



Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Construção de um ambiente Integrado de Monitoramento, Interpretação e Suporte à tomada de decisão acerca da qualidade interna do produto de software, com uso de um ambiente de Data Warehousing: um Estudo de Caso preliminar na Caixa

Autor: Nilton Cesar Campos Araruna
Orientador: Prof. Msc. Hilmer Rodrigues Neri

Brasília, DF

2015



Nilton Cesar Campos Araruna

**Construção de um ambiente Integrado de
Monitoramento, Interpretação e Suporte à tomada de
decisão acerca da qualidade interna do produto de
software, com uso de um ambiente de Data
Warehousing: um Estudo de Caso preliminar na Caixa**

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Msc. Hilmer Rodrigues Neri

Brasília, DF

2015

Nilton Cesar Campos Araruna

Construção de um ambiente Integrado de Monitoramento, Interpretação e Suporte à tomada de decisão acerca da qualidade interna do produto de software, com uso de um ambiente de Data Warehousing: um Estudo de Caso preliminar na Caixa/ Nilton Cesar Campos Araruna. – Brasília, DF, 2015-

162 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Msc. Hilmer Rodrigues Neri

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2015.

1. Métricas de Código-Fonte. 2. *Data Warehousing*. I. Prof. Msc. Hilmer Rodrigues Neri. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Construção de um ambiente Integrado de Monitoramento, Interpretação e Suporte à tomada de decisão acerca da qualidade interna do produto de software, com uso de um ambiente de Data Warehousing: um Estudo de Caso preliminar na Caixa

CDU a obter

Nilton Cesar Campos Araruna

Construção de um ambiente Integrado de Monitoramento, Interpretação e Suporte à tomada de decisão acerca da qualidade interna do produto de software, com uso de um ambiente de Data Warehousing: um Estudo de Caso preliminar na Caixa

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 07 de Julho de 2015:

Prof. Msc. Hilmer Rodrigues Neri
Orientador

Profa. Milene Serrano
Convidado 1

Prof. Luiz Augusto Fontes Laranjeira
Convidado 2

Brasília, DF
2015

Este trabalho é dedicado ao meu pai, Aurenilton Araruna, minha mãe, Cláudia Araruna e ao meu irmão, Gustavo Campos. Estas inseriram na minha vida as virtudes do esforço.

Agradecimentos

Agradeço primeiramente aos meus pais, Aurenilton e Claudia, por sempre acreditarem em mim e no meu potencial, por me darem apoio, força e amor sempre que necessário, por me proporcionarem uma vida excelente cheia de conforto e por não medirem esforços na hora de me amparar. Obrigado por me ensinarem os grandes valores da vida, do valor do esforço, do tempo e do dinheiro. Vocês sempre serão meus heróis e minhas grandes referências.

Agradeço ao meu irmão, Gustavo Henrique, que sempre me mostrou o que é ser um homem intelectual e de caráter, mesmo com tamanhas diferenças entre nós, sempre me serviu como modelo. Se não fossem suas conquistas não acreditaria que as minhas seriam possíveis.

Agradeço a minha namorada, Giovana Giugliani, que esteve ao meu lado em toda essa fase da minha vida, do cursinho até a graduação, vivenciando todo meu esforço, minhas dúvidas e minha ansiedade. Agradeço pelo apoio, pela paciência, pela ajuda, pela companhia em noites de estudo e as tornando bem melhor, também peço desculpa pelo estresse e pelo tempo que tive que me ausentar do seu lado.

Agradeço ao meu orientador Hilmer Rodrigues Neri pela confiança depositada em mim, pelo aprendizado dentro e fora de sala e por ter me motivado ao longo da graduação. Agradeço aos meus amigos e colegas Matheus, Pedro Tomioka e Gustavo Rodrigues pela colaboração e ajuda não só durante a execução desse trabalho mas também em toda a graduação.

Agradeço a toda a minha família, avós, madrinha, padrinho, tios, tias, primos, primas. Em especial ao meu avô, Gentil Cruz, que mesmo nos deixando em fevereiro deste ano continua a me dar força. E aos meus amigos de infância, vulgo frangos, por entenderem a minha ausência durante essa fase da minha vida e por sempre desejarem a minha graduação e um belo futuro profissional para poder pagar a conta do bar.

Agradeço a equipe PEDeS da CAIXA, onde tive o prazer de estagiar, em especial ao meu supervisor, Diego Costa, pela oportunidade, pelo aprendizado que levarei pelo resto da vida, por me prepararem para o mercado de trabalho, por me ensinarem tudo o que sei sobre ser um profissional e pelos momentos de alegria. Sei que aprendi com os melhores. Ainda agradeço a confiança da minha atual preposta, Fernanda Paiva, que me deu a oportunidade de dar continuidade neste aprendizado.

Agradeço ao grupo kanburn, Thiago Kairala, Bruno Contessoto, Rafael Fazzolino, Eduardo Brasil, Thabata Granja e em especial ao meu primo João Araruna que ingressou

na universidade junto comigo e participou da minha luta, pelos encontros regados de estudos, pizza e alegria. Os melhores e mais chatos amigos que não escolhi viver mas fui obrigado e hoje a obrigação é manter cada um na minha vida.

"Learn from yesterday, live for today, hope for tomorrow. The important thing is not to stop questioning."
(Albert Einstein)

Resumo

A qualidade do software depende da qualidade do código-fonte, um bom código-fonte é um bom indicador de qualidade interna do produto de *software*. Portanto, o monitoramento de métricas de código-fonte de um *software* significa melhorar a sua qualidade. Existem diversas soluções e ferramentas para se obter um monitoramento de métricas de *software* e que conseguem extrair valores de métricas de código com facilidade. Porém, a decisão sobre o que fazer com os dados extraídos ainda é uma dificuldade relacionada à visibilidade e interpretação dos dados. O objetivo principal deste trabalho é evoluir o ambiente de *Data Warehousing(DW)* proposto por [Neri et al. \(2014\)](#) e utilizá-la para assistir ao processo de aferição de qualidade interna de um produto de software desenvolvido em um contrato de prestação de serviço para uma organização pública. Este ambiente visa facilitar a interpretação, visibilidade e avaliação dos *bugs*, das violações e das métricas de código-fonte, associando-as a cenários de limpeza. Os cenários de limpeza buscam apoiar as tomadas de decisão que refletem na alteração do código-fonte e um *software* com mais qualidade. Para um melhor entendimento da solução DW proposta e dos elementos que dizem respeito a sua arquitetura e seus requisitos de negócio foram apresentadas as fundamentações teóricas necessárias. Um projeto para a realização de uma investigação empírica foi elaborado utilizando a técnica de estudo de caso. O projeto visa responder questões qualitativas e quantitativas a respeito do poder de assistência ao processo de aferição da qualidade da solução citada na CAIXA Econômica Federal.

Palavras-chaves: Métricas de Código-Fonte. *Data Warehousing*. *Data Warehouse*.

Abstract

The quality of the software depends on the quality of the source code, a good source code is a good indicator of the software internal quality. Therefore, the monitoring of metrics of a software source-code results in improving its quality. There are several solutions and tools to achieve software monitoring of metrics that can easily extract values of metrics of code. However, the decision regarding what to do with the extracted data is still an issue related to the visibility and interpretation of this data. The main goal of this project is to improve the Data Warehousing environment proposed by [Neri et al. \(2014\)](#) and use to attend to internal quality assesment process of a software developed in a outsourcing services contract to a public organization. This environment aim to facilitate interpretation, visibility and evaluation of bugs, violation and source code metrics, associating them with cleansing scenarios. The cleansing scenarios are to support decision making processes that reflect on the alteration of the source code and a software with greater quality. For a better understanding of the DW solution proposedand of the elements that concern its arquitecture and its business requirements, the necessary theoretical fundaments have been presented. The project aims to answer qualitative and quantitative questions about the power about the power of assistance to the solution of quality benchmarking process cited in CAIXA Econômica Federal.

