302680>. Acesso em: 19 mar. 2017

LAITENBERGER, O.; DEBAUD, J. M. *Perspective-based reading of code documents at Robert Bosch GmbH.* Information and *Software* Technology, 1997.

LAITENBERGER, O., Debaud, J.M., 2000. *An encompassing life cycle centric survey of software inspection.* In: J. of Syst. and *Software*, 50(1):5-31.

LAITENBERGER, O.; EL EMAM, K.; HARBICH, T. G. *An internally replicated quasi-experimental comparison of checklist and perspective based reading of code documents.* IEEE Transactions on *Software* Engineering, v. 27, n. 5, p. 387–421, maio 2001.

LANGE, C.F., CHAUDRON, M.R., 2005. *Managing model quality in UML-based software development.* IEEE International Workshop on *Software* Technology and Engineering Practice, pages 7-16.

LARMAN, C., 2004. *Applying UML and Patterns.* 3rd Edition, Prentice Hall.

LOTT, C. M.; ROMBACH, H. D. *Repeatable software engineering experiments for comparing defect-detection techniques.* Empirical *Software* Engineering, 1996.

MAFRA, S. N. *Definição de uma Técnica de Leitura Baseada em Perspectiva (OO-PBR) Apoiada por Estudos Experimentais.* [s.l.] PESC/COPPE/UFRJ, 2006.

MARIN, B. *et al. A Tool for Automatic Defect Detection in Models Used in Model-Driven Engineering.* 2010 Seventh International Conference on the Quality of Information and Communications Technology, 2010.

MASSOLLAR, J. L.; MELLO, R. M. DE; TRAVASSOS, G. H. *Investigating the Feasibility of a Specification and Quality Assessment Approach Suitable for Web Functional Requirements.* 2011 30th International Conference of the Chilean Computer Science Society. Anais...IEEE, nov. 2011Disponível em: <http://ieeexplore.ieee.org/document/6486569/>. Acesso em: 19 mar. 2017

OBERKAMPF, W. L.; TRUCANO, T. G. *Verification and validation benchmarks*. Nuclear Engineering and Design, v. 238, n. 3, p. 716–743, 1 mar. 2008.

OMG. *Unified Modeling Language Specification, Superstructure- Version 2.5.1.* Disponível em: <https://www.omg.org/spec/UML/2.5.1/PDF>.

PETERSEN, K. *et al. Systematic Mapping Studies in Software Engineering.* Proceedings of the 12th International Conference on Evaluation and Assessment in *Software* Engineering. Anais...: EASE'08.Swindon, UK: BCS Learning & Development Ltd., 2008Disponível em: <http://dl.acm.org/citation.cfm?id=2227115.2227123>

PORTER, A. A.; VOTTA, L. G. *An experiment to assess different defect detection methods for software requirements inspections.* Proceedings of 16th International Conference on *Software* Engineering, 1994.

ROCHA, A. C. O.; RAMALHO, F.; MACHADO, P. D. L. *Automating test-based inspection of design models. Software* Quality Journal, v. 23, n. 1, p. 3–28, 2015.

SABALIAUSKAITE, G. *et al. An experimental comparison of checklist-based reading and perspective-based reading for UML design document inspection.* ISESE 2002 - Proceedings, 2002 International Symposium on Empirical *Software* Engineering. Anais...IEEE Comput. Soc, 2002. Disponível em: <http://ieeexplore.ieee.org/document/1166934/>. Acesso em: 19 mar. 2017

SABALIAUSKAITE G., MATSUKAWA F., KUSUMOTO S., INOUE K., 2003. *Further investigations of reading techniques for object-oriented design inspection.* Information and *Software* Technology, 45(9): 571–585.

SABALIAUSKAITE, G.; KUSUMOTO, S.; INOUE, K. *Assessing defect detection performance of interacting teams in object-oriented design inspection.* Information and *Software* Technology, v. 46, n. 13, p. 875–886, 2004.

SABOU, M.; WINKLER, D.; PETROVIC, S. *Expert Sourcing to Support the Identification of Model Elements in System Descriptions.* In: [s.l.] Springer, Cham, 2018. p. 83–99.

SAUER, C. *et al. The effectiveness of software development technical reviews: A behaviorally motivated program of research.* IEEE Transactions on *Software* Engineering 26, 1, 2000.

SHULL, F.; CARVER, J.; TRAVASSOS, G. H. *An empirical methodology for introducing software processes.* ACM SIGSOFT *Software* Engineering Notes, 2001.

SHULL, F. J. *Developing Techniques for Using Software Documents: A Series of Empirical Studies.* Dissertation Abstracts International, B: Sciences and Engineering, 1999.

STARON, M.; KUZNIARZ, L.; THURN, C. *An Empirical Assessment of Using Stereotypes to Improve Reading Techniques in Software Inspections.* Proceedings of the Third Workshop on *Software* Quality. Anais...: 3-WoSQ.New York, NY, USA: ACM, 2005Disponível em: <http://doi.acm.org/10.1145/1083292.1083308>

THELIN, T.; RUNESON, P.; WOHLIN, C. *Prioritized use cases as a vehicle for software inspections.* IEEE *Software*, v. 20, n. 4, p. 30–33, 2003.

THEOCHARIS, G., KUHRMANN, M., MÜNCH, J. DIEBOLD, P., 2015. *Is water-scrum-fall reality? on the use of agile and traditional development practices.* International Conference on Product-Focused *Software* Process Improvement, pages 149-166.

TRAVASSOS, G. *et al. Detecting defects in object-oriented designs.* Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and

applications - OOPSLA '99. Anais...New York, New York, USA: ACM Press, 1999. Disponível em: <http://portal.acm.org/citation.cfm?doid=320384.320389>. Acesso em: 16 jul. 2018

TRAVASSOS, G. H. *Revisões de Software.* In: ROCHA, A. R. C. .; MALDONADO, J. C. E WEBER, K. C. (Eds.). . *Qualidade de Software - Teoria e Prática.* 1. ed. São Paulo - SP: Ed Makron Books., 2001. p. 84 – 94.

TRAVASSOS, G. H. *Software defects: Stay away from them. Do inspections!* Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology, QUATIC 2014. Anais...2014

TRAVASSOS, G. H.; SHULL, F.; CARVER, J. *Working with UML: A software design process based on inspections for the unified modeling language.* Advances in Computers, v. 54, p. 35–98, 1 jan. 2002.

TURNER, M., KITCHENHAM, B., BRERETON, P., 2010. *Does the technology acceptance model predict actual use? A systematic literature review.* Information and *Software* Technology, vol. 52: 463-479.

VAN SOLINGEN, R. *et al. Goal Question Metric (GQM) Approach.* In: Encyclopedia of *Software* Engineering. [s.l: s.n.].

WALLIN, C. *Verification and validation of software components and component based software systems.* Building Reliable Component-Based *Software* Systems. Artech House Publishers, 2002.

WIERINGA, R. *et al. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion.* Requirements Engineering, v. 11, n. 1, p. 102–107, 2006.

WINKLER, D. *et al. Improving Model Inspection with Crowdsourcing*. 2017 IEEE/ACM 4th International Workshop on CrowdSourcing in *Software* Engineering (CSI-SE), 2017a.

WINKLER, D. *et al. Improving Model Inspection Processes with Crowdsourcing: Findings from a Controlled Experiment.* In: [s.l.] Springer, Cham, 2017b. p. 125–137.

WOHLIN, C. *et al. Experimentation in software engineering: an introduction.* [s.l: s.n.].

WOHLIN, C. *et al. Experimentation in Software Engineering.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

ZELKOWITZ, M. V.; WALLACE, D. R. *Experimental models for validating technology.* Computer, v. 31, n. 5, p. 23–31, maio 1998.

APÊNDICE A – LISTA DE ESTUDOS SELECIONADOS PARA O MAPEAMENTO SISTEMÁTICO

P1 Chan, L., Jiang, K., & Karunasekera, S. (2005, March). A tool to support perspective based approach to *software* code inspection. In *Software* Engineering Conference, 2005. Proceedings. 2005 Australian (pp. 110-117). IEEE.

P2 Conradi, R., Mohagheghi, P., Arif, T., Hegde, L. C., Bunde, G. A., & Pedersen, A. (2003, July). Object-oriented reading techniques for inspection of UML models–an industrial experiment. In European Conference on Object-Oriented Programming (pp. 483-500). Springer, Berlin, Heidelberg.

P3 Cooper, D. J., von Konsky, B. R., Robey, M. C., & McMeekin, D. A. (2007, April). Obstacles to comprehension in usage based reading. In *Software* Engineering Conference, 2007. ASWEC 2007. 18th Australian (pp. 233-244). IEEE.

P4 Denger, C., & Ciolkowski, M. (2003, September). High quality statecharts through tailored, perspective-based inspections. In null (p. 316). IEEE.

P5 Dunsmore, A., Roper, M., & Wood, M. (2002, May). Further investigations into the development and evaluation of reading techniques for object-oriented code inspection. In Proceedings of the 24th international conference on *Software* engineering (pp. 47-57). ACM.

P6 Dunsmore, A., Roper, M., & Wood, M. (2003). Practical code inspection techniques for object-oriented systems: an experimental comparison. IEEE *software*, 20(4), 21-29.

P7 Dunsmore, A., Roper, M., & Wood, M. (2003). The development and evaluation of three diverse techniques for object-oriented code inspection. IEEE transactions on *software* engineering, 29(8), 677-686.

P8 He, J., Wang, L., & Zhao, J. (2013, June). Supporting Automatic Code Review via Design. In *Software* Security and Reliability-Companion (SERE-C), 2013 IEEE 7th International Conference on (pp. 211-218). IEEE.

P9 Hungerford, B. C., Hevner, A. R., & Collins, R. W. (2004). Reviewing *software* diagrams: A cognitive study. IEEE Transactions on *Software* Engineering, 30(2), 82-96..

P10 Kirner, T. G., & da Cruz, E. R. (2007, July). Evaluation of the OORT Techniques for Inspection of Requirements Specifications in UML: an Empirical Study. In SEKE (p. 649).

P11 Laitenberger, O., & Atkinson, C. (1999, May). Generalizing perspective-based inspection to handle object-oriented development artifacts. In Proceedings of the 21st international conference on *Software* engineering (pp. 494-503). ACM.

P12 Laitenberger, O., Atkinson, C., Schlich, M., & El Emam, K. (2000). An experimental comparison of reading techniques for defect detection in UML design documents. Journal of Systems and *Software*, 53(2), 183-204.

P13 Marín, B., Giachetti, G., Pastor, O., & Vos, T. E. (2010, September). A tool for automatic defect detection in models used in model-driven engineering. In International Conference on the Quality of Information and Communications Technology (pp. 242-247). IEEE.

P14 Massollar, J. L., de Mello, R. M., & Travassos, G. H. (2011, November). Investigating the Feasibility of a Specification and Quality Assessment Approach Suitable for Web Functional Requirements. In Chilean Computer Science Society (SCCC), 2011 30th International Conference of the (pp. 108-117). IEEE.

P15 McMeekin, D., von Konsky, B., Robey, M., & Cooper, D. (2009). The significance of participant experience when evaluating *software* inspection techniques. In Proceedings of the 20th Australian *software* engineering conference (ASWEC 2009) (pp. 200-209). IEEE Computer Society.

P16 Mohan, K. K., Harun, R. S., Srividya, A., & Verma, A. K. (2010). Quality framework for reliability improvement in SAP netweaver business intelligence environment through lean *software* development—a practical perspective. International Journal of System Assurance Engineering and Management, 1(4), 316-323.

P17 Ogata, S., & Matsuura, S. (2013). A review method for UML requirements analysis model employing system-side prototyping. SpringerPlus, 2(1), 134.

P18 Ohgame, Y., & Hazeyama, A. (2006). Design and implementation of a *software* inspection support system for UML diagrams. IEICE TRANSACTIONS on Information and Systems, 89(4), 1327-1336.

P19 Rocha, A. C. O., Ramalho, F., & Machado, P. D. (2015). Automating test-based inspection of design models. *Software* Quality Journal, 23(1), 3-28.

P20 Sabaliauskaite, G., Kusumoto, S., & Inoue, K. (2004). Assessing defect detection performance of interacting teams in object-oriented design inspection. Information and *Software* Technology, 46(13), 875-886.

P21 Sabaliauskaite, G., Kusumoto, S., & Inoue, K. (2004). Comparing reading techniques for Object-Oriented design inspection. IEICE TRANSACTIONS on Information and Systems, 87(4), 976-984.