Key-words: Source Code Metrics, Data Warehousing, Data Warehouse, Effectiveness, efficiency

Listas de ilustrações

Figura 1 – Metodologia de Pesquisa	21
Figura 2 – Passos do Estudo de Caso	24
Figura 3 – Estrutura do <i>Kanban</i>	25
Figura 4 – Modelo de Qualidade do Produto da ISO/IEC 9126 (2001)	28
Figura 5 – Atividades de aquisição extraída de ISO/IEC 15939 (2008)	47
Figura 6 – Arquitetura de um ambiente de Data Warehousing	52
Figura 7 – Esquema estrela extraído de Times (2012)	54
Figura 8 – Exemplo de operações <i>Drill Down</i> (direita) e <i>Drill up</i> (esquerda) extraídos de Golfarelli (2009)	55
Figura 9 – Exemplo de operações <i>Slice</i> (acima) e <i>Dice</i> (embaixo) extraídos de Golfarelli (2009)	56
Figura 10 – Exemplo da operação <i>Drill Across</i> extraído de Golfarelli (2009)	57
Figura 11 – Exemplo da operação <i>Pivoting</i> extraído de Golfarelli (2009)	57
Figura 12 – Metodologia de Projeto de <i>Data Warehouse</i> proposta por Kimball e Ross (2002) extraída de Rêgo (2014)	62
Figura 13 – Projeto físico do <i>Data Warehouse</i> extraído de Rêgo (2014)	65
Figura 14 – Projeto físico do <i>Data Warehouse</i> extraído de Rêgo (2014)	67
Figura 15 – Parte do projeto físico contendo as tabelas relacionados ao PMD	68
Figura 16 – Parte do projeto físico contendo as tabelas relacionados ao <i>FindBugs</i>	69
Figura 17 – Projeto físico do <i>Data Warehouse</i> finalizado.	70
Figura 18 – Arquitetura Ambiente de <i>Data Warehousing</i> para Métricas de Código Fonte extraído de Rêgo (2014) e adaptado.	72
Figura 19 – Escopo do Estudo de Caso	75
Figura 20 – Estrutura do Estudo de Caso	78
Figura 21 – Fórmula de Índice de Defeitos	82
Figura 22 – <i>Dashboard</i> Quantidade Total	87
Figura 23 – <i>Dashboard</i> PMD	88
Figura 24 – <i>Dashboard</i> Cenários	89
Figura 25 – <i>Dashboard</i> <i>FindBugs</i>	90
Figura 26 – Exemplo dos dados coletados a partir do <i>dashboard</i>	96
Figura 27 – Gráfico de Dispersão entre Cenários de Limpeza e <i>Bugs</i>	102
Figura 28 – Gráfico de Dispersão entre Cenários de Limpeza e Violações	102
Figura 29 – Gráfico de Dispersão entre Violações e <i>Bugs</i>	103
Figura 30 – Componentes do Kettle que foram utilizadas nas Transformações	113
Figura 31 – Primeira Transformação realizada no Kettle	114
Figura 32 – Segunda Transformação realizada no Kettle	115

Figura 33 – Terceira Transformação realizada no Kettle	128
Figura 34 – Quarta Transformação realizada no Kettle	134
Figura 35 – Quinta Transformação realizada no Kettle	136
Figura 36 – Componentes do Kettle que foram utilizadas nos <i>Jobs</i>	137
Figura 37 – <i>Job</i> deste Trabalho	138
Figura 38 – <i>Dashboard Geral</i>	139
Figura 39 – <i>Dashboard Cenários</i>	144
Figura 40 – <i>Dashboard FindBugs</i>	145
Figura 41 – <i>Dashboard PMD</i>	146

Lista de tabelas

Tabela 2 – Descrição das classificações adotadas de pesquisa, conceitos extraídos de Silva e Menezes (2005) parte 1.	22
Tabela 4 – Descrição das classificações adotadas de pesquisa, conceitos extraídos de Silva e Menezes (2005) parte 2.	23
Tabela 5 – Percentis para métrica NOC em projetos Java extraídos de Meirelles (2013)	32
Tabela 6 – Nome dos Intervalos de Frequência extraídos de Rêgo (2014)	33
Tabela 7 – Configurações para os Intervalos das Métricas para Java extraídas de Rêgo (2014)	34
Tabela 8 – Conceitos de Limpeza levantados por Machini et al. (2010) e adaptados por Rêgo (2014) parte 1.	36
Tabela 9 – Conceitos de Limpeza levantados por Machini et al. (2010) e adaptados por Rêgo (2014) parte 2.	37
Tabela 10 – Conceitos de Limpeza levantados por Machini et al. (2010) e adaptados por Rêgo (2014) parte 3.	38
Tabela 11 – Conceitos de Limpeza levantados por Machini et al. (2010) e adaptados por Rêgo (2014) parte 4.	39
Tabela 12 – Cenários de Limpeza extraídos de Rêgo (2014)	41
Tabela 13 – Comparativo das ferramentas utilizadas	44
Tabela 14 – Comparativo das ferramentas utilizadas	45
Tabela 15 – Diferenças entre OLTP e OLAP extraídas de Neri (2002)	55
Tabela 16 – Fatos e dimensões identificadas por Rêgo (2014)	63
Tabela 17 – Fatos e dimensões identificadas neste trabalho.	64
Tabela 18 – Tabelas fatos e tabelas dimensões elaboradas por Rêgo (2014)	66
Tabela 19 – Descrição das Tabelas do Metadados do <i>Data Warehouse</i>	67
Tabela 20 – Tabelas fatos e tabelas dimensões adicionadas à solução de Rêgo (2014)	71
Tabela 21 – Multa por defeitos	83
Tabela 22 – Resultados do EQM do Findbugs	92
Tabela 23 – Resultados do EQM do PMD	93
Tabela 24 – Continuaçao dos resultados do EQM do PMD	94
Tabela 25 – Continuaçao dos resultados do EQM do PMD	95
Tabela 26 – Exemplo dos dados coletados a partir do arquivo gerado pela ferramenta PMD	97
Tabela 27 – Continuação exemplo dos dados coletados a partir do arquivo gerado pela ferramenta PMD	98

Tabela 28 – Continuação exemplo dos dados coletados a partir do arquivo gerado pela ferramenta PMD	99
Tabela 29 – Significância de índice de Correlação	100
Tabela 30 – Resultado do cálculo do coeficiente de correlação linear	101

Lista de abreviaturas e siglas

ACC	<i>Afferent Connections per Class</i>
ACCM	<i>Average Cyclomatic Complexity per Method</i>
AMLOC	<i>Average Method Lines of Code</i>
ANPM	<i>Average Number of Parameters per Method</i>
CBO	<i>Coupling Between Objects</i>
CETEC	<i>Centralizadora Nacional de Tecnologia da Informação</i>
CSV	<i>Comma-Separated Values</i>
DIT	<i>Depth of Inheritance Tree</i>
DW	<i>Data Warehouse</i>
DWing	<i>Data Warehousing</i>
SIGET	<i>Sistema de Gestão Financeira de Bens e Serviços de Tecnologia da Informação</i>
GEGAT	<i>Gestão de Ativos de TI</i>
ETL	<i>Extraction-Transformation-Load</i>
GITECBR	<i>Gerencia de Filial de Suporte Tecnológico de Brasília</i>
GQM	<i>Goal-Question-Metric</i>
IEC	<i>International Electrotechnical Commission</i>
ISO	<i>International Organization for Standardization</i>
LCOM4	<i>Lack of Cohesion in Methods</i>
LOC	<i>Lines of Code</i>
MCTI	<i>Modelo de Contratações de Soluções de TI</i>
NPA	<i>Number of Public Attributes</i>
NOC	<i>Number of Children</i>
NOM	<i>Number of Methods</i>

OLAP	<i>On-Line Analytical Processing</i>
OLTP	<i>Online Transaction Processing</i>
RFC	<i>Response For a Class</i>
SCAM	<i>IEEE International Working Conference on Source Code Analysis and Manipulation</i>
CDE	<i>community Dashboard Editor</i>
CDA	<i>Community Dashboard Access</i>
CDF	<i>Community Dashboard Framework</i>
CCC	<i>Community Chart Components</i>
SISP	<i>Sistema de Administração de Recursos de Tecnologia da Informação e Informática</i>
S&SC	<i>Software e Serviços Correlatos</i>
XML	<i>Extensible Markup Language</i>

Sumário

1	INTRODUÇÃO	18
1.1	Contexto	18
1.2	Justificativa	18
1.3	Problema	19
1.4	Objetivos	19
1.5	Metodologia de pesquisa	20
1.6	Metodologia de Desenvolvimento	25
1.7	Organização do Trabalho	26
2	MÉTRICAS DE SOFTWARE	27
2.1	Processo de Medição	27
2.2	Definição das métricas de software	27
2.3	Métricas de código fonte	29
2.3.1	Métricas de tamanho e complexidade	29
2.3.2	Métricas de Orientação a Objetos	30
2.4	Configurações de qualidade para métricas de código fonte	31
2.5	Cenários de limpeza	35
2.5.1	Verificadores de Erro	42
2.5.2	Ferramentas de verificação de Erro	42
2.6	Considerações Finais do Capítulo	45
3	CONTRATAÇÕES DE FORNECEDORES DE DESENVOLVIMENTO DE SOFTWARE	46
3.1	Importância da Contratação de Fornecedores de Desenvolvimento de Software	46
3.2	Instrução Normativa Nº 04	48
3.3	Considerações finais do capítulo	50
4	DATA WAREHOUSE	51
4.1	Definição	51
4.2	<i>Extraction- Transformation- Load</i>	52
4.3	Modelagem Dimensional	53
4.3.1	OLAP (<i>On-Line Analytic Processing</i>)	54
4.4	Visualização de Dados	57
4.4.1	Ferramenta para criação de <i>Dashboards</i>	59

4.5	Ambiente de <i>Data Warehousing</i> para Métricas, bugs e violações de Código-Fonte	60
4.5.1	Ferramentas de <i>Data Warehousing</i>	71
4.6	Considerações finais do capítulo	72
5	ESTUDO DE CASO	74
5.1	Planejamento do Estudo de Caso	74
5.2	Background	76
5.2.1	Trabalhos Antecedentes e Relacionados	76
5.2.2	Questão de Pesquisa	77
5.3	Design	80
5.4	Seleção	80
5.5	O Software Analisado	84
5.6	Fonte dos Dados Coletados e Método de Coleta	85
5.7	Processo de análise dos dados	85
5.8	Ameaças a validade do estudo de caso	86
5.9	Execução do Estudo de Caso e Análise dos Dados	86
5.9.1	Análise do Erro Quadrático Médio	90
5.9.2	Análise do Coeficiente de Correlação	100
5.9.3	Analise do Questionário	103
5.10	Considerações finais do capítulo	105
6	CONCLUSÃO	106
6.1	Limitações	107
	Referências	109
	APÊNDICE A – DESCRIÇÃO DO PROCESSO DE ETL NO KET-TLE	113
A.1	Implementações das <i>Transformations</i>	113
A.2	Implementação do <i>Job</i>	137
	APÊNDICE B – IMPLEMENTAÇÃO DO DASHBOARD	139
B.1	<i>Dashboard</i> Geral	139
B.2	<i>Dashboards</i> Cenários, FindBugs e PMD	143
	APÊNDICE C – QUESTIONÁRIO	155
	APÊNDICE D – RELATÓRIO KANBANFLOW	159

1 Introdução

1.1 Contexto

Segundo(WILLCOCKS; LACITY, 2001) a terceirização do desenvolvimento de software é uma prática cada vez mais adotada por organizações. A terceirização de atividades, ou seja, o ato de transferir para fora da organização uma parte do seu processo produtivo não é uma prática recente. Atividades e processos que eram muito específicos dentro de uma organização foram transferidos, de forma parcial ou total, para outras organizações ou agentes externos (LEITE, 1997).

Uma das motivações para a terceirização é a qualidade do serviço prometida pelas empresas fornecedoras. No entanto, existem vários riscos associados à decisão pela terceirização, que podem comprometer a qualidade esperada, como por exemplo: as expectativas de serviço e a resposta rápida não serem atendidas adequadamente; o serviço prestado apresentar qualidade inferior ao existente anteriormente; e as tecnologias utilizadas não corresponderem ao esperado(WILLCOCKS; LACITY, 2001). Segundo SOFTEX (2013a) a aquisição de um *software* é um processo complexo, principalmente no que diz respeito à caracterização dos requisitos necessários ao *software* e às condições envolvidas na contratação como a qualidade esperada.

Acompanhando o ritmo da terceirização o cenário com empresas terceirizadas contratadas para o desenvolvimento de *software* está cada vez maior em organizações públicas. Tais organizações não são diretamente responsáveis pelo desenvolvimento do *software* mas são responsáveis pelo processo de verificação de sua qualidade conforme a norma (IN4, 2014) que será explicada no capítulo 3 deste trabalho.

1.2 Justificativa

Segundo (BECK, 1999)(FOWLER, 1999), a qualidade de *software* é medida pela qualidade de seu código-fonte. Conforme a (ISO/IEC 15939, 2002), medição é uma ferramenta primordial para avaliar a qualidade dos produtos e a capacidade de processos organizacionais, portanto, o Órgão Público contratante pode fazer uso do monitoramento de métricas de código-fonte para assistir ao processo de aferição de qualidade do *software* desenvolvido pela contratada.

Neri et al. (2014) propôs uma solução automatizada para o monitoramento e interpretação de métricas de código-fonte com suporte de um ambiente de *Data Warehousing*, que será explicada no Capítulo 4. Este trabalho trata da evolução dessa solução, aplicada

em uma Organização Pública Federal. Uma das principais contribuições deste trabalho foi a adição de: i)uma camada de visualização de métricas de código-fonte; ii) novos coletores de métricas de código-fonte. Estas adições unificaram a visualização e interpretação de diferentes tipos de métricas de código-fonte em um mesmo ambiente.

1.3 Problema

Com foco na avaliação de controles gerais de Tecnologia da Informação nos Órgãos públicos, em 2011, o Tribunal de Contas da União-TCU detectou, por meio de auditorias de governança de TI em diversos Órgãos, uma considerável frequência de irregularidades relacionadas à inexistência, deficiências e a falhas de processos de *software* que comprometem a eficácia e eficiência das contratações de desenvolvimento de sistemas ([TCU, 2011](#)).

A inexistência de parâmetros de referência que para aferição da qualidade interna de código-fonte em contratações públicas expõe uma fragilidade nesse processo, decorrente da falta de uma metodologia que assegure a análise e interpretação dessas medidas. Os critérios indicados pelo TCU no acórdão ([TCU, 2011](#)) foram : Constituição Federal, art. 37, caput; Instrução Normativa 4/2008, SLTI/MPOG, art. 12, inciso II; Lei 8666/1993, art. 6º, inciso IX; Norma Técnica - ITGI - Cobit 4.1, PO8.3 - Padrões de desenvolvimento e de aquisições; Norma Técnica - NBR ISO/IEC - 12.207 e 15.504;e Resolução 90/2009, CNJ, art. 10.

A auditoria do acordão ([TCU, 2011](#)) reforça o problema da falta de capacidade da administração pública em aferir a qualidade interna dos produtos de software desenvolvido por terceirizadas. Tendo esse problema como motivação desta investigação científica, a questão geral de pesquisa deste trabalho é:

O uso de um ambiente de *Data Warehousing* para aferição da qualidade interna de software, apoiado por ferramentas de análise estática de código-fonte, pode assistir ao processo de aferição da qualidade interna em uma organização pública federal?

1.4 Objetivos

O objetivo geral deste trabalho é evoluir a solução proposta por ??) e utilizá-la para assistir ao processo de aferição de qualidade interna de um produto de software desenvolvido em um contrato de prestação de serviço para uma organização pública. Dentre os objetivos específicos deste trabalho estão:

- Adicionar à solução de *DWing* a análise de erros e violações de código-fonte fornecidas por ferramentas já em uso na organização pública analisada, de forma a prover

uma ambiente integrado e automatizado para análise, interpretação e visualização de dados de código-fonte;

- Construir uma camada de visualização de dados de forma a demonstrar as medidas e/ou suas interpretações, por meio de painéis informativos no ambiente de DWing.
- Analisar um sistema adquirido por um Órgão Púlico utilizando à solução de *DWing* evoluída.

1.5 Metodologia de pesquisa

O que é uma pesquisa? De acordo com (GIL, 2002) é o procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos. A pesquisa é requerida quando não se dispõe de informação suficiente para responder ao problema, ou então quando a informação disponível se encontra em tal estado de desordem que não possa ser adequadamente relacionada ao problema.

Nessa seção, apresenta-se a metodologia de pesquisa adotada neste trabalho. Para isso, foram definidos: a natureza da pesquisa; o tipo de metodologia de pesquisa; o tipo de abordagem de pesquisa; os métodos de procedimentos de pesquisa e os tipos de técnicas de coletas de dados.

Os procedimentos de pesquisa selecionados foram pesquisa bibliográfica, documental, levantamento e estudo de caso. As técnicas de coleta de dados selecionadas foram entrevistas, questionários e registro de observação na vida real. A seleção metodológica é apresentada na Figura 1 e a descrição das classificações adotadas se encontra nas Tabelas 2 e 4.