P22 Sabaliauskaite, G., Matsukawa, F., Kusumoto, S., & Inoue, K. (2002, October). An experimental comparison of checklist-based reading and perspective-based reading for UML design document inspection. In null (p. 148). IEEE.

P23 Sabaliauskaite, G., Matsukawa, F., Kusumoto, S., & Inoue, K. (2003). Further investigations of reading techniques for object-oriented design inspection. Information and *Software* Technology, 45(9), 571-585.

P24 Staron, M., Kuzniarz, L., & Thurn, C. (2005). An empirical assessment of using stereotypes to improve reading techniques in *software* inspections. ACM SIGSOFT *Software* Engineering Notes, 30(4), 1-7.

P25 Thelin, T., Runeson, P., & Wohlin, C. (2003). Prioritized use cases as a vehicle for *software* inspections. IEEE *software*, 20(4), 30-33.

P26 Travassos, G., Shull, F., Fredericks, M., & Basili, V. R. (1999, October). Detecting defects in object-oriented designs: using reading techniques to increase *software* quality. In ACM Sigplan Notices (Vol. 34, No. 10, pp. 47-56). ACM.

P27 T Geraldi, R., Oliveira Jr, E., Conte, T., & Steinmacher, I. (2015, April). Checklist-based Inspection of SMarty Variability Models. In Proceedings of the 17th International Conference on Enterprise Information Systems-Volume 2 (pp. 268-276). SCITEPRESS-Science and Technology Publications, Lda.

P28 Anda, B., & Sjøberg, D. I. (2002, July). Towards an inspection technique for use case models. In Proceedings of the 14th international conference on *Software* engineering and knowledge engineering (pp. 127-134). ACM.

P29 Kuzniarz, L., Staron, M., & Wohlin, C. (2004, June). An empirical study on using stereotypes to improve understanding of UML models. In Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on (pp. 14-23). IEEE.

P30 Alshazly, A. A., Elfatatry, A. M., & Abougabal, M. S. (2014). Detecting defects in *software* requirements specification. Alexandria Engineering Journal, 53(3), 513-527.

P31 Cruz-Lemus, J. A., Genero, M., Caivano, D., Abrahão, S., Insfrán, E., & Carsí, J. A. (2011). Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments. Information and *Software* Technology, 53(12), 1391-1403.

P32 Laitenberger, O., El Emam, K., & Harbich, T. G. (2001). An internally replicated quasi-experimental comparison of checklist and perspective based reading of code documents. IEEE Transactions on *Software* Engineering, 27(5), 387-421.

P33 Travassos, G. H. (2014, September). *Software* Defects: Stay Away from Them. Do Inspections!. In Quality of Information and Communications Technology (QUATIC), 2014 9th International Conference on the (pp. 1-7). IEEE.

APÊNDICE B – MATERIAIS UTILIZADOS NO EXPERIMENTO

Todos os formulários descritos no capítulo 4 estão presentes nesse apêndice. Ressaltando que o idioma apresentado aos participantes foi preservado nesse apêndice.

B.1 DESCRIÇÃO DA TAREFA (ABORDAGEM)

Junto com esta descrição da tarefa, você recebeu os seguintes documentos:

- Uma especificação funcional (requisitos) correta de um sistema a ser desenvolvido, contendo:
  - Um Escopo de um Módulo Real;
  - Uma lista de Requisitos Funcionais;
  - Um Diagrama de Casos de Uso;
  - As Descrições dos Casos de Uso.
- Uma Lista de Elementos do Modelo Esperados.
- Um Diagrama de Classes que contém defeitos.
- Um Formulário de Notificação de Defeitos.

É solicitado que:

- Inspecione o projeto de *software*, da seguinte maneira:
  1. **Leia o escopo.**
  2. **Leia os requisitos funcionais.**
  3. **Leia a descrição de um caso de uso.**
  4. **Leia as classes da lista de elementos (EMEs) do caso de uso.**
     a. Verifique se há defeitos no diagrama de classes;
     b. Caso encontre algum defeito, registre o defeito no "Formulário de Notificação de Defeitos" informando seu local, tipo e descrição.
  5. **Leia os atributos da lista de elementos (EMEs) do caso de uso.**
     a. Verifique se há defeitos no diagrama de classes;
     b. Caso encontre algum defeito, registre o defeito no "Formulário de Notificação de Defeitos" informando seu local, tipo e descrição.
  6. **Leia os relacionamentos da lista de elementos (EMEs) do caso de uso.**
     a. Verifique se há defeitos no diagrama de classes;
     b. Caso encontre algum defeito, registre o defeito no "Formulário de Notificação de Defeitos" informando seu local, tipo e descrição.
  7. **Continue no passo 3** (*até que o tempo seja excedido ou todos os elementos de cada caso de uso tenham sido cobertos*).

TIPOS DE DEFEITOS
Omissão: Algum item importante não foi incluído.
Ambiguidade: Um item pode ter várias interpretações.
Fato Incorreto: Um item retrata algo que não é verdadeiro.
Informação Estranha: Um item não é necessário.
Outros: Outros defeitos não definidos acima.

Ao fazê-lo, considere:

- A inspeção é um trabalho individual e discussões sobre a inspeção não são permitidas.

Muito obrigado pela sua participação!

B.2 DESCRIÇÃO DA TAREFA (*AD-HOC*)

Junto com esta descrição da tarefa, você recebeu os seguintes documentos:

- Uma especificação funcional (requisitos) correta de um sistema a ser desenvolvido, contendo:
  - Um Escopo de um Módulo Real;
  - Uma lista de Requisitos Funcionais;
  - Um Diagrama de Casos de Uso;
  - As Descrições dos Casos de Uso.
- Um Diagrama de Classes que contém defeitos.
- Um Formulário de Notificação de Defeitos.

É solicitado que:

- Inspecione o projeto de *software*, da seguinte maneira:

8. Leia o escopo.

9. Leia os requisitos funcionais.

10. Leia a descrição de um caso de uso.

11. Verifique se há defeitos no diagrama de classes enquanto lê a descrição.

c. Caso encontre algum defeito, registre o defeito no "Formulário de Notificação de Defeitos" informando seu local, tipo e descrição.

12. Continue no passo 3 (até que o tempo seja excedido ou todos os casos de uso tenham sido cobertos).

TIPOS DE DEFEITOS
Omissão: Algum item importante não foi incluído.
Ambiguidade: Um item pode ter várias interpretações.
Fato Incorreto: Um item descreve algo que não é verdadeiro.
Informação Estranha: Um item não é necessário.
Outros: Outros defeitos não definidos acima.

Ao fazê-lo, considere:

- **A inspeção é um trabalho individual e discussões sobre a inspeção não são permitidas.**

Muito obrigado pela sua participação!

B.3 FORMULÁRIO DE CARACTERIZAÇÃO

Nome: _____ Nível (B.Sc/Ms.c/D.Sc.): _____

1. Qual é sua experiência com desenvolvimento de *software*? (É possível selecionar mais de uma opção. Especifique, ao lado da opção escolhida, quanto tempo durou a experiência)

- ☐ Eu nunca desenvolvi *software*.
- ☐ Eu tenho desenvolvido *software* para uso próprio. ____
- ☐ Eu tenho desenvolvido *software* como parte de uma equipe, relacionado a um curso. ____
- ☐ Eu tenho desenvolvido *software* como parte de uma equipe, na indústria. ____

2. Por favor, indique o grau de sua experiência nesta seção seguindo a escala de 5 pontos abaixo (Olhe para o subtítulo):

Subtítulo:

1 = nenhum

2 = estudei em aula ou em livro

3 = pratiquei em 1 projeto em sala de aula

4 = usei em 1 projeto na indústria

5 = usei em vários projetos na indústria

Experiência em Projeto					
Experiência geral com análise e projeto orientados a objetos	1	2	3	4	5
Experiência na concepção de *software* baseado em requisitos	1	2	3	4	5
Experiência na concepção de diagramas de casos de uso	1	2	3	4	5
Experiência na elaboração de diagrama de classes	1	2	3	4	5
Experiência com princípios/padrões de design (GRASP, GoF)	1	2	3	4	5
Experiência geral com inspeções de *software*	1	2	3	4	5
Experiência inspecionando casos de uso	1	2	3	4	5
Experiência inspecionando diagramas de classes	1	2	3	4	5
Experiência inspecionando outros diagramas UML	1	2	3	4	5
Experiência com técnicas de inspeção de *software* (OORTs, PBR, UBR)	1	2	3	4	5

B.4 FORMULÁRIO DE NOTIFICAÇÃO DE DEFEITOS

Nome: _____

Horário de início da inspeção: _____

Horário de término da inspeção: _____

Lembre-se também de registrar a hora de início e término de qualquer pausa que você possa ter tomado durante o exercício.

Lista de Discrepâncias

TIPOS DE DEFEITOS

Ambiguidade (A) Fato Incorreto (FI) Informação Estranha (IE) Omissão (O) Outros (R)

ID	Localização	Tipo	Descrição

B.5 QUESTIONÁRIO DE ACOMPANHAMENTO (ABORDAGEM)

Nome: _____

1. Considerou o tempo suficiente para concluir a sua tarefa (Sim / Não): _____

Em caso negativo, explique a sua resposta: _____

_____

2. Gostaria de receber algumas orientações adicionais para conduzir a inspeção de projeto (Sim / Não)? _____

Em caso positivo, que tipo de orientação (opcional)? _____

_____

3. Em relação à complexidade da tarefa realizada:

Eu considerei a tarefa muito complexa.

Eu considerei a tarefa complexa.

Eu considerei a tarefa simples.

Eu considerei a tarefa muito simples.

4. Que dificuldades você encontrou para realizar esta tarefa?

_____

5. Em sua opinião, o que poderia tornar a realização desta tarefa mais agradável?

_____

6. Em relação à sua percepção da utilidade da abordagem:

	Discordo	Discordo Parcialmente	Neutro	Concordo Parcialmente	Concordo
Usar a abordagem (EMEs) melhoraria meu desempenho (detectar defeitos mais rápido) em realizar o trabalho de inspeção.					
Usar a abordagem (EMEs) aumentaria minha eficácia (detectar mais defeitos) em realizar o trabalho de inspeção.					
A abordagem (EMEs) seria útil para o trabalho de inspeção.					

7. Em relação à sua percepção sobre a facilidade de utilização:

	Discordo	Discordo Parcialmente	Neutro	Concordo Parcialmente	Concordo

Eu acho que a abordagem (EMEs) é fácil de usar.					

8. Em relação à sua intenção de adoção:

	Discordo	**Discordo Parcialmente**	**Neutro**	**Concordo Parcialmente**	**Concordo**
Utilizaria a abordagem (EMEs) regularmente para realizar trabalhos de inspeção.					

9. Use o espaço a seguir para comentários gerais que julgar necessários sobre a abordagem de revisão.

_____

B.6 QUESTIONÁRIO DE ACOMPANHAMENTO (*AD-HOC*)

Nome: _____

1. Considerou o tempo suficiente para concluir a sua tarefa (Sim / Não): _____

Em caso negativo, explique a sua resposta: _____

_____

2. Gostaria de receber algumas orientações adicionais para conduzir a inspeção de projeto (Sim / Não)? _____

Em caso positivo, que tipo de orientação (opcional)? _____

_____

3. Em relação à complexidade da tarefa realizada:

Eu considerei a tarefa muito complexa.

Eu considerei a tarefa complexa.

Eu considerei a tarefa simples.

Eu considerei a tarefa muito simples.

4. Que dificuldades você encontrou para realizar esta tarefa?

_____

5. Em sua opinião, o que poderia tornar a realização desta tarefa mais agradável?

_____

APÊNDICE C – DOCUMENTAÇÃO DE REQUISITOS E DIAGRAMAS UTILIZADOS NO EXPERIMENTO

Os subconjuntos da documentação real de requisitos e diagramas apresentados aos participantes na fase de execução do experimento (ver Subseção 5.2.7), além das listas de defeitos semeados e das listas de EMEs estão apresentados nesse apêndice.

C.1 SUBCONJUNTO DA DOCUMENTAÇÃO REAL DE REQUISITOS - Módulo Administração (MADM)

C.1.1 Escopo do Módulo

O Módulo de Administração do Sistema Gerencial Integrado objetiva permitir que informações básicas de cadastro (normalmente utilizada nos demais módulos do sistema) sejam mantidas (consulta, inclusão, alteração e exclusão). O *software* deve permitir que os clientes, os navios, as áreas de atuação, as atividades, os centros de custo, os impostos, as embarcações das empresas e afretadas, as informações das empresas, os locais, os estados e os tipos de documentos fiscais utilizados sejam mantidos no sistema. Esta manutenção envolve a consulta, inclusão, exclusão e alteração. Deve permitir, ainda, que o percentual de cálculo do AFRMM seja mantido no sistema. Esta manutenção envolve a consulta, inclusão, exclusão e alteração. Além disso, o *software* deve ser capaz de manter um histórico dos percentuais utilizados para que a informação associada ao faturamento (o percentual vigente na época do faturamento) não seja perdida. Destaca-se, por fim, que o módulo de administração não possui funcionalidades de importação/exportação de dados provenientes de sistemas externos.

C.1.2 Requisitos Funcionais (Recorte Real)

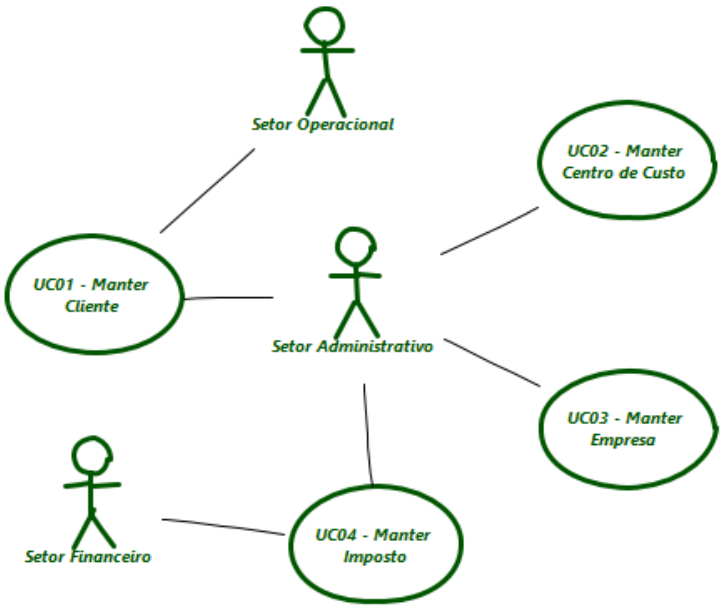
RF1: O *software* deve permitir que o Setor Operacional e o Setor Administrativo efetuem a manutenção de clientes (consulta, inclusão, alteração e exclusão).

RF2: O *software* deve permitir que o Setor Administrativo efetue a manutenção de centros de custo (consulta, inclusão e alteração).

RF3: O *software* deve permitir que o Setor Administrativo efetue a manutenção de empresas (consulta, inclusão, alteração e exclusão).

RF4: O *software* deve permitir que o Setor Administrativo e o Setor Financeiro efetuem a manutenção de impostos (consulta, inclusão, alteração e exclusão).

C.1.3 Diagrama de Casos de Uso (Recorte Real)



C.1.4 Descrição De Alguns Casos De Uso (Recorte Real)

UC01 – Manter Cliente

Objetivo:	Permitir que clientes sejam mantidos (consulta, inclusão, alteração e exclusão) no sistema.
Requisitos:	RF01
Atores:	Setor Operacional, Setor Administrativo
Trigger:	O ator seleciona a opção Manter Cliente.
Fluxo Principal:	O sistema exibe uma tela de filtro com as seguintes informações: - Sigla ou - Cliente; - CNPJ; - Estado. Opções: - Buscar; - Adicionar Novo Cliente. - Voltar O ator preenche os filtros e seleciona a opção Buscar [RN1][A1] [A2] O sistema exibe todos os clientes retornados após o filtro com as seguintes informações (somente leitura): - Sigla; - Cliente; - CNPJ; - Endereço; - Bairro; - Município; - Estado; - CEP; - Inscrição Municipal; - Inscrição Estadual; - Órgão Público Federal (opções: Sim e Não, default: Não). - A opção Editar Cliente - A opção Excluir Cliente O sistema apresenta ao final as opções: - Adicionar Novo Cliente - Voltar O ator seleciona a opção Editar Cliente [A2][A3][A4] O sistema exibe uma tela de edição de cliente com as seguintes informações: - Sigla; - Cliente; - CNPJ; - Endereço; - Bairro; - Município; - Estado; - CEP; - Inscrição Municipal; - Inscrição Estadual; - Órgão Público Federal (opções: Sim e Não, default: Não) - Tabela de Preços, contendo * DWT Inicial; * DWT Final; * Preço; * Tipo de Preço (Por Navio ou por Rebocador do Serviço) [RN4] O sistema apresenta ao final as opções: - Salvar - Voltar O ator preenche as informações e seleciona a opção Salvar [A5] O sistema valida os dados [RN2] [RN3] O sistema apresenta tela de confirmação de edição de cliente O caso de uso é encerrado.

Fluxo Alternativo [A1] O ator escolhe a opção Voltar
O sistema retorna para a tela inicial.
O caso de uso é encerrado
[A2] O ator escolhe a opção Adicionar Novo Cliente
O sistema exibe uma tela de cadastro de novo cliente com as seguintes informações:
- Sigla; - Cliente; - CNPJ; - Endereço; - Bairro; - Município; - Estado; - CEP;
- Inscrição Municipal; - Inscrição Estadual; - Órgão Público Federal (opções: Sim e Não, default: Não)
- Tabela de Preços, contendo
  * DWT Inicial; * DWT Final; * Preço; * Tipo de Preço (Por Navio ou por Rebocador do Serviço)
O sistema apresenta ao final as opções:
- Salvar
- Voltar
O ator preenche as informações e seleciona a opção Salvar [A6]
O sistema valida os dados [RN2] [RN3]
O sistema apresenta tela de confirmação de cadastro de cliente
O sistema retorna para a tela inicial.
O caso de uso é encerrado.
[A3] O ator escolhe a opção Voltar
O sistema retorna ao passo 1 do fluxo principal
[A4] O ator escolhe a opção Excluir Cliente
O sistema apresenta mensagem de confirmação de exclusão de cliente.
O ator confirma a exclusão.
O sistema exclui o cliente.
O caso de uso é encerrado.
[A5] O ator escolhe a opção Voltar
O sistema retorna ao passo 3 do fluxo principal

Regras de negócio: [RN1] Todos os campos de filtro são opcionais.
[RN2] As informações a seguir são obrigatórias: Sigla textual, Cliente, CNPJ, Endereço, Bairro, Município, Estado, CEP.
[RN3] É obrigatório o preenchimento de um dos dois campos: Inscrição Municipal ou Inscrição Estadual.
[RN4] Se for por Navio deve dividir entre os rebocadores envolvidos no serviço. Se for por Rebocador este é o preço para ser considerado para cada um dos rebocadores. Só pode ter um registro, ou é por Navio ou por Rebocador.

UC02 – Manter Centro de Custo

Objetivo: Permitir que centros de custo sejam mantidos (consulta, inclusão, alteração e exclusão) no sistema.

Requisitos: RF02

Atores: Setor Administrativo

Trigger: O ator seleciona a opção Manter Centro de Custo.

Fluxo Principal: O sistema exibe uma tela de filtro com as seguintes informações:
- Empresa [RN3]; - Centro de Custo;
- Data de Inclusão
  - De
  - Até
- Rateio (contendo as opções Sim e Não).
- Opções:
  - Buscar
  - Adicionar Novo Centro de Custo
  - Voltar
O usuário preenche os filtros e seleciona a opção Buscar [RN1][A1] [A2]
O sistema exibe todos os centros de custo retornados após o filtro com as seguintes informações (somente leitura):
- Empresa; - Centro de Custo; - Data de Inclusão; - Rateio.
- A opção Editar Centro de Custo
O sistema apresenta ao final as opções:
- Adicionar Novo Centro de Custo
- Voltar
O ator seleciona a opção Editar Centro de Custo [A2][A3]
O sistema exibe uma tela de edição de Centro de Custo com as seguintes informações:
- Empresa (somente leitura); - Centro de Custo (somente leitura); - Data de Inclusão (somente leitura).
(Rateios Definidos)

- Tabela com os rateios já definidos para o centro de custo que está sendo editado. Cada rateio possui as seguintes informações: Rateio (somente leitura) e Data e a opção Excluir Rateio [A5]
(Novo Rateio)
- Rateio; - Data.
- A opção Adicionar Rateio [RN3] [A6]
O sistema apresenta ao final as opções:
- Voltar
O ator preenche as informações e adiciona os rateios desejados [A4]
O sistema valida os dados [RN2]
O caso de uso é encerrado.

Fluxo Alternativo

[A1] O ator escolhe a opção Voltar
O sistema retorna para a tela inicial.
O caso de uso é encerrado
[A2] O ator escolhe a opção Adicionar Novo Centro de Custo
O sistema exibe uma tela de cadastro de novo Centro de Custo com as seguintes informações:
- Empresa; - Centro de Custo; - Data de Inclusão; - Rateio.
(Rateios Definidos)
- Tabela com os rateios já definidos para o centro de custo que está sendo editado. Cada rateio possui as seguintes informações: Rateio, Data e a opção Excluir Rateio [A6]
(Novo Rateio)
- Rateio; - Data.
- A opção Adicionar Rateio [A7]
O sistema apresenta ao final as opções: - Salvar e - Voltar
O ator preenche as informações e seleciona a opção Salvar [A5]
O sistema valida os dados [RN2]
O caso de uso é encerrado.
[A3] O ator escolhe a opção Voltar
O sistema retorna ao passo 1 do fluxo principal
[A4] O ator escolhe a opção Voltar
O sistema retorna ao passo 3 do fluxo principal
[A5] O ator seleciona a opção Excluir Rateio
O sistema solicita confirmação da exclusão com as opções confirmar e cancelar.
O ator confirma a exclusão
O sistema exclui o rateio da tabela de rateios definidos
O sistema retorna ao passo 5 do fluxo principal ou ao passo 1 do fluxo alternativo [A2]
[A6] O ator seleciona a opção Adicionar Rateio
O sistema adiciona o rateio à tabela de rateios definidos
O sistema retorna ao passo 5 do fluxo principal ou ao passo 1 do fluxo alternativo [A2]

Regras de negócio:

[RN1] Todos os campos de filtro são opcionais.
[RN2] Todas as informações são obrigatórias.
[RN3] O usuário que tiver acesso a somente uma empresa só poderá consultar os centros de custo dessa empresa.

UC03 – Manter Empresa

Objetivo: Permitir que empresas sejam mantidas (consulta, inclusão, alteração e exclusão) no sistema.

Requisitos: RF03

Atores: Setor Administrativo

Trigger: O ator seleciona a opção Manter Empresa.

Fluxo Principal:

O sistema exibe as empresas cadastradas no sistema com as seguintes informações:
- Sigla; - Nome; - Logotipo (contendo um link para abrir a imagem cadastrada em uma nova janela); - CNPJ; - Endereço; - Bairro;
- Município; - Estado; -CEP; - Inscrição Municipal; - Inscrição Estadual;
- Contas para Depósito (Banco, Agência e Conta).
- A opção Editar Empresa
O sistema apresenta ao final as opções:
  - Adicionar Nova Empresa
  - Voltar
O ator seleciona a opção Editar Empresa [A1][A2]
O sistema exibe uma tela de edição de Empresa com as seguintes informações:
- Sigla; - Nome; - Logotipo (contendo ao lado a opção Enviar Logotipo) [A4]; - CNPJ; - Endereço; - Bairro; - Município; - Estado; - CEP;
- Inscrição Municipal; - Inscrição Estadual; - Contas para Depósito (Banco, Agencia e Conta)
O sistema apresenta ao final as opções:

- Salvar
- Voltar
O ator preenche as informações e seleciona a opção Salvar [A3]
O sistema valida os dados [RN1] [RN2]
O sistema apresenta tela de confirmação de edição da Empresa
O sistema retorna para a tela inicial.
O caso de uso é encerrado.

Fluxo Alternativo [A1] O ator escolhe a opção Voltar
O sistema retorna para a tela inicial.
O caso de uso é encerrado
[A2] O ator escolhe a opção Adicionar Nova Empresa
O sistema exibe uma tela de cadastro de nova empresa com as seguintes informações:
- Sigla; - Nome; - Logotipo (contendo ao lado a opção Enviar Logotipo) [A4]; - CNPJ; - Endereço; - Bairro; - Município; - Estado; - CEP;
- Inscrição Municipal; - Inscrição Estadual; - Contas para Depósito (Banco, Agencia e Conta).
O sistema apresenta ao final as opções:
- Salvar
- Voltar
O ator preenche as informações e seleciona a opção Salvar [A3]
O sistema valida os dados [RN1] [RN2]
O sistema apresenta tela de confirmação de cadastro de Empresa
O caso de uso é encerrado.
[A3] O ator escolhe a opção Voltar
O sistema retorna ao passo 1 do fluxo principal
[A4] O ator escolhe a opção Enviar Logotipo
O sistema apresenta tela de envio de logotipo contendo as informações:
- Arquivo (contendo ao lado a opção de seleção de arquivo padrão do sistema operacional)
- As opções: * Enviar; * Voltar
O ator informa o arquivo e seleciona a opção Enviar. [A5]
O sistema armazena a imagem enviada.
O sistema retorna ao passo 1 do fluxo alternativo [A2] ou ao passo 3 do fluxo principal com o campo Logotipo definido com o nome da imagem.
[A5] O ator escolhe a opção Voltar
O sistema retorna ao passo 1 do fluxo alternativo [A2] ou ao passo 3 do fluxo principal.