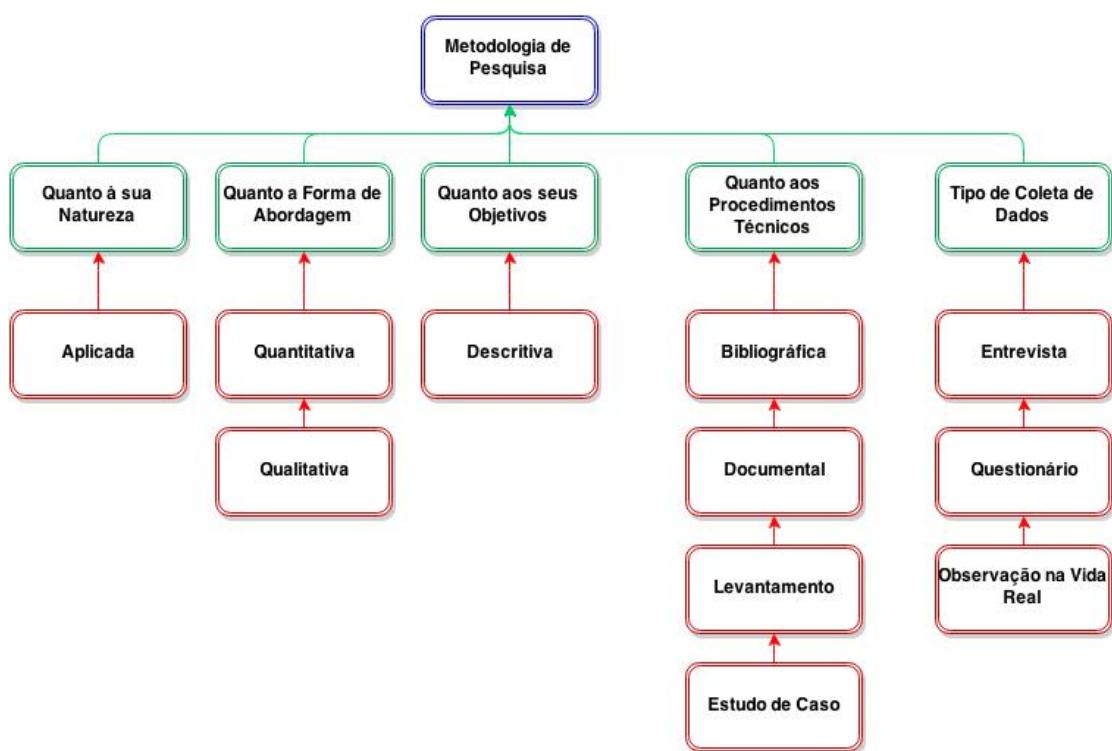


Figura 1 – Metodologia de Pesquisa

Ponto de Vista	Selecionada	Descrição
Natureza	Aplicada	Objetiva gerar conhecimentos para aplicação prática e dirigidos à solução de problemas específicos. Envolve verdades e interesses locais.
Forma de Abordagem	Quantitativa	Considera que tudo pode ser quantificável, o que significa traduzir em números opiniões e informações para classificá-las e analisá-las. Requer o uso de recursos e de técnicas estatísticas (percentagem, média, moda, mediana, desvio-padrão, coeficiente de correlação e análise de regressão).
	Qualitativa	Considera que há uma relação dinâmica entre o mundo real e o sujeito, isto é, um vínculo indissociável entre o mundo objetivo e a subjetividade do sujeito que não pode ser traduzido em números. A interpretação dos fenômenos e a atribuição de significados são básicas no processo de pesquisa qualitativa. O ambiente natural é a fonte direta para coleta de dados e o pesquisador é o instrumento-chave. Os pesquisadores tendem a analisar seus dados indutivamente
Objetivos	Descriptiva	Visa descrever as características de determinada população ou fenômeno ou o estabelecimento de relações entre variáveis. Envolve o uso de técnicas padronizadas de coleta de dados: questionário e observação sistemática. Assume, em geral, a forma de Levantamento.
Procedimentos Técnicos	Pesquisa Bi-bibliográfica	Quando elaborada a partir de material já publicado, constituído principalmente de livros, artigos de periódicos e atualmente com material disponibilizado na Internet.
	Pesquisa Documental	Quando elaborada a partir de materiais que não receberam tratamento analítico.
	Levantamento	Quando a pesquisa envolve a interrogação direta das pessoas cujo comportamento se deseja conhecer.
	Estudo de Caso	Quando envolve o estudo profundo e exaustivo de um ou poucos objetos de maneira que se permita o seu amplo e detalhado conhecimento.

Tabela 2 – Descrição das classificações adotadas de pesquisa, conceitos extraídos de [Silva e Menezes \(2005\)](#) parte 1.

Tipo de Coleta de Dados	Entrevista	É a obtenção de informações de um entrevistado, sobre determinado assunto ou problema.
	Questionário	É uma série ordenada de perguntas que devem ser respondidas por escrito pelo informante. O questionário deve ser objetivo, limitado em extensão e estar acompanhado de instruções. As instruções devem esclarecer o propósito de sua aplicação, ressaltar a importância da colaboração do informante e facilitar o preenchimento.
	Observação na Vida Real	Quando se utilizam os sentidos na obtenção de dados de determinados aspectos da realidade. Na observação na vida real os registros de dados são feitos à medida que ocorrem.

Tabela 4 – Descrição das classificações adotadas de pesquisa, conceitos extraídos de [Silva e Menezes \(2005\)](#) parte 2.

Segundo ([YIN, 2001](#)), o estudo de caso é um conjunto de procedimentos pré-especificados para se obter uma investigação empírica que investiga um fenômeno contemporâneo dentro de seu contexto da vida real, especialmente quando os limites entre o fenômeno e o contexto não estão claramente definidos. Uma grande vantagem do estudo de caso é a sua capacidade de lidar com uma ampla variedade de evidências - documentos, artefatos, entrevistas e observações - além do que pode estar disponível no estudo histórico convencional. Além disso, em algumas situações, como na observação participante, pode ocorrer manipulação informal.

([WOHLIN et al., 2012](#)) fraciona o estudo de caso em cinco passos. Nesta pesquisa, o estudo de caso compreende os passos: Planejar o Estudo de Caso; Coletar Dados; Analisar Dados Coletados e Compartilhar os Resultados. Portanto, os passos Projetar o Estudo de Caso e Preparar a Coleta de Dados definidas por [Wohlin et al. \(2012\)](#) foram agrupadas no passo Planejar o estudo de caso como na Figura 2.

O passo Planejar o Estudo de Caso consiste na determinação do objetivo e da questão de pesquisa, da escolha da metodologia de pesquisa, da definição das fases da pesquisa, da definição dos procedimentos de pesquisa, do protocolo, das técnicas de coleta de dados e da proposta do trabalho final.

No passo Coletar Dados são executados os procedimentos de pesquisa e as técnicas de coletas de dados a seguir:

- Pesquisa Bibliográfica: pesquisa realizada a partir de livros, dissertações e trabalhos relacionados à área de pesquisa;

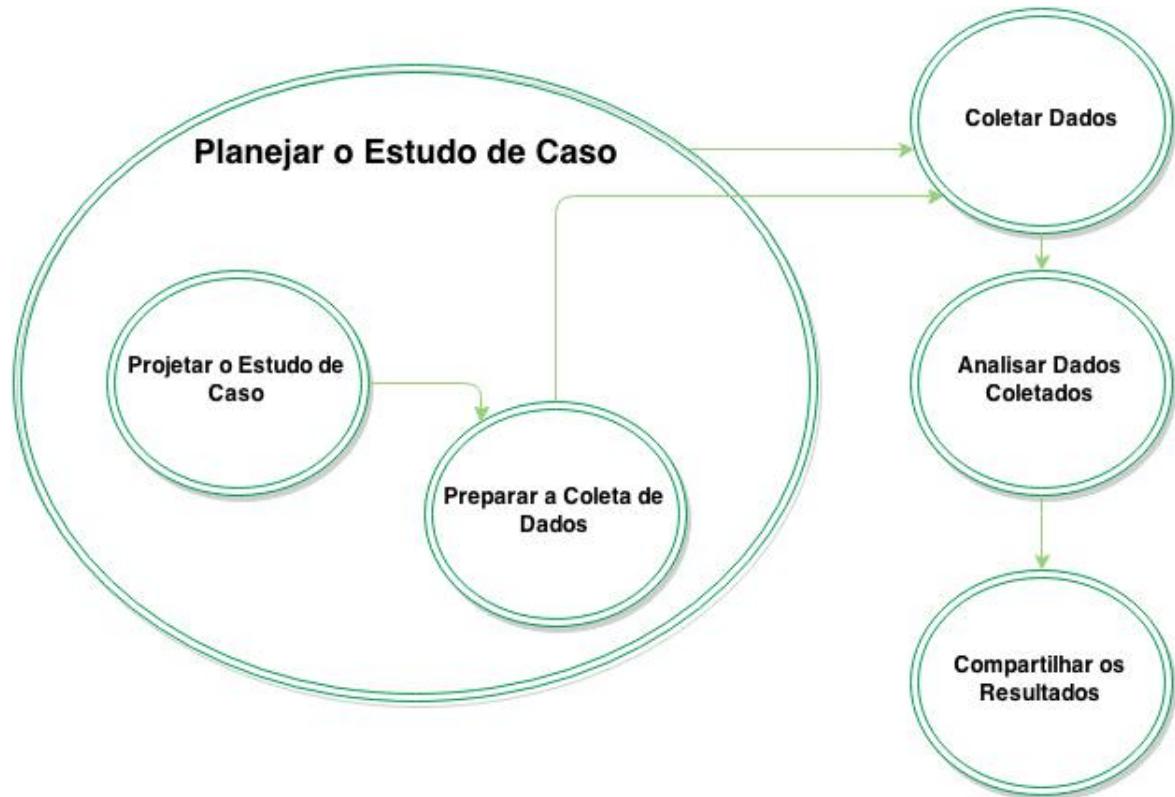


Figura 2 – Passos do Estudo de Caso

- Pesquisa Documental: pesquisa realizada a partir de documentos publicados por organizações públicas;
- Estudo de Caso: utilizar um estudo real de uma organização pública brasileira;
- Entrevistas: dados serão coletados por meio de entrevistas, além de questionário, para incremento do estudo de caso;
- Documentos: coleta de dados dos documentos dos processos fornecidos pelo Órgão público do estudo de caso será realizada para coleta de dados para análise;
- Observação na Vida Real: coleta de dados a partir da observação dentro do Órgão Público;

O passo Analisar Dados Coletados é onde os dados coletados serão analisados e interpretados. A análise compreende tanto a análise quantitativa quanto a análise qualitativa.