Regras de negócio: [RN1] As informações a seguir são obrigatórias: Sigla, Nome, CNPJ, Endereço, Bairro, Município, Estado, CEP.
[RN2] É obrigatório o preenchimento de um dos dois campos: Inscrição Municipal ou Inscrição Estadual.

UC04 – Manter Imposto

Objetivo: Permitir que os impostos sejam mantidos (consulta, inclusão, alteração e exclusão) no sistema.

Requisitos: RF04

Atores: Setor Administrativo e Setor Financeiro

Trigger: O ator seleciona a opção Manter Imposto.

Fluxo Principal: O sistema exibe os impostos cadastrados no sistema com as seguintes informações:
- Imposto; - Tipo de Imposto (Federal, Estadual ou Municipal);
- Alíquota; - Estado; - Município; - Status (Ativo ou Inativo).
- A opção Editar Imposto
- A opção Excluir Imposto
O sistema apresenta ao final as opções:
- Adicionar Novo Imposto
- Voltar
O ator seleciona a opção Editar Imposto [A1][A2][A3]
O sistema exibe uma tela de edição de Imposto com as seguintes informações:
- Imposto; - Tipo de Imposto (Federal, Estadual ou Municipal);
- Alíquota; - Estado; - Município; - Status (Ativo ou Inativo).
O sistema apresenta ao final as opções:
- Salvar
- Voltar
O ator preenche as informações e seleciona a opção Salvar [A4]
O sistema valida os dados [RN1]
O caso de uso é encerrado.

Fluxo Alternativo	[A1] O ator escolhe a opção Voltar O sistema retorna para a tela inicial. O caso de uso é encerrado [A2] O ator escolhe a opção Adicionar Novo Imposto O sistema exibe uma tela de cadastro de novo Imposto com as seguintes informações: - Imposto; - Tipo de Imposto (Federal, Estadual ou Municipal); - Alíquota; - Estado; - Município; - Status (Ativo ou Inativo); O sistema apresenta ao final as opções: - Salvar; - Voltar. O ator preenche as informações e seleciona a opção Salvar [A5] O sistema valida os dados [RN2] O sistema apresenta tela de confirmação de cadastro de Imposto O sistema retorna para a tela inicial. O caso de uso é encerrado. [A3] O ator escolhe a opção Excluir Imposto O sistema apresenta mensagem de confirmação de exclusão de Imposto. O ator confirma a exclusão. O sistema exclui o imposto. O caso de uso é encerrado. [A4] O ator escolhe a opção Voltar O sistema retorna ao passo 1 do fluxo principal
Regras de negócio:	[RN1] Todas as informações são obrigatórias.

Regras de negócio: [RN1] Todas as informações são obrigatórias.

C.1.5 Defeitos Semeados no Diagrama de Classes

Omissão

1. A classe Preço ausente no diagrama. (Fácil)
2. Atributo OrgaoPublico ausente na classe Cliente. (Fácil)
3. Relacionamento ausente entre Empresa e ContaParaDeposito. (Difícil)
4. Relacionamento ausente entre Organização e Estado.(Médio)
5. Atributo descricao ausente na classe CentroDeCusto. (Fácil)
6. Atributo logotipo ausente na classe Empresa. (Fácil)
7. Classe Enumeration TipoImposto deveria ter valor FEDERAL. (Fácil)

Fato Incorreto

1. Navegação invertida de CentroDeCusto para Rateio. (Médio)
2. Imposto não deve estar relacionado com Empresa. (Médio)
3. Cliente é um tipo de Organização, logo não é uma associação simples. (Difícil)
4. Navegação de generalização invertida de Organização para Empresa (Médio)
5. Relacionamento inconsistente de agregação entre Imposto e Aliquota. (Difícil)
6. Atributos da Classe Organização deveriam ser protegidos. (Difícil)
7. Relacionamento (multiplicidade) inconsistente entre CentrodeCusto e Rateio. Pode não haver rateio. (Difícil)

Ambiguidade

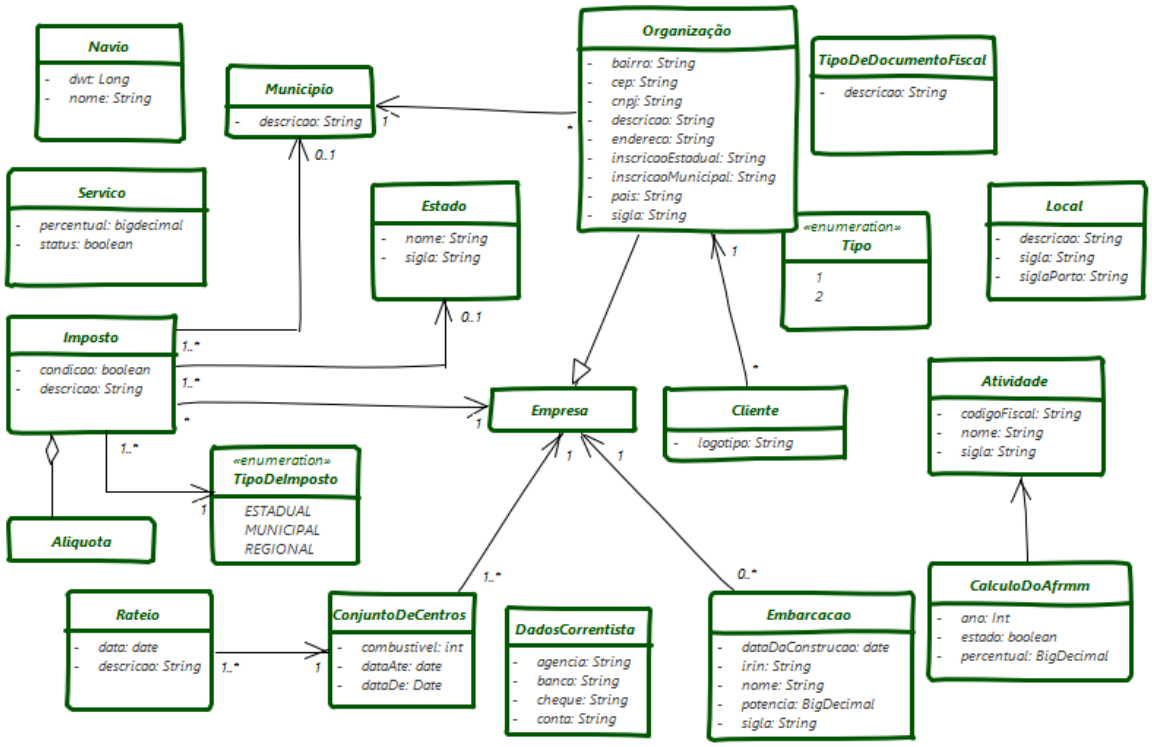
1. Atributo condicao definido na classe Imposto, enquanto que o correto seria status. (Difícil)
2. DadosCorrentista no documento de requisitos é referenciado como ContaParaDeposito. (Difícil)
3. Classe ConjuntodeCentros é refenciada no documento como CentroDeCusto (Fácil)
4. Classe Tipo é refenciada no documento como TipoDePreco (Fácil)
5. Atributo TipoPreco definido na classe Enumeration como 1 e 2, enquanto que no Cenário é informado (Por Navio ou por Rebocador do Serviço). (Difícil)
6. Atributo descricao definido na classe Organização, enquanto que o correto seria nome. (Difícil)
7. Atributo dataDe na classe CentroDeCusto, enquanto que o correto seria DataDeInclusao. (Fácil)

Informação Estranha

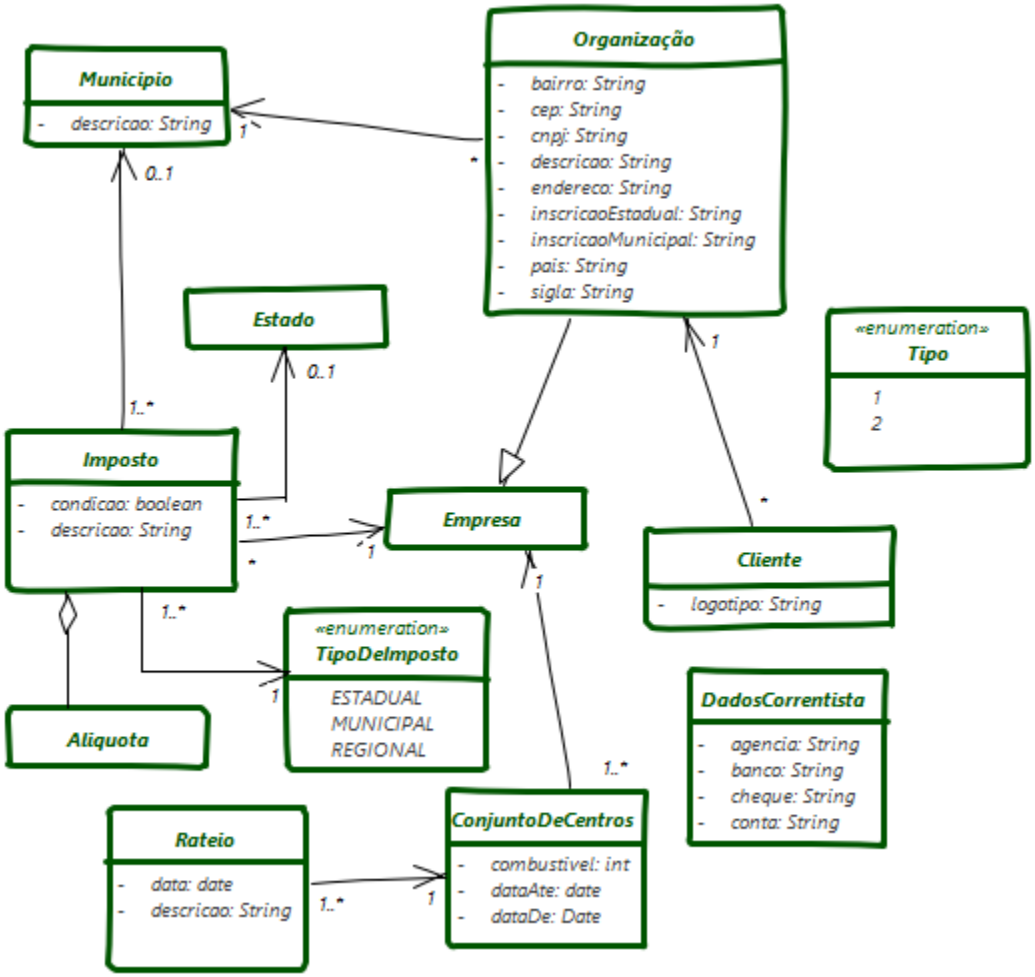
1. Classe ContaParaDeposito não tem o atributo cheque. (Médio)
2. Classe ConjuntoDeCentros não tem o atributo combustivel. (Médio)
3. Classe Cliente não tem o atributo logotipo. (Fácil)
4. Classe Municipio não tem o atributo descricao. (Médio)
5. Classe Organização não tem o atributo país. (Médio)
6. Classe CentroDeCusto não tem atributo DataAte. (Fácil)
7. Classe Enumeration TipoImposto não tem valor REGIONAL. (Fácil)

C.1.6 Diagramas

C.1.6.1 Diagrama de Classes Sem Definição de Escopo



C.1.6.2 Diagrama de Classes Com Definição de Escopo



C.1.7 Elementos do Modelo Esperados (EMEs)

UC01 - Manter Cliente

Classes Esperadas

1.Cliente
2.Estado
3.Municipio
4.Preco
5.TipoDePreco

Relacionamentos Esperados

1.Cliente ► Municipio
2.Cliente ► Estado
3.Cliente ► Preco
4.Preco ► TipoDePreco

c. Atributos Esperados

1.Cliente.bairro
2.Cliente.cep
3.Cliente.cliente
4.Cliente.cnpj
5.Cliente.endereco
6.Cliente.estado
7.Cliente.inscricaoEstadual
8.Cliente.inscricaoMunicipal
9.Cliente.município
10.Cliente.orgaoPublicoFederal
11.Cliente.preco
12.Cliente.sigla
13.Preco.dwtFinal
14.Preco.dwtInicial
15.Preco.preco
16.Preco.tipoDePreco