Por fim, o passo Compartilhar os resultados diz respeito expor os resultados de forma adequada para o leitor alvo.

1.6 Metodologia de Desenvolvimento

A metodologia de desenvolvimento utilizada no Trabalho de Conclusão de Cursos 1 foi baseada em reuniões quinzenais junto ao orientador cujo objetivo era discutir sobre o tema, levantar as necessidades, atribuir atividades para suprir as necessidades e validar os resultados das atividades já realizadas. Ao final do Trabalho de Conclusão de Cursos 1 foi elaborado um cronograma com as macro atividades para serem realizadas durante a o Trabalho de Conclusão de Cursos 2. Contudo, após a defesa do Trabalho de Conclusão de Cursos 1, houve um realinhamento sobre os objetivos esperados, o que resultou a adequação do planejamento inicial.

Objetivando maximizar a produtividade, as reuniões passaram a ser semanais e passamos a utilizar a ferramenta *kanbanFlow* para gerir tarefas usando o método *Kanban*. O *Kanban* foi dividido em 4 colunas como mostra a Figura 3 :

- **Main Backlog:** Onde ficam todas as atividades que devem ser realizadas durante todo o Trabalho de Conclusão de Cursos 2.
- **Sprint Backlog:** Onde ficam as atividades que devem ser realizadas durante a semana (*Sprint*).
- **In Progress:** Onde ficam as atividades que estão em andamento.
- **Done:** Onde ficam as atividades finalizadas.

The screenshot shows the KanbanFlow application interface with four columns:

- Main Backlog:** Contains tasks like "elaborar um manual de uso formal" and "escrever sessão sobre metodologia de desenvolvimento".
- Sprint Backlog:** Contains tasks like "Escrever sobre erro quadrático médio no tcc" and "Documentar alterações no modelo existente".
- In progress:** Contains tasks like "redefinir protocolo do estudo de caso do tcc1" and "revisar parte escrita do tcc".
- Done:** Contains tasks like "alterar resumo e abstract", "estudar ferramenta r", and "gerar os erro quadrático médio entre as analises do pmr e da solução de dw, por tipo e por release".

Figura 3 – Estrutura do *Kanban*.

O Apêndice D contém um relatório que demonstra as atividades realizadas durante todo o Trabalho de Conclusão de Cursos 2 e as horas gastas por atividade divididas pelas colunas descritas acima.

1.7 Organização do Trabalho

Esse trabalho está dividido em 6 capítulos:

- **Capítulo 2 - Métricas de Software:** Capítulo responsável pela explicação teórica a respeito do que são métricas de código e verificadores de erros e como elas foram utilizadas no desenvolvimento da solução que esse trabalho busca analisar.
- **Capítulo 3 - Contratações de Fornecedores de Desenvolvimento de Software:** Capítulo responsável por introduzir as principais informações referentes à Contratação de Fornecedores de Desenvolvimento de Software. Para isso, o capítulo é iniciado com uma visão geral sobre a importância das contratações e suas principais características. Posteriormente, é apresentado um resumo da Instrução Normativa 04.
- **Capítulo 4 - Data Warehouse:** Nesse capítulo, serão apresentados conceitos teóricos sobre *Data Warehousing*, assim como a maneira como foi desenvolvido e como funciona o ambiente de *Data Warehouse* para armazenamento de métricas de código fonte. Também trata a importância da visualização dos dados e quais as ferramentas foram utilizadas para a implantação do *dashboard*.
- **Capítulo 5 - Estudo de Caso:** É apresentado o projeto do estudo de caso resultante do passo planejar Estudo de Caso, buscando demonstrar o escopo e elaborar um protocolo para o estudo de caso realizado. Elementos de pesquisa foram identificados e explicados, como o problema a ser resolvido, os objetivos a serem alcançados no estudo de caso, quais os métodos de coleta, sua execução e a análise dos dados.
- **Capítulo 6 - Conclusão:** Capítulo responsável por apresentar as conclusões, limitações do trabalho e trabalhos futuros.

2 Métricas de Software

2.1 Processo de Medição

Segundo o [SOFTEX \(2013b\)](#), a Medição apoia a gerência e a melhoria de processo e de produto, sendo um dos principais meios para gerenciar as atividades do ciclo de vida do trabalho e avaliar a viabilidade dos planos de trabalho. O propósito da Medição é coletar e analisar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus trabalhos, de forma a apoiar o efetivo gerenciamento e demonstrar objetivamente a qualidade dos produtos ([ISO/IEC 15939, 2008](#)).

Ainda segundo o [SOFTEX \(2013b\)](#), entende-se por método de medição uma sequência lógica de operações, descritas genericamente, usadas para quantificar um atributo com respeito a uma escala especificada. Esta escala pode ser nominal, ordinal ou de razão (de proporção), bem como definida em um intervalo:

- **Nominal:** A ordem não possui significado na interpretação dos valores. ([MEIRELLES, 2013](#))
- **Ordinal:** A ordem dos valores possui significado, porém a distância entre os valores não. ([MEIRELLES, 2013](#))
- **Intervalo:** A ordem dos valores possui significado e a distância entre os valores também. Porém, a proporção entre os valores não necessariamente possui significado. ([MEIRELLES, 2013](#))
- **Racional:** Semelhante a medida com escala do tipo intervalo, porém, a proporção possui significado. ([MEIRELLES, 2013](#))

A [ISO/IEC 15939 \(2002\)](#) também divide o processo de medição em dois métodos diferentes, que se distinguem pela natureza do que é quantificado:

- **Subjetiva:** Quantificação envolvendo julgamento de um humano.
- **Objetiva:** Quantificação baseada em regras numéricas. Essas regras podem ser implementadas por um humano.

2.2 Definição das métricas de software

As métricas de software são medidas resultantes da medição do produto ou do processo do *software* pelo qual é desenvolvido, sendo que o produto de *software* deve ser visto como um objeto abstrato que se desenvolveu a partir de uma declaração inicial da

necessidade de um sistema para um software finalizado, incluindo o código-fonte e as várias formas de documentação produzidas durante o desenvolvimento (MILLS, 1999). Estas medidas resultantes podem ser estudas para serem utilizadas para medir a produtividade e a qualidade do produto.

Mills (1999) classifica as métricas de software como métricas de produtos ou métricas de processo, essa divisão pode ser vista na Figura 4.

- **Métricas de produtos:** são as medidas do produto de software em qualquer fase do seu desenvolvimento, a partir dos requisitos do sistema. Métricas de produto podem medir a complexidade da arquitetura do software, o tamanho do programa (código-fonte), ou o número de páginas de documentos produzidos.
- **Métricas de processo:** são as medidas do processo de desenvolvimento de software, como o tempo de desenvolvimento global, o tipo de metodologia utilizada, ou o nível médio da experiência da equipe de programação.

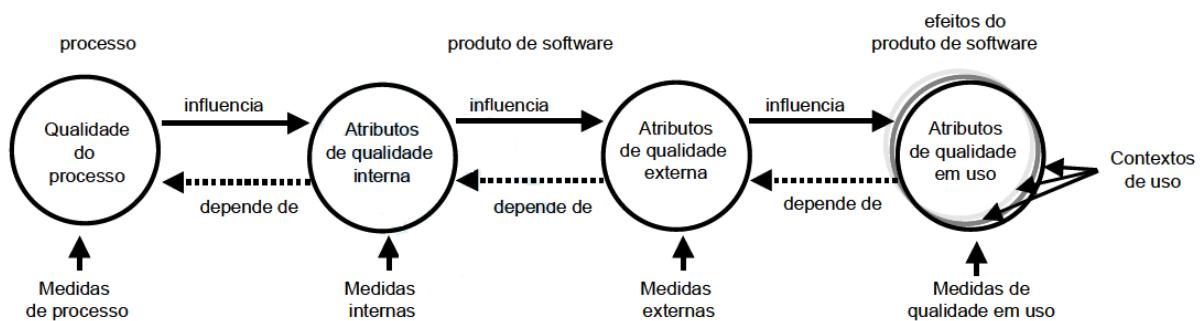


Figura 4 – Modelo de Qualidade do Produto da ISO/IEC 9126 (2001)

Na figura 4 também pode ser notada a classificação das métricas de acordo com os diferentes tipos de medição, refletindo como as métricas influenciam nos contextos em que elas estão envolvidas. A qualidade do produto de software pode ser avaliada medindo-se os atributos internos (tipicamente medidas estáticas de produtos intermediários), os atributos externos (tipicamente pela medição do comportamento do código quando executado) ou os atributos de qualidade em uso (ISO/IEC 9126, 2001).