UC02 - Manter Centro de Custo

Classes Esperadas

1.CentroDeCusto
2.Empresa
3.Rateio

Relacionamentos Esperados

1.CentroDeCusto ► Empresa
2.CentroDeCusto ► Rateio

Atributos Esperados

1.CentroDeCusto.descricao
2.CentroDeCusto.dataAte
3.CentroDeCusto.dataDe
4.CentroDeCusto.empresa
5.CentroDeCusto.rateio
6.Rateio.data
7.Rateio.descricao

UC03 - Manter Empresa

Classes Esperadas

1.ContaParaDeposito
2.Empresa
3.Estado
4.Municipio

Relacionamentos Esperados

1.Empresa ► Municipio
2.Empresa ► Estado
3.Empresa ► ContaParaDeposito

Atributos Esperados

1.ContaParaDeposito.agencia
2.ContaParaDeposito.banco
3.ContaParaDeposito.conta
4.Empresa.bairro
5.Empresa.cep
6.Empresa.cnpj
7.Empresa.contaParaDeposito
8.Empresa.endereco
9.Empresa.estado
10.Empresa.inscricaoEstadual
11.Empresa.inscricaoMunicipal
12.Empresa.logotipo
13.Empresa.municipio
14.Empresa.nome
15.Empresa.sigla

UC04 - Manter Imposto

Classes Esperadas

1.Aliquota
2.Estado
3.Imposto
4.Municipio
5.TipoDeImposto

Relacionamentos Esperados

1.Imposto ► TipoDeImposto
2.Imposto ► Aliquota
3.Imposto ► Estado
4.Imposto ► Municipio

Atributos Esperados

1.Imposto.alíquota
2.Imposto.estado
3.Imposto.descricao
4.Imposto.municipio
5.Imposto.status
6.Imposto.tipoDeImposto

C.2 SUBCONJUNTO DA DOCUMENTAÇÃO REAL DE REQUISITOS - Módulo de Faturamento (MFAT)

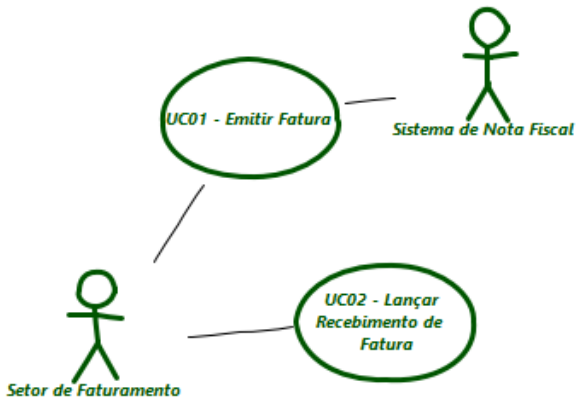
C.2.1 Escopo do Módulo

O Módulo de Faturamento do Sistema Gerencial da empresa objetiva controlar os processos relacionados ao faturamento dos serviços executados pelas embarcações. O *software* deve permitir que os serviços efetuados pelas embarcações sejam faturados via nota fiscal ou recibo. Esta funcionalidade será acessada a partir da tela de consulta de serviços. Ao faturar um serviço, o sistema deverá considerar os devidos impostos (federais, estaduais e/ou municipais) e a forma de pagamento correspondente (boleto bancário ou débito em conta). O *software* deve permitir que os pagamentos das faturas geradas sejam administrados. Este pagamento pode ser feito de forma total ou parcialmente. Neste último caso, o *software* deve permitir lançar parcelas referentes à fatura de forma que o total das parcelas lançadas seja igual ao valor da fatura

C.2.2 Requisitos Funcionais (Recorte Real)

RF1: O *software* deve permitir que o setor de faturamento emita nota fiscal referente aos serviços realizados pelas embarcações.
RF2: O *software* deve permitir que o setor de faturamento emita boleto bancário para o pagamento dos serviços das embarcações.
RF3: O *software* deve permitir que o setor de faturamento emita recibo referente aos serviços realizados pelas embarcações.
RF4: O *software* deve permitir que o setor de faturamento administre o pagamento das faturas emitidas total ou parcialmente.

C.2.3 Diagrama de Casos de Uso (Recorte Real)



C.2.4 Descrição De Alguns Casos De Uso (Recorte Real)

UC01 – Emitir Fatura

Objetivo:	Permitir que o Setor de Faturamento efetue o faturamento de serviços.
Requisitos:	RF1, RF2, RF3
Atores:	Setor de Faturamento
Trigger:	Ator seleciona a opção Emitir Fatura
Fluxo Principal:	1. O sistema apresenta tela para consulta de serviços com estado em aberto contendo as informações: - Número do Serviço ou - Embarcação; - Número da Viagem; - Data Inicial e Data Final; - Cliente; - Atividade.

- A opção Buscar Serviços

2. O ator informa os dados de busca e seleciona a opção Buscar Serviços **[RN1]**
3. O sistema apresenta a lista de serviços recuperados, ordenados pelo número do serviço, com as seguintes informações:

- Opção de Seleção; - Número do Serviço; - Data Inicial e Data Final; - Cliente; - Atividade; - Embarcações; - Valor.

4. O sistema apresenta ao final as opções:

- Definir Fatura; - Voltar

5. O ator seleciona um ou mais serviços a serem faturados e seleciona a opção Definir Fatura **[A1]**
6. O sistema verifica os serviços selecionados **[RN2] [A2]**
7. O sistema apresenta tela de fatura contendo as informações:

- Cliente

(Serviços da Fatura) (somente leitura)

- Lista de serviços selecionados contendo para cada um:
  * Número do Serviço; * Data Inicial e Data Final; * Atividade; * Embarcações; * Valor
- Valor Bruto **[RN3]**

(Detalhes da Fatura) (campos editáveis)

- Número do Contrato **[RN6];** - Número do Relatório de Medição **[RN6];** - Dias Faturados **[RN6];** - Motivo dos descontos (Downtime, Multa, Combustível) **[RN6];** - Observação sobre os descontos **[RN6];** - Tipo de Documento Fiscal (contendo as opções: Nota Fiscal ISS, Nota Fiscal ICMS, Recibo);
- Descrição (campo obrigatório);
- Valor do Desconto; - Carta de Crédito.
- Retenções Federais, contendo para cada imposto **[RN4]**:
  * Nome do Imposto; * Alíquota; * Valor do Imposto
- Imposto Estadual (ICMS) contendo **[RN4]**:
  * Opção de Seleção (caixa de seleção - radio)
  * Nome do Imposto (selecionado); * Alíquota; * Valor do Imposto**;** * Opção Substituição Tributária (caixa de seleção)
- Imposto Municipal (ISS) contendo **[RN4]**:
  * Opção de Seleção (caixa de seleção - radio)
  * Nome do Imposto; * Alíquota; * Valor do Imposto **;**
  * Município (preenchido por padrão com Rio de Janeiro)
  * Opção Retido pelo Cliente (caixa de seleção)
- Valor Líquido **;** - Valor da Fatura **[RN7]**

(Dados do Pagamento)

- Lista de Parcelas cadastradas contendo para cada parcela:
  * Número da Parcela (somente leitura – número sequencial)
  * Tipo de Pagamento (somente leitura)
  * Número do Boleto (somente leitura) **[RN5]**
  * Data de Vencimento (somente leitura)
  * Valor da Parcela (somente leitura)
  * Opção Excluir Parcela **[A3]**
- Nova Parcela
  * Data de Vencimento (campo obrigatório)
  * Valor da Parcela (campo obrigatório)
  * Tipo de Pagamento (contendo as opções: Boleto Bancário, Depósito em Conta) (campo obrigatório)
  * Tipo de Cobrança (contendo as opções: Nota Fiscal, Recibo)
  * Conta para Depósito; * Número do Boleto
  * Opção Incluir Parcela **[RN4] [A4]**

8. O sistema apresenta ao final as opções:

- Emitir Fatura; - Voltar

9. O ator informa os dados da fatura e seleciona a opção Emitir Fatura.
10. O sistema verifica os dados de pagamento da fatura
11. O sistema emite a fatura :

- O sistema altera o estado dos serviços para Faturado.
- O sistema define cada parcela criada para o estado Faturada.
- O sistema armazena os dados da fatura.
- O sistema define o estado da fatura como Emitida.

12. O sistema apresenta tela de confirmação contendo as informações:

(Sua fatura foi emitida com sucesso e contempla as seguintes informações)

- Cliente

(Serviços da Fatura)

- Lista de serviços selecionados contendo para cada um:
  * Número do Serviço; * Data Inicial e Data Final; * Atividade * Embarcações; * Valor
- Valor Bruto

(Detalhes da Fatura)

- Número do Contrato **[RN6];**
- Número do Relatório de Medição **[RN6]**
- Dias Faturados **[RN6]**
- Lista de motivos dos descontos (Downtime, Multa, Combustível) **[RN6]**
- Observação sobre os descontos **[RN6]**
- Tipo de Documento Fiscal (contendo as opções: Nota Fiscal ISS, Nota Fiscal ICMS, Recibo)
- Descrição (campo obrigatório)
- Valor do Desconto
- Carta de Crédito
- Data de Emissão (contendo a data atual)
- Retenções Federais, contendo para cada imposto:
  * Nome do Imposto; * Alíquota; * Valor do Imposto
- Imposto Estadual (ICMS) contendo:
  * Alíquota; * Valor do Imposto; * Retido pelo Cliente
- Imposto Municipal (ISS) contendo:
  * Município (preenchido por padrão com Rio de Janeiro)
  * Alíquota; * Valor do Imposto; * Retido pelo Cliente
- Valor Líquido; - Valor da Fatura

(Dados do Pagamento)

- Lista de Parcelas cadastradas contendo para cada parcela:
  * Número da Parcela
  * Tipo de Pagamento
  * Número do Boleto **[RN5]**
  * Data de Vencimento
  * Valor da Parcela [RN]

13. O sistema apresenta ao final a opção Gravar Fatura
14. O ator confirma.
15. O sistema grava os dados da fatura e retorna para a tela inicial.
16. O caso de uso é encerrado.

Fluxo Alternativo

[A1] O ator seleciona a opção Voltar

1. O sistema retorna ao passo 1 do fluxo principal.

[A2] Serviços de diferentes clientes selecionados

1. O apresenta uma mensagem indicando que uma fatura só poderá conter vários serviços caso estes sejam do mesmo cliente com a opção OK ao final.
2. O ator seleciona a opção ok.
3. O caso de uso retorna ao passo 3 do fluxo principal.

[A3] O ator seleciona a opção Excluir Parcela: O sistema retira a parcela selecionada da lista de parcelas incluídas.

[A4] O ator seleciona a opção Incluir Parcela: O sistema inclui a parcela definida na lista de parcelas incluídas considerando o menor número de parcela definido ao definir o número da parcela.

Regras de negócio:

[RN1] Os campos de busca não são obrigatórios.
[RN2] Só é possível emitir uma fatura com vários serviços caso os serviços sejam do mesmo cliente.
[RN3] O Valor Bruto da fatura, neste momento é calculado pela soma de todos os serviços selecionados para a fatura.
[RN4] Caso o tipo seja recibo, impostos estaduais e municipais NÃO podem ser selecionados.
[RN5] O campo Número do boleto é apresentado apenas se o tipo de pagamento for boleto bancário, caso contrário a Conta de Depósito deve ser selecionada.
[RN6] Caso o serviço seja de contrato.
[RN7] No Apoio Marítimo Contrato o valor total é calculado pelo Valor Base + Reajuste. Onde: Valor Base é o Número de Dias Faturados x Valor da Diária. Reajuste é % de reajuste x Valor da Diária X Número de Dias Faturados.

UC02 – Lançar Recebimento de Fatura

Objetivo: Permitir que o Setor de Faturamento confirme o recebimento do pagamento de uma fatura (recibo ou nota fiscal).

Requisitos: RF4

Atores: Setor de Faturamento

Trigger: O ator seleciona a opção Lançar Recebimento de Fatura.