- **Métricas de qualidade interna:** Aplicadas em um produto de software não executável, como código fonte. Oferecem aos usuários, desenvolvedores ou avaliadores o benefício de poder avaliar a qualidade do produto antes que ele seja liberado para o uso.
- **Métricas de qualidade externa:** Aplicadas a um produto de software executável, medindo o comportamento do sistema, do qual o software é uma parte, por meio de testes, operação ou mesmo observação. Oferecem aos usuários, desenvolvedores ou avaliadores o benefício de poder avaliar a qualidade do produto durante seu processo de teste ou operação.

- **Métricas de qualidade em uso:** Aplicadas para medir o quanto um produto atende as necessidades de um usuário para que sejam atingidas metas especificadas como eficácia, produtividade, segurança e satisfação.

2.3 Métricas de código fonte

A definição de código-fonte segundo a SCAM é qualquer descrição executável de um sistema de software sendo incluído código de máquina, linguagens de alto nível e por representações gráficas executáveis ([HARMAN, 2010](#)).

Segundo [Mills \(1999\)](#), a maior parte do trabalho inicial em métricas de produto analisou as características do código-fonte. A partir da experiência com métricas e modelos, tornou-se cada vez mais evidente que as informações de métricas obtidas anteriormente do ciclo de desenvolvimento pode ser de grande valor no controle do processo e dos resultados, o que sucedeu uma série de trabalhos tratando sobre o tamanho ou complexidade do software.

Neste capítulo, serão evidenciadas, em duas categorias as métricas de código-fonte que serão utilizadas neste trabalho de conclusão de curso, métricas de tamanho e complexidade e métricas de orientação a objetos. Estas métricas são objetivas e serão calculadas a partir da análise estática do código-fonte de um software.

2.3.1 Métricas de tamanho e complexidade

Cada produto do desenvolvimento de software é uma entidade física, como tal, pode ser descrito em termos de tamanho. Desde outros objetos físicos são facilmente mensuráveis (em comprimento, volume, massa, ou outra medida padrão), medir o tamanho do software deve ser relativamente simples e coerente de acordo com os princípios da teoria da medição. No entanto, na prática, a medição de tamanho apresenta grandes dificuldades ([FENTON; PFLEEGER, 1998](#)).

Segundo [Honglei, Wei e Yanan \(2009\)](#), as métricas de complexidade de *software* pertencem as principais medições de software e é o principal método para assegurar a qualidade do *software*. Quanto menor a complexidade dos programas, melhor eles são. Assim, as métricas de complexidade também podem ser usadas para prever os defeitos ou erros. A seguir são apresentadas algumas métricas de tamanho e complexidade.

- **LOC (Lines of Code):** Métrica simples em que são contadas as linhas executáveis de um código, desconsiderando linhas em branco e comentários. ([KAN, 2002](#))
- **ACCM (Average Cyclomatic Complexity per Method):** Mede a complexidade do programa, podendo ser representada através de um grafo de fluxo de controle. ([MC-CABE, 1976](#))

- **AMLOC** (*Average Method Lines of Code*): Indica a distribuição de código entre os métodos. Quanto maior o valor da métrica, mais pesado é o método. É preferível que haja muitos métodos com pequenas operações do que um método grande e de entendimento complexo. ([MEIRELLES, 2013](#))

2.3.2 Métricas de Orientação a Objetos

Segundo [Budd \(2002\)](#), a programação orientada a objetos tornou-se extremamente popular nos últimos anos, inúmeros livros e edições especiais de revistas acadêmicas e comerciais têm surgido sobre o assunto. A julgar por essa atividade frenética, a programação orientada a objeto está sendo recebida com ainda mais entusiasmo do que vimos em ideias revolucionárias mais antigas, tais como programação estruturada "ou sistemas especialistas."

Há uma série de razões importantes pelas quais, nas últimas duas décadas, a programação orientada a objetos tornou-se o paradigma de programação dominante. A programação orientada a objeto possui uma ótima escala, desde o mais trivial dos problemas para a maioria das tarefas complexas. Ele fornece uma forma de abstração que reflete em técnicas de como pessoas usam para resolver problemas em sua vida cotidiana. E para a maioria das linguagens orientadas a objeto dominante há um número cada vez maior de bibliotecas que auxiliam no desenvolvimento de aplicações para muitos domínios ([BUDD, 2002](#)).

Serão adotadas neste trabalho as seguintes métricas de orientação a objetos:

- **ACC** (*Afferent Connections per Class* - Conexões Aferentes por Classe): Mede a conectividade entre as classes. Quanto maior a conectividade entre elas, maior o potencial de impacto que uma alteração pode gerar. ([MEIRELLES, 2013](#))
- **ANPM** (*Average Number of Parameters per Method* - Média do Número de Parâmetros por Método): Indica a média de parâmetros que os métodos possuem. Um valor muito alto para quantidade de parâmetros pode indicar que o método está tendo mais de uma responsabilidade. ([BASILI; ROMBACH, 1987](#))
- **CBO** (*Coupling Between Objects* - Acoplamento entre Objetos): Essa é uma métrica que diz respeito a quantas outras classes dependem de uma classe. É a conta das classes às quais uma classe está acoplada. Duas classes estão acopladas quando métodos de uma delas utilizam métodos ou variáveis de outra. Altos valores dessa métrica aumentam a complexidade e diminuem a manutenibilidade. ([LAIRD, 2006](#)).
- **DIT** (*Depth of Inheritance Tree* - Profundidade da Árvore de Herança): Responsável por medir quantas camadas de herança compõem uma determinada hierarquia de classes ([LAIRD, 2006](#)). Segundo [Meirelles \(2013\)](#), quanto maior o valor de DIT,

maior o número de métodos e atributos herdados, portanto, maior a complexidade.

- **LCOM4** (*Lack of Cohesion in Methods* - Falta de Coesão entre Métodos): A coesão de uma classe é indicada por quão próximas as variáveis locais estão relacionadas com variáveis de instância locais. Alta coesão indica uma boa subdivisão de classes. A LCOM mede a falta de coesão através dissimilaridade dos métodos de uma classe pelo emprego de variáveis de instância. (KAN, 2002). A métrica LCOM foi revista e passou a ser conhecida como LCOM4, sendo necessário para seu cálculo a construção de um gráfico não-orientado em que os nós são os atributos e métodos de uma classe. Para cada método deve haver uma aresta entre ele e outro método ou variável. O valor da LCOM4 é o número de componentes fracamente conectados a esse gráfico (MEIRELLES, 2013)
- **NOC** (*Number of Children* - Número de Filhos): É o número de sucessores imediatos, (portanto filhos) de uma classe. Segundo Laird (2006), altos valores indicam que a abstração da super classe foi diluída e uma reorganização da arquitetura deve ser considerada.
- **NOM** (*Number of Methods* - Número de Métodos): Indica a quantidade de métodos de uma classe, medindo seu tamanho. Classes com muitos métodos são mais difíceis de serem reutilizadas pois são propensas a serem menos coesas. (MEIRELLES, 2013)
- **NPA** (*Number of Public Attributes* - Número de Atributos Públicos): Mede o encapsulamento de uma classe, através da medição dos atributos públicos. O número ideal para essa métrica é zero (MEIRELLES, 2013)
- **RFC** (*Response For a Class* - Respostas para uma Classe): Kan (2002) define essa métrica como o número de métodos que podem ser executados em respostas a uma mensagem recebida por um objeto da classe.

2.4 Configurações de qualidade para métricas de código fonte

Meirelles (2013) apresentou uma abordagem para a observação e interpretação das métricas de código-fonte em sua tese de doutorado, onde estas métricas foram estudadas através de suas distribuições estatísticas e associações. Foram avaliados a distribuição e correlações dos valores das métricas de 38 projetos de software livre, sendo coletados e analisados valores para cada métrica em mais de 344.872 classes e módulos. Segundo o próprio Meirelles (2013), entre as principais contribuições de sua tese foi a análise detalhada, em relação ao comportamento, valores e estudos de caso, de 15 métricas de código-fonte.