Fluxo Principal:

1. O sistema apresenta tela de busca de faturas emitidas contendo as informações:

- Número da Fatura

OU

- Número da Nota Fiscal

OU

- Número do Recibo
OU
- Cliente (opções: todos + lista de clientes cadastrados)
- Valor (de até)
- Tipo de Cobrança (contendo as opções: Todos, Nota Fiscal, Recibo)
- Período de Emissão
  * De; * Até
- A opção Buscar

2. O ator informa os dados de busca e seleciona a opção Buscar **[RN1]**
3. O sistema apresenta tela de faturas que estão no estado "Emitida" ou "Pagamento Parcial", contendo as informações para cada fatura:

- Número; - Tipo de Cobrança; - Cliente; - Descrição; - Data de Emissão; - Parcela; - Valor; - Valor Pago **[RN2];** - Valor a Receber **[RN3];** - Data de Vencimento
- Opção Lançar Recebimento

4. O ator seleciona a opção Lançar Recebimento
5. O sistema apresenta um formulário para registro do recebimento da fatura com as seguintes informações:

(Dados da Fatura – somente leitura)
- Cliente
(Serviços da Fatura)
- Lista de serviços selecionados contendo para cada um:
  * Número do Serviço; * Data Inicial e Data Final; * Atividade * Embarcações; * Valor
- Valor Bruto
(Detalhes da Fatura)
- Tipo de Cobrança; - Número do Documento de Cobrança;
- Tipo de Pagamento; - Número do Boleto; - Valor do Desconto;
- Data de Emissão; - Data de Vencimento; - Valor Total Recebido **[RN7];**
- Impostos da Quitação **[RN10]**, contendo para cada imposto:
  * Nome do Imposto; * Alíquota; * Valor do Imposto
- Valor Líquido
- Bonus
- Valor da Fatura
(Dados dos Pagamentos – campos editáveis)
- Data do Pagamento **[RN8]**
- Valor do Pagamento (contendo inicialmente valor definido para a parcela)
- Opção para Adicionar na lista de pagamentos.
- Quitado (Sim ou Não) **[RN9]**
- Opção Confirmar Recebimento
- Opção Voltar

6. O ator informa os dados do pagamento e seleciona a opção Confirmar Recebimento **[A1]**
7. O sistema valida os dados **[RN4] [A2]**
8. O sistema solicita confirmação do recebimento
9. O ator confirma o recebimento **[A3]**
10. O sistema efetiva o pagamento da fatura:

- O sistema armazena os dados do pagamento.
- O sistema altera o estado da fatura para "Paga" ou "Parcialmente Paga". **[RN5] [RN6]**
- O sistema altera o estado da parcela para Paga.
- O sistema atualiza o Valor Total Recebido

11. O sistema retorna ao passo 2 do fluxo principal, atualizando a lista de faturas pendentes de confirmação de recebimento.

Fluxo Alternativo

[A1] Ator seleciona a opção Voltar

1. O sistema retorna ao passo 3 do fluxo principal.

[A2] Ator preenche dados inválidos

1. O sistema apresenta mensagem de dados inválidos, não registra os dados e continua no mesmo fluxo alternativo.

[A3] Ator seleciona a opção Cancelar

1. O sistema retorna ao passo 3 do fluxo principal.

Regras de negócio:

[RN1] Os campos de filtro de busca não são obrigatórios.
[RN2] O campo valor pago é dado pelo somatório dos valores pagos pelo cliente até o momento.
[RN3] O campo valor a receber é dado pela fórmula: Valor da Fatura -Valor Pago.
[RN4] Todos os campos do formulário são obrigatórios.
[RN5] Caso tenha sido marcada como quitada, então estado da fatura = paga.
[RN6] Caso não tenha sido marcada como quitada, então estado da fatura = parcialmente paga.
[RN7] Contém o valor total recebido para a fatura (ou parcela) até o momento.

[RN8] A data de recebimento deve ser menor ou igual à data atual e maior ou igual à data de emissão da fatura.
[RN9] Só deve permitir a quitação abaixo do valor da parcela para determinado perfil.
[RN10] Desvinculado dos impostos da fatura, os efetivamente utilizados na quitação também precisam ser cadastrados. Inicialmente correspondem aos da fatura, mas eles podem ser editados. Sempre que os impostos da quitação forem alterados o valor líquido e total da fatura deve ser recalculado.

C.2.5 Defeitos Semeados no Diagrama de Classes

Omissão

1. Atributo dataInicio está ausente na classe Servico. (Fácil)
2. Classe StatusFatura ausente no diagrama. (Difícil)
3. Atributo motivoCancelamento está ausente na classe Fatura/Documento. (Médio)
4. Relacionamento Fatura/Documento e MotivoDeconto ausente. (Difícil)
5. Atributo DataVencimento ausente na classe Parcela. (Fácil)
6. Classe Embarcacao ausente no diagrama. (Médio)
7. Em TipoMotivoDeducao só há 2 opções (Multa, Combustível), estando inconsistente com o Cenário, onde é informado também Downtime (Fácil)

Fato Incorreto

1. Navegação invertida de ParcelaFatura para StatusParcela. (Médio)
2. Relacionamento entre Fatura/Documento e Aliquota não existe. (Médio)
3. Relacionamento Documento/Fatura e TipoDeCobranca não existe, correto seria com Parcela.. (Médio)
4. Atributo relatoriosDeMedicao da classe Fatura/Documento não é uma Lista. (Difícil)
5. Navegação invertida de Fatura/Documento para Servico. (Médio)
6. Classe TipoDocumentoFiscal não é agregada a Documento. (Difícil)
7. Relacionamento entre Serviço e Cliente não existe. (Médio)

Ambiguidade

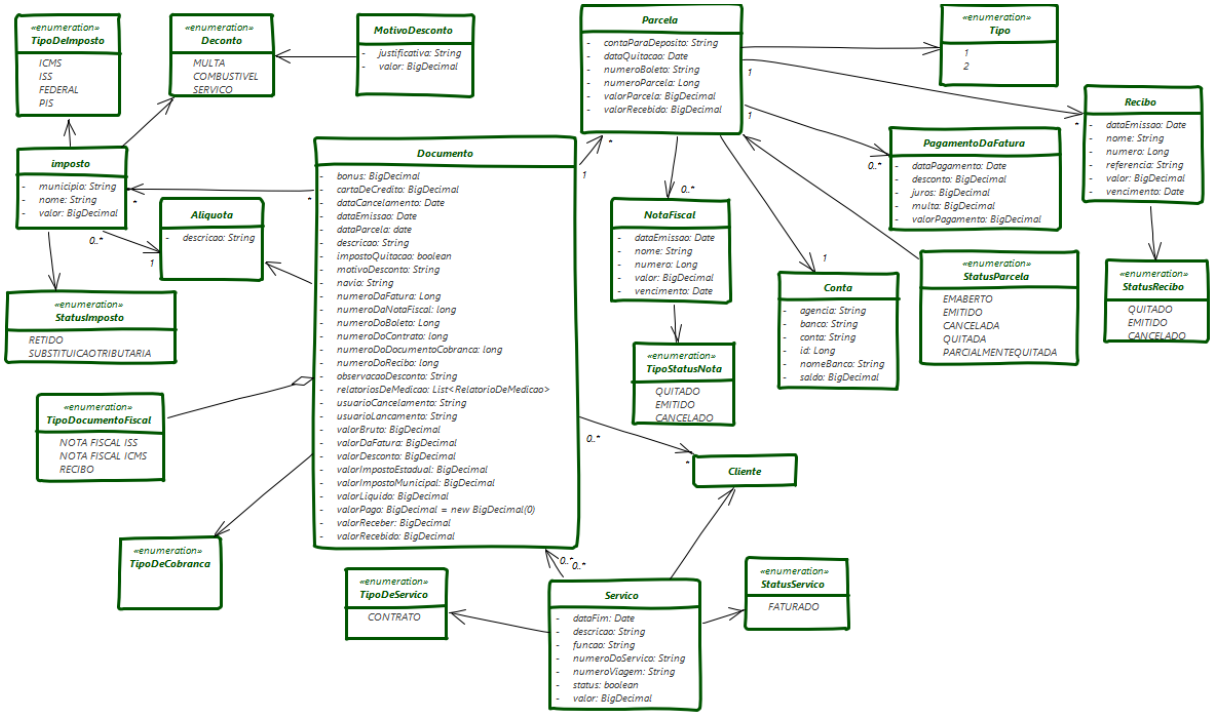
1. Classe PagamentoDaFatura deveria se chamar DadosPagamento ou PagamentoParcela e não da Fatura. (Fácil)
2. Atributo funcao da classe Servico no documento de requisitos é referenciado como atividade. (Difícil)
3. Classe Documento é refenciada no documento como Fatura. (Fácil)
4. TipoPagamentoFatura definido na classe Enumeration como 1 e 2, enquanto que no Cenário é informado (Por Boleto ou por Depósito em Conta). (Difícil)
5. Classe Tipo deveria ser chamado TipoPagamentoFatura (Difícil).
6. Classe TipoDoMotivoDoDesconto apenas referenciado como Desconto (Fácil).
7. Classe Parcela tem o atributo ValorRecebido e o correto seria ValorTotalRecebido. (Médio)

Informação Estranha

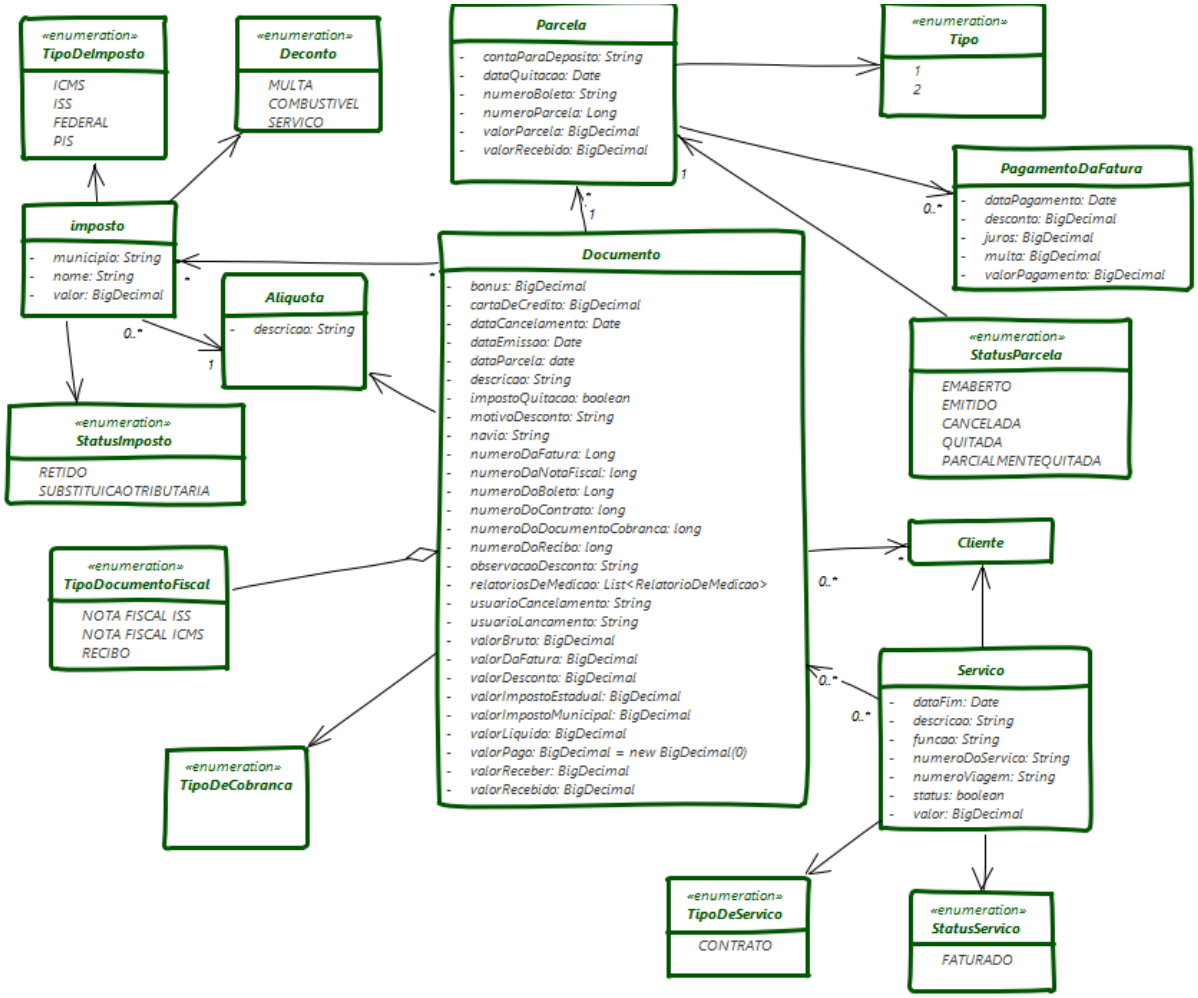
1. Atributo "multa" da classe PagamentoDaFatura não existe. (Fácil)
2. Classe TipoMotivoDedução tem um valor "Serviço" não informado nos Cenários. (Fácil)
3. Classe Fatura/Documento não tem o atributo navio. (Médio)
4. Classe TipoDeImposto tem um valor "PIS" não informado nos Cenários. (Fácil)
5. Classe Fatura/Documento não tem o atributo dataParcela. (Médio)
6. Imposto não deve estar relacionado com TipoMotivoDesconto. (Médio)
7. Atributo "juros" da classe PagamentoDaFatura não existe. (Fácil)