Dentre os objetivos científicos da tese de Meirelles (2013) o mais crucial para este

trabalho de conclusão de curso é o objetivo OC4, que trata das distribuições estatísticas dos valores de métricas em 38 projetos de software livre, a fim de compreender qual a abordagem estatística mais informativa para o monitoramento dessas métricas, bem como observar os valores frequentes para essas métricas, de forma a servirem de referência para projetos futuros.

[Meirelles \(2013\)](#), para conduzir o seu estudo quantitativo, utilizou-se da técnica de estatística descritiva: percentil. O percentil separa os dados em cem grupos que apresentam o mesmo número de valores, sendo a definição do percentil de ordem $p \times 100 (0 < p < 1)$, em um conjunto de dados de tamanho n , é o valor da variável que ocupa a posição $p(n+1)$ do conjunto de dados ordenados. O percentil de ordem p (ou p -quartil) deixa $p \times 100\%$ das observações abaixo dele na amostra ordenada. O percentil 25 recebe o nome de primeiro quartil, o percentil 50 de segundo quartil ou mediana, e o percentil 75 de terceiro quartil. Uma das hipóteses que [Meirelles \(2013\)](#) utiliza é que só a partir do terceiro quartil será possível obter dados informativos sobre as métricas.

Após a aplicação da técnica percentil na série de dados das métricas de código-fonte obtidos da análise estática do código-fonte dos projetos de software livre, tabelas apresentando os valores percentis de cada métrica nos 38 projetos avaliados foram criadas, como a Tabela 5, que mostra os percentis para a métrica NOC.

	Mín	1%	5%	10%	25%	50%	75%	90%	95%	99%	Máx
Eclipse	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,0	3,0	10,0	243,0
Open JDK8	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	3,0	9,0	301,0
Ant	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	3,0	12,0	162,0
Checkstyle	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,0	5,0	144,0
Eclipse Metrics	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,0	11,0	24,0
Findbugs	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	2,0	5,0	55,0
GWT	0,0	0,0	0,0	0,0	0,0	0,0	0,0	39,0	1,0	8,0	398,0
Hudson	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	2,0	7,0	23,0
JBoss	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	2,0	6,0	256,0
Kalibro	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	2,0	9,0	101,0
Log4J	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	3,0	9,0	23,0
Netbeans	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	2,0	6,0	989,0
Spring	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,0	3,0	7,0	175,0
Tomcat	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	2,0	8,0	82,0

Tabela 5 – Percentis para métrica NOC em projetos Java extraídos de [Meirelles \(2013\)](#)

Através dos resultados obtidos para cada métrica, [Meirelles \(2013\)](#) observou que era possível identificar valores frequentes analisando os percentis. Na tabela 5, por exemplo, considerando o projeto **Open JDK8** como referência, foram observados os intervalos de valores de 0, como muito frequente, 1 e 2 como frequente, 3 como pouco frequente e acima de 3 como não frequente ([MEIRELLES, 2013](#)).

Rêgo (2014), observando o trabalho de análise de percentis de Meirelles (2013), percebeu que é possível utilizar os intervalos de frequência obtidos como uma evidência empírica de qualidade do código-fonte. Rêgo (2014) também renomeou os intervalos de frequência obtidos por Meirelles (2013), como na Tabela 6, a fim de facilitar a interpretação de métricas de código-fonte.

Intervalo de Frequência	Intervalo Qualitativo
Muito Frequentes	Excelente
Frequente	Bom
Pouco Frequentes	Regular
Não Frequentes	Ruim

Tabela 6 – Nome dos Intervalos de Frequência extraídos de Rêgo (2014)

Rêgo (2014) observando a diferença dos valores das métricas entre os projetos analisados e na tentativa de diminuir tamanha diferença, considerou dois cenários. Foram utilizado dois produtos de *software* como referências para cada uma das métricas na linguagem de programação Java. Em um primeiro cenário, foi analisado o *Open JDK8*, software que contia os menores valores percentis para as métricas. Já em um segundo cenário, foi considerado o Tomcat, contendo os valores percentis mais altos. A tabela 7 mostra o resultado desta análise:

Métrica	Intervalo Qualitativo	OpenJDK8 Metrics	Tomcat Metrics
LOC	Excelente	[de 0 a 33]	[de 0 a 33]
	Bom	[de 34 a 87]	[de 34 a 105]
	Regular	[de 88 a 200]	[de 106 a 276]
	Ruim	[acima de 200]	[acima de 276]
ACCM	Excelente	[de 0 a 2,8]	[de 0 a 3]
	Bom	[de 2,9 a 4,4]	[de 3,1 a 4,0]
	Regular	[de 4,5 a 6,0]	[de 4,1 a 6,0]
	Ruim	[acima de 6]	[acima de 6]
AMLOC	Excelente	[de 0 a 8,3]	[de 0 a 8]
	Bom	[de 8,4 a 18]	[de 8,1 a 16,0]
	Regular	[de 19 a 34]	[de 16,1 a 27]
	Ruim	[acima de 34]	[acima de 27]
ACC	Excelente	[de 0 a 1]	[de 0 a 1,0]
	Bom	[de 1,1 a 5]	[de 1,1 a 5,0]
	Regular	[de 5,1 a 12]	[de 5,1 a 13]
	Ruim	[acima de 12]	[acima de 13]
ANPM	Excelente	[de 0 a 1,5]	[de 0 a 2,0]
	Bom	[de 1,6 a 2,3]	[de 2,1 a 3,0]
	Regular	[de 2,4 a 3,0]	[de 3,1 a 5,0]
	Ruim	[acima de 3]	[acima de 5]
CBO	Excelente	[de 0 a 3]	[de 0 a 2]
	Bom	[de 4 a 6]	[de 3 a 5]
	Regular	[de 7 a 9]	[de 5 a 7]
	Ruim	[acima de 9]	[acima de 7]
DIT	Excelente	[de 0 a 2]	[de 0 a 1]
	Bom	[de 3 a 4]	[de 2 a 3]
	Regular	[de 5 a 6]	[de 3 a 4]
	Ruim	[acima de 6]	[acima de 4]
LCOM4	Excelente	[de 0 a 3]	[de 0 a 3]
	Bom	[de 4 a 7]	[de 4 a 7]
	Regular	[de 8 a 12]	[de 8 a 11]
	Ruim	[acima de 12]	[acima de 11]
NOC	Excelente	[0]	[1]
	Bom	[1 a 2]	[1 a 2]
	Regular	[3]	[3]
	Ruim	[acima de 3]	[acima de 3]
NOM	Excelente	[de 0 a 8]	[de 0 a 10]
	Bom	[de 9 a 17]	[de 11 a 21]
	Regular	[de 18 a 27]	[de 22 a 35]
	Ruim	[acima de 27]	[acima de 35]
NPA	Excelente	[0]	[0]
	Bom	[1]	[1]
	Regular	[de 2 a 3]	[de 2 a 3]
	Ruim	[acima de 3]	[acima de 3]
RFC	Excelente	[de 0 a 9]	[de 0 a 11]
	Bom	[de 10 a 26]	[de 12 a 30]
	Regular	[de 27 a 59]	[de 31 a 74]
	Ruim	[acima de 59]	[acima de 74]

Tabela 7 – Configurações para os Intervalos das Métricas para Java extraídas de Rêgo (2014)

2.5 Cenários de limpeza

Segundo [Marinescu \(2005\)](#), a utilização de métricas isoladas dificulta a interpretação de anomalias do código, reduzindo a eficácia da medição. Além disso, o autor ainda afirma que a métrica por si só não contém informação suficiente para motivar uma transformação no código que melhore sua qualidade. Uma das soluções para sanar o problema da interpretação do código através de métricas isoladas é interpretar o código através de cenários de limpeza.

[Machini et al. \(2010\)](#) apresenta uma maneira de interpretar os valores das métricas através de cenários problemáticos e suas possíveis melhorias, para que as métricas possam ser mais facilmente incorporadas no cotidiano dos programadores. [Machini et al. \(2010\)](#) também apresenta um estilo de programação baseado no paradigma da Orientação a Objetos que busca o que denominamos de “Código Limpo”, concebido e aplicado por renomados desenvolvedores de software como Robert C. Martin ([MARTIN, 2008](#)) e Kent Beck ([BECK, 2007](#))

Segundo [Beck \(2007\)](#), um código limpo está inserido em um estilo de programação que busca a proximidade a três valores: expressividade, simplicidade e flexibilidade.

- **Expressividade:** Um código se expressa bem quando alguém que o lê é capaz de compreendê-lo e modificá-lo. [Beck \(2007\)](#) destaca que quando foi necessário modificar um código, ele gastou muito mais tempo lendo o que já havia sido feito do que escrevendo sua modificação.
- **Simplicidade:** Um código é simples quando pode ser facilmente lido, havendo uma redução da quantidade de informação que o leitor deve compreender para fazer alterações. Eliminar o excesso de complexidade faz com que aqueles que estejam lendo o código o entendam mais rapidamente.
- **Flexibilidade:** Capacidade de estender a aplicação alterando o mínimo possível a estrutura já criada.