C.2.6 Diagramas

C.2.6.1 Diagrama de Classes Sem Definição de Escopo



C.2.6.2 Diagrama de Classes Com Definição de Escopo



C.2.7 Elementos do Modelo Esperados (EMEs)

UC01 – Emitir Fatura

Classes Esperadas

1.Aliquota
2.Fatura
3.Imposto
4.Parcela
5.Servico
6.StatusDaFatura
7.StatusDaParcela
8.StatusDoImposto
9.StatusDoServico
10.TipoDeCobranca
11.TipoDeDesconto
12.TipoDeDocumentoFiscal
13.TipoDeImposto
14.TipoDePagamento
15.TipoDeServico

Relacionamentos Esperados

1.Fatura ► Imposto
2.Fatura ► Servico
3.Fatura ► StatusdaFatura
4.Fatura ► TipoDeDesconto
5.Fatura ►TipoDeDocumentoFiscal
6.Imposto ► StatusDoImposto
7.Imposto ► Aliquota
8.Imposto ► TipoDeImposto
9.Parcela ► TipoDePagamento
10.Parcela ► StatusDaParcela
11.Parcela ► TipoDeCobranca
12.Servico ► StatusDoServico
13.Servico ► TipoDeServico

Atributos Esperados

1.Fatura.cartaDeCredito
2.Fatura.cliente
3.Fatura.dataEmissao
4.Fatura.descricao
5.Fatura.diasFaturados
6.Fatura.impostoEstadual
7.Fatura.impostoMunicipal
8.Fatura.motivoDosDescontos
9.Fatura.numeroDoContrato
10.Fatura.numeroRelatorioMedicao
11.Fatura.observacaoDosDescontos
12.Fatura.parcela
13.Fatura.retencaoFederal
14.Fatura.servico
15.Fatura.status
16.Fatura.tipoDeDocumentoFiscal
17.Fatura.valorBruto
18.Fatura.valorDaFatura
19.Fatura.valorDoDesconto
20.Fatura.valorLiquido
21.Imposto.aliquota
22.Imposto.municipio
23.Imposto.nomeImposto
24.Imposto.statusImposto
25.Imposto.tipoDeImposto
26.Imposto.valorImposto
27.Parcela.contaParaDeposito
28.Parcela.dataDeVencimento
29.Parcela.numeroDaParcela
30.Parcela.numeroDoBoleto
31.Parcela.statusParcela
32.Parcela.tipoDeCobranca
33.Parcela.tipoDePagamento
34.Parcela.valorDaParcela
35.Servico.atividade
36.Servico.cliente
37.Servico.dataFinal
38.Servico.dataInicial
39.Servico.embarcacao
40.Servico.numero
41.Servico.numeroViagem
42.Servico.statusServico
43.Servico.valor

UC02 – Lançar Recebimento de Fatura

Classes Esperadas

1.Aliquota
2.Cliente
3.DadosPagamento
4.Embarcacao
5.Fatura
6.Imposto
7.Parcela
8.Servico
9.StatusFatura
10.TipoDeCobranca
11.TipoDeImposto
12.TipoDePagamento

Relacionamentos Esperados

1.Fatura ► Cliente
2.Fatura ► Imposto
3.Fatura ► Servico
4.Fatura ► StatusdaFatura
5.Imposto ► Aliquota
6.Imposto ► TipoDeImposto
7.Parcela ► DadosPagamento
8.Parcela ► TipoDeCobranca
9.Parcela ► TipoDePagamento
10.Servico ► Embarcacao

Atributos Esperados

1.DadosPagamento.dataPagamento
2.DadosPagamento.quitado
3.DadosPagamento.valorPagamento
4.Fatura.bonus
5.Fatura.cliente
6.Fatura.dataEmissao
7.Fatura.dataVencimento
8.Fatura.descricao
9.Fatura.impostoQuitacao
10.Fatura.numeroBoleto
11.Fatura.numeroDaFatura
12.Fatura.numeroDaNotaFiscal
13.Fatura.numeroDocumentoCobranca
14.Fatura.numeroDoRecibo
15.Fatura.parcela
16.Fatura.statusDaFatura
17.Fatura.tipoDeCobranca
18.Fatura.tipoPagamento
19.Fatura.valorBruto
20.Fatura.valorDesconto
21.Fatura.valorFatura
22.Fatura.valorLiquido
23.Fatura.valorPago
24.Fatura.valorReceber
25.Fatura.valorTotalRecebido
26.Imposto.tipoDeImposto
27.Imposto.aliquota
28.Imposto.nome
29.Imposto.valorImposto
30.Servico.atividade
31.Servico.dataFinal
32.Servico.dataInicial
33.Servico.embarcacao
34.Servico.numeroDoServico
35.Servico.valor

APÊNDICE D – DOCUMENTAÇÃO DE REQUISITOS E DIAGRAMAS UTILIZADOS NO CAPÍTULO 4

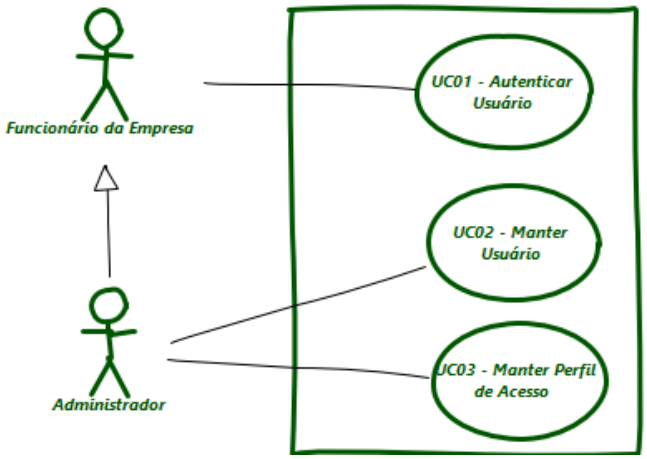
D.1 ESCOPO DO MÓDULO DE GESTÃO DE USUÁRIOS (MGU)

O Módulo de Gestão de Usuários objetiva controlar o acesso dos funcionários da Empresa às funcionalidades do SISTEMA.
O software deve permitir que o acesso às funcionalidades do sistema seja controlado através da autenticação de usuários. Essa autenticação deverá ser através do uso de perfis de acesso aos quais funcionalidades do sistema estarão associadas.
O software deve permitir que usuários sejam cadastrados no sistema com objetivo de criação de login e senha de acesso. Ao ser cadastrado, o usuário será associado a um perfil de acesso e terá seu login e senha enviados por e-mail.
O software deve permitir que perfis de acesso sejam cadastrados no sistema. Ao fazer isso, o usuário deverá associar funcionalidades ao perfil de acesso e também poderá indicar quais usuários estão associados a ele.

D.2 REQUISITOS FUNCIONAIS (RECORTE REAL)

RF1: O software deve permitir que o Administrador efetue a manutenção de usuários (consulta, inclusão e alteração).
RF2: O software deve permitir que o Administrador efetue a manutenção de perfis de acesso (consulta, inclusão, alteração e exclusão).
RF3: O software deve permitir que os Funcionários da Empresa se autentiquem no sistema.
RF4: O software deve permitir que o Administrador associe usuários a perfis de acesso.
RF5: O software deve permitir que o Administrador associe funcionalidades do sistema a perfis de acesso.
RF6: O software deve permitir que o Administrador ative/desative usuários.
RF7: O software deve permitir que o Administrador reinicie a senha dos usuários.

D.2.1 Diagrama de Casos de Uso



D.2.2 Descrição de Alguns Casos de Uso

UC01 – Autenticar Usuário

Objetivo:	Permitir o acesso dos usuários da Empresa ao SISTEMA
Requisitos:	RF3
Atores:	Funcionário da Empresa
Pré-condições:	Não há

Trigger: O ator acessa a autenticação do SISTEMA

Fluxo Principal:

1. O sistema apresenta uma tela contendo as informações:
- Login
- Senha
- Empresa

- As opções:
  - Efetuar Login
  - Esqueci a minha senha

2. O ator informa seu login e senha, seleciona a empresa e depois seleciona a opção Efetuar Login **[A1]**.
3. O sistema verifica se as informações de autenticação são válidas **[RN1][A2]**.
4. O sistema apresenta o logo da empresa selecionada e o menu de funcionalidades de acordo com o perfil do usuário **[RN2]**.

Fluxo Alternativo

[A1] O ator seleciona a opção Esqueci a minha senha

1. O sistema apresenta uma tela contendo as informações:
- Login
- A opção Confirma

2. O ator informa o seu login e seleciona a opção confirma **[A3]**.
3. O sistema apresenta uma mensagem informando o endereço de e-mail para onde a senha foi enviada **[RN3]**.

[A2] Login ou senha inválida

1. O sistema apresenta o formulário de login seguido de uma mensagem de "Login e/ou senha inválido".
2. O caso de uso retorna ao passo 2 do fluxo principal.

[A3] Login inválido

1. O sistema apresenta uma lista com os erros encontrados seguida do próprio formulário com os dados previamente fornecidos.
2. O caso de uso retorna ao passo 2 do fluxo alternativo **[A1]**.

Extensões: Não há

Pós-condições: O ator é autenticado no SISTEMA e as funcionalidades associadas ao seu perfil são apresentadas.

Regras de negócio:
[RN1] O usuário deve ter acesso à empresa que selecionou ao efetuar o login.
[RN2] Cada usuário deve estar associado a somente um perfil e cada perfil deve estar associado a um conjunto funcionalidades que serão disponibilizadas para todos os usuários daquele perfil.
[RN3] O endereço de e-mail para reenvio de senha será aquele cadastrado nos dados do usuário.

UC02 – Manter Usuário

Objetivo: Permitir ao administrador a consulta, inclusão, exclusão ou alteração dos usuários do SISTEMA.

Requisitos: RF1, RF6, RF7

Atores: Administrador

Pré-condições: Administrador deve estar autenticado no SISTEMA

Trigger: O ator seleciona a opção Manter Usuário

Fluxo Principal:

1. O sistema exibe uma tela de filtro com as seguintes informações:
- Login
ou
- Nome
- Perfil (contendo a lista de perfis do sistema)
- Estado (contendo as opções: Todos, Ativado, Desativado)
- Empresa

- Opções:
  - Buscar
  - Adicionar Novo Usuário

- Voltar

2. O usuário preenche os filtros e seleciona a opção Buscar **[RN1][A1] [A2]**
3. O sistema exibe todos os usuários retornados após o filtro com as seguintes informações (somente leitura):
- Login
- Nome
- Perfil
- Estado
- Empresa

- A opção Editar Usuário
- A opção Ativar Usuário **[RN4]**
- A opção Desativar Usuário **[RN3]**
- A opção Reiniciar Senha

O sistema apresenta ao final as opções:
- Adicionar Novo Usuário
- Voltar

4. O ator seleciona a opção Editar Usuário **[A2][A3][A5][A6][A7]**
5. O sistema exibe uma tela de edição de usuário com as seguintes informações:

- Login
- Nome (somente leitura)
- Perfil (contendo a lista de perfis do sistema)
- Empresa (contendo as opções: EmpresaX, Empresa Y ou Ambas)

O sistema apresenta ao final as opções:
- Salvar
- Voltar

6. O ator preenche as informações e seleciona a opção Salvar **[A4]**
7. O sistema valida os dados **[RN2]**
8. O sistema apresenta tela de confirmação de edição de usuário
9. O sistema retorna para a tela inicial.
10. O caso de uso é encerrado.

Fluxo Alternativo

[A1] O ator escolhe a opção Voltar
1. O sistema retorna para a tela inicial.
2. O caso de uso é encerrado

[A2] O ator escolhe a opção Adicionar Novo Usuário
1. O sistema exibe uma tela de cadastro de novo usuário com as seguintes informações:

- Nome (somente leitura)(contendo a opção Buscar Funcionário) **[A8]**
- Login
- Perfil (contendo a lista de perfis do sistema)
- Empresa (contendo as opções: EmpresaX, EmpresaY ou Ambas)

O sistema apresenta ao final as opções:
- Salvar
- Voltar

2. O ator preenche as informações e seleciona a opção Salvar **[A4]**
3. O sistema valida os dados **[RN2]**
4. O sistema gera uma nova senha para o usuário **[RN5]**.
5. O sistema envia um e-mail para o usuário informando o login e a senha.
6. O sistema apresenta tela de confirmação de cadastro de usuário
7. O sistema retorna para a tela inicial.
8. O caso de uso é encerrado.

[A3] O ator escolhe a opção Voltar
1. O sistema retorna ao passo 1 do fluxo principal

[A4] O ator escolhe a opção Voltar

1. O sistema retorna ao passo 3 do fluxo principal

[A5] O ator seleciona a opção Desativar Usuário
1. O sistema solicita a confirmação da operação.
2. O ator confirma.
3. O sistema registra o usuário como desativado
4. O sistema retorna ao passo 3 do Fluxo Principal.

[A6] O ator seleciona a opção Ativar Usuário
1. O sistema solicita a confirmação da operação.
2. O ator confirma.
3. O sistema registra o usuário como ativado.
4. O sistema retorna ao passo 3 do Fluxo Principal.

[A7] O ator seleciona a opção Reiniciar Senha
1. O sistema solicita a confirmação da operação.
2. O ator confirma.
3. O sistema gera uma nova senha para o usuário **[RN5]**.
4. O sistema envia um e-mail para o usuário informando a nova senha.
5. O sistema retorna ao passo 3 do Fluxo Principal.

[A8] O ator seleciona a opção Buscar Funcionário
1. O sistema apresenta tela de busca contendo as informações:
- Nome
- As opções
  * Buscar
  * Voltar
2. O ator informa o dado de busca e seleciona a opção Buscar **[A9]**
3. O sistema apresenta para cada funcionário recuperado:
- Opção de seleção
- Nome
4. O sistema apresenta ao final as opções:
- Selecionar
- Voltar
5. O ator seleciona um funcionário e seleciona a opção Selecionar. **[A10]**
6. O sistema retorna ao passo 1 do fluxo alternativo **[A2]** com o nome do funcionário preenchido.

[A9] O ator seleciona a opção Voltar
1. O sistema retorna ao passo 1 do fluxo alternativo **[A2]**

[A10] O ator seleciona a opção Voltar
1. O sistema retorna ao passo 1 do fluxo alternativo **[A8]**

Extensões: Não se aplica

Pós-condições: Não se aplica

Regras de negócio: **[RN1]** Todos os campos de filtro são opcionais.
[RN2] As informações a seguir são obrigatórias: Login, Nome, Perfil, Empresas (pelo menos uma).
[RN3] A opção de desativação não deve ser apresentada para o próprio administrador que está usando essa funcionalidade. Esta opção é apresentada apenas para usuários com estado Ativo.
[RN4] Esta opção é apresentada apenas para usuários com estado Desativado.
[RN5] A senha de acesso ao SISTEMA deve ser composta de 6 a 8 caracteres, diferentes do login. O conjunto de caracteres válidos para formação de senhas é: [a..z, A..Z, 0..9, _, @]. Caracteres maiúsculos e minúsculos são considerados diferentes.

UC03 – Manter Perfil de Acesso

Objetivo: Permitir ao administrador a inclusão, exclusão ou alteração dos perfis de acesso ao SISTEMA.

Requisitos: RF2, RF4, RF5

Atores: Administrador

Pré-condições: Administrador deve estar autenticado no SISTEMA

Trigger: O ator seleciona a opção Manter Perfis de Acesso

Fluxo Principal:

1. O sistema apresenta a lista de perfis de acesso existentes, ordenada por nome do perfil:
- Nome do Perfil
- Descrição
- A opção Definir Usuários
- A opção Definir Funcionalidades
- A opção Editar Perfil de Acesso
- A opção Excluir Perfil de Acesso **[RN1]**

- O sistema apresenta ao final as opções:
  - Adicionar Novo Perfil de Acesso
  - Voltar

2. O ator seleciona a opção Adicionar Novo Perfil de Acesso **[A1][A2][A3][A4][A6]**
3. O sistema apresenta tela de inclusão de novo perfil de acesso contendo as informações:
- Nome do Perfil
- Descrição

- As opções:
  - Salvar
  - Voltar
4. O ator informa os dados do novo perfil de acesso e seleciona a opção Salvar **[A5]**
5. O sistema verifica se as informações estão válidas **[RN2][RN3][A7]**
6. O sistema inclui o novo perfil de acesso.
7. O sistema apresenta mensagem de perfil de acesso cadastrado com sucesso.
8. O caso de uso retorna ao passo 1 do fluxo principal.

Fluxo Alternativo

[A1] O ator seleciona a opção Definir Usuários

1. O sistema apresenta tela de associação de usuários ao perfil de acesso contendo:

(Usuários Associados)
- Lista de usuários associados contendo para cada um deles:
  - Login
  - Nome
  - A opção Desassociar Usuário

(Novos Usuários)
- Usuário (contendo a lista com o Nome dos usuários cadastrados no sistema)
- A opção Associar Usuário

- O sistema apresenta ao final as opções:
  - Voltar

2. O ator seleciona um usuário e seleciona a opção Associar Usuário **[A8] [A5]**
3. O sistema inclui o usuário selecionado na lista de Usuários associados **[RN3]**.
4. O caso de uso retorna ao passo 1 do Fluxo Alternativo **[A1]**

[A2] O ator seleciona a opção Definir Funcionalidades

1. O sistema apresenta todas as funcionalidades do SISTEMA estruturadas da mesma forma como que são apresentadas no menu do sistema (itens e sub-itens). Para cada item folha são apresentadas as opções **[RN4] [RN5] [RN6]**:
- Acessa (caixa de seleção)
- Tipo de Acesso (contendo as opções: Leitura, Edição, Total)

- O sistema apresenta ao final as opções:
  - Salvar
  - Voltar

2. O ator seleciona as funcionalidades associadas ao perfil e seleciona a opção Salvar **[A5]**.
3. O sistema registra as funcionalidades associadas ao perfil.
4. O caso de uso retorna ao passo 1 do Fluxo Principal.

[A3] O ator seleciona a opção Editar Perfil de Acesso
1. O sistema apresenta uma tela contendo as seguintes informações:
- Nome do Perfil
- Descrição

As opções:
- Salvar
- Voltar

2. O ator informa a alteração e seleciona a Opção Salvar **[A5]**
3. O sistema verifica os dados informados **[RN2][A7]**
4. O sistema atualiza os dados do perfil.
5. O caso de uso retorna ao passo 1 do Fluxo Principal.

[A4] O ator seleciona a opção Excluir Perfil de Acesso
1. O sistema solicita a confirmação da operação.
2. O ator confirma a exclusão.
3. O sistema exclui o perfil de acesso.
4. O caso de uso retorna ao passo 1 do Fluxo Principal.

[A5] O ator seleciona a opção Voltar
1. O caso de uso retorna ao passo 1 do Fluxo Principal

[A6] O ator seleciona a opção Voltar
1. O sistema retorna para a tela inicial.
2. O caso de uso é encerrado.

[A7] Dados fornecidos não estão corretos
1. O sistema apresenta uma lista com os erros encontrados seguida do próprio formulário com os dados previamente fornecidos.
2. O caso de uso retorna ao passo 4 do Fluxo Principal ou ao passo 2 do Fluxo Alternativo **[A3]**.

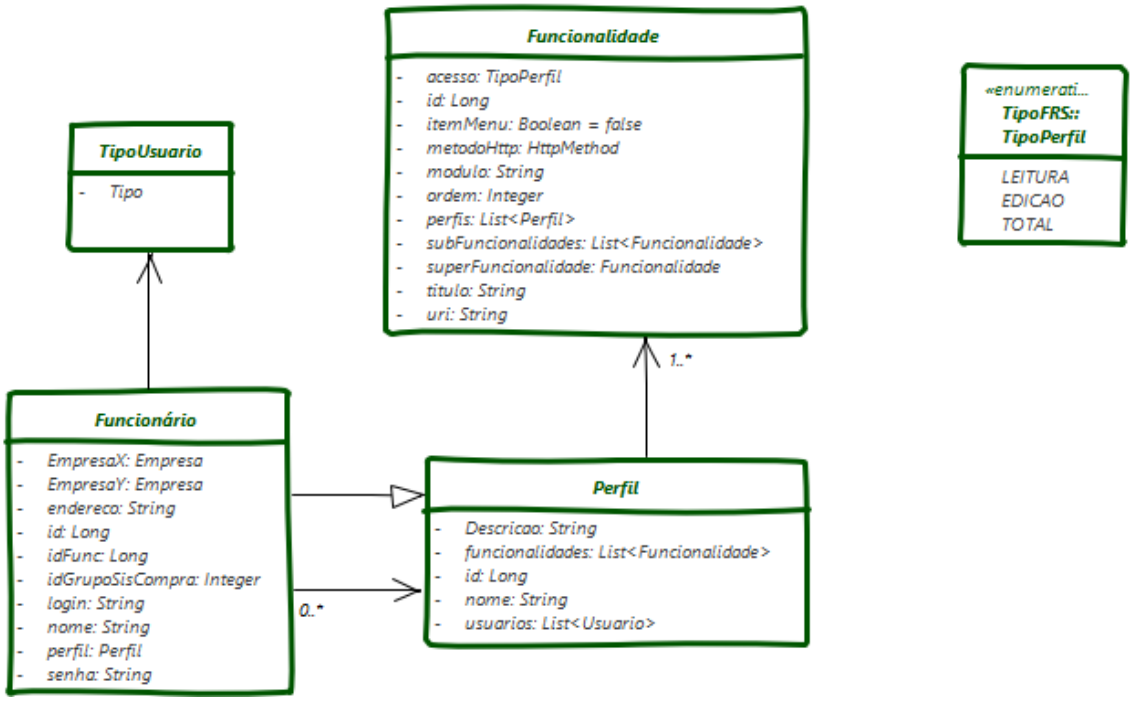
[A8] O ator seleciona a opção Desassociar Usuário
1. O sistema retira da lista de usuários associados o usuário escolhido.
2. O caso de uso retorna ao passo 1 do Fluxo Alternativo **[A1]**.

Extensões: Não há

Pós-condições: Não se aplica.

Regras de negócio: **[RN1]** A opção de exclusão só estará disponível se o perfil não estiver associado a nenhum usuário.
[RN2] O nome do perfil é obrigatório e deve ser único, ou seja, não podem existir dois perfis com o mesmo nome.
[RN3] Caso já exista um perfil associado ao usuário, essa associação será substituída pela atual. Um usuário só pode estar associado a um perfil de acesso.
[RN4] Ao selecionar qualquer tipo de acesso, o campo Acesso é automaticamente selecionado.
[RN5] Ao marcar o campo Acesso, o campo tipo de acesso é alterado automaticamente para Leitura.
[RN6] Caso nenhuma opção seja selecionada, o ator não terá acesso a ela. Caso seja selecionada a opção Leitura, o ator terá apenas acesso de leitura àquela funcionalidade. Caso seja selecionada a opção Edição, o ator terá acesso de leitura, inclusão e edição da funcionalidade associada. Por fim, caso seja selecionada a opção Total, o ator terá acesso de leitura, inclusão, exclusão e edição da funcionalidade associada.

D.2.3 Diagrama de Classes com Defeitos



D.2.4 Elementos do Modelo Esperados

CLASSES ESPERADAS

1. Funcionalidade
2. Perfil
3. Usuario

RELACIONAMENTOS ESPERADOS

1. Perfil ► Funcionalidade
2. Perfil ► Usuario

ATRIBUTOS ESPERADOS

1. Funcionalidade.id
2. Funcionalidade.itemMenu
3. Funcionalidade.metodoHttp
4. Funcionalidade.modulo
5. Funcionalidade.ordem
6. Funcionalidade.subFuncionalidades
7. Funcionalidade.superFuncionalidades
8. Funcionalidade.titulo
9. Funcionalidade.uri
10. Usuario.estado
11. Usuario.id
12. Usuario.idFunc
13. Usuario.idGrupoSisCompra
14. Usuario.luaNova
15. Usuario.nome
16. Usuario.perfil
17. Usuario.senha
18. Usuario.transShip
19. Perfil.decricao
20. Perfil.funcionalidades
21. Perfil.id
22. Perfil.nome