[Machini et al. \(2010\)](#) apresenta em seu trabalho diversas técnicas para a obtenção de um código limpo. Um resumo destas técnicas são apresentadas nas tabelas 8, 9, 10 e 11.

Técnicas de Limpeza	Descrição	Contribuições	Consequências
Composição de Métodos	Compor os métodos em chamadas para outros rigorosamente no mesmo nível de abstração.	<ul style="list-style-type: none"> Facilidade de entendimento de métodos menores Criação de métodos menores com nomes explicativos 	<ul style="list-style-type: none"> Menos Operações por Método Mais Parâmetros de Classe Mais Métodos na Classe
Métodos Explicativos	Criar um método que encapsule uma operação pouco clara, geralmente associada a um comentário	<ul style="list-style-type: none"> O código cliente do método novo terá uma operação com nome que melhor se encaixa no contexto. 	<ul style="list-style-type: none"> Mais métodos na classe.
Métodos como Condicionais	Criar um método que encapsule uma expressão booleana para obter condicionais mais claras.	<ul style="list-style-type: none"> Facilidade na leitura de condicionais no código cliente. Encapsulamento de uma expressão booleana. 	<ul style="list-style-type: none"> Mais métodos na classe.
Evitar Estruturas Encadeadas	Utilizar a composição de métodos para minimizar a quantidade de estruturas encadeadas em cada método (if, else).	<ul style="list-style-type: none"> Facilidade para a criação de testes. Cada método terá estruturas mais simples e fáceis de serem compreendidas. 	<ul style="list-style-type: none"> Menos Estruturas encadeadas por método (if e else) Benefícios do Uso de Composição de Métodos

Tabela 8 – Conceitos de Limpeza levantados por Machini et al. (2010) e adaptados por Rêgo (2014) parte 1.

Técnicas de Limpeza	Descrição	Contribuições	Consequências
Cláusulas Guarda	Criar um retorno logo no inicio de um método ao invés da criação de estruturas encadeadas com if sem else.	<ul style="list-style-type: none"> Estruturas de condicionais mais simples. Leitor não espera por uma contrapartida do condicional(ex:if sem else). 	<ul style="list-style-type: none"> Menos estruturas encadeadas na classe.
Objeto Método	Criar uma classe que encapsule uma operação complexa simplificando a original(cliente).	<ul style="list-style-type: none"> O código cliente terá um método bastante simples. Nova classe poderá ser refatorada sem preocupações com alterações no código cliente. Nova classe poderá ter testes separados. 	<ul style="list-style-type: none"> Menos operações no método cliente. Menos responsabilidades da classe cliente. Mais classes. Mais acoplamento da classe cliente com a nova classe.
Evitar <i>Flags</i> como Argumentos	Ao invés de criar um método que recebe uma flag e tem diversos comportamentos, criar um método para cada comportamento.	<ul style="list-style-type: none"> Leitor não precisará entender um método com muitos condicionais. Testes de unidade independentes. 	<ul style="list-style-type: none"> Mais métodos na classe.

Tabela 9 – Conceitos de Limpeza levantados por Machini et al. (2010) e adaptados por Rêgo (2014) parte 2.

Técnicas de Limpeza	Descrição	Contribuições	Consequências
Maximizar a Coesão	Quebrar uma classe que não segue o Princípio da Responsabilidade Única: as classes devem ter uma única responsabilidade, ou seja, ter uma única razão para mudar.	<ul style="list-style-type: none"> • Cada classe terá uma única responsabilidade. • Cada classe terá seus testes independentes. • Sem interferências na implementação das responsabilidades. 	<ul style="list-style-type: none"> • Mais Classes • Menos Métodos em cada Classe • Menos Atributos em cada Classe
Objeto como Parâmetro	Localizar parâmetros que formam uma unidade e criar uma classe que os encapsule.	<ul style="list-style-type: none"> • Menor número de parâmetros facilita testes e legibilidade. • Criação de uma classe que poderá ser reutilizada em outras partes do sistema. 	<ul style="list-style-type: none"> • Menos Parâmetros sendo passados para Métodos • Mais Classes
Parâmetros como Variável de Instância	Localizar parâmetro muito utilizado pelos métodos de uma classe e transformá-lo em variável de instância.	<ul style="list-style-type: none"> • Não haverá a necessidade de passar longas listas de parâmetro através de todos os métodos. 	<ul style="list-style-type: none"> • Menos Parâmetros passados pela Classe • Possível diminuição na coesão
Uso de Exceções	Criar um fluxo normal separado do fluxo de tratamento de erros utilizando exceções ao invés de valores de retornos e condicionais.	<ul style="list-style-type: none"> • Clareza do fluxo normal sem tratamento de erros através de valores de retornos e condicionais. 	<ul style="list-style-type: none"> • Menos Estruturas encadeadas.

Tabela 10 – Conceitos de Limpeza levantados por [Machini et al. \(2010\)](#) e adaptados por [Rêgo \(2014\)](#) parte 3.

Técnicas de Limpeza	Descrição	Contribuições	Consequências
Delegação de Tarefa	Transferir um método que utiliza dados de uma classe "B" para a "B".	<ul style="list-style-type: none"> Redução do acoplamento entre classes. Proximidade dos métodos e dados sobre os quais trabalham. 	<ul style="list-style-type: none"> Menos métodos na classe inicial. Mais métodos na classe que recebe o novo método.
Objeto Centralizador	Criar uma classe que encapsule uma operação com alta dependência entre classes.	<ul style="list-style-type: none"> Simplificação da classe cliente. Redução do acoplamento da classe cliente com as demais. Nova classe poderá receber testes e melhorias independentes. 	<ul style="list-style-type: none"> Menos operações no método cliente. Menos responsabilidades da classe cliente. Mais classes. Mais acoplamento da classe cliente com a nova classe.
Uso Excessivo de Herança	Localizar uso excessivo de herança e transformá-lo em agregação simples.	<ul style="list-style-type: none"> Redução do acoplamento entre as classes. 	<ul style="list-style-type: none"> Maior Flexibilidade de Adição de Novas Classes. Menor Acoplamento entre as classes.
Exposição Pública Excessiva	Localizar uso excessivo de parâmetros públicos e transformá-lo em parâmetros privados.	<ul style="list-style-type: none"> Menor Acoplamento entre as classes. 	<ul style="list-style-type: none"> Maior Encapsulamento de Parâmetros.

Tabela 11 – Conceitos de Limpeza levantados por [Machini et al. \(2010\)](#) e adaptados por [Rêgo \(2014\)](#) parte 4.

Após apresentar as técnicas para obtenção de um código limpo, [Machini et al. \(2010\)](#) adota uma abordagem baseada em cenários para identificar trechos de código com características indesejáveis. Os cenários devem possuir um contexto criado a partir de

poucos conceitos de código limpo no qual um pequeno conjunto de métricas é analisado e interpretado através da combinação de seus valores. A ideia principal desta abordagem é facilitar melhorias de implementação e a procura por problemas quanto a limpeza de código através da aproximação dos valores das métricas com os esperados nos contextos de interpretação ([MACHINI et al., 2010](#)).

[Rêgo \(2014\)](#) utiliza a mesma abordagem de cenários que [Machini et al. \(2010\)](#), onde algumas técnicas de limpeza de código apresentadas nas tabelas [8](#), [9](#), [10](#) e [11](#) são correlacionadas com as métricas da Seção [2.3](#). Adicionalmente, utiliza-se a configuração do *Open JDK8* considerando como valores altos os valores obtidos pelos intervalos Regular e Ruim tal como mostrados na Tabela [7](#).

Aproveitando inicialmente os cenários **Classe pouco coesa** e **Interface dos métodos** extraídos de [Machini et al. \(2010\)](#), [Rêgo \(2014\)](#) elaborou mais alguns cenários de limpeza. O resultado dessa atividade pode ser visto na tabela [12](#):

Cenário de Limpeza	Conceito de Limpeza	Características	Recomendações	Forma de Detecção pelas Métricas de Código-Fonte	Padrões de Projeto Associados
Classe Coesa	Pouco Maximização da Coesão	Classe Subdivida em grupos de métodos que não se relacionam	Reducir a subdivisão da Classe	Intervalos Regulares e Ruins de LCOM4, RFC.	<i>Chain of Responsibilities, Mediator, Decorator.</i>
Interface dos Métodos	Objetos como Parâmetro e Parâmetros repassados como Variáveis de Instância	Elevada Média de parâmetros repassados pela Classe	Minimizar o número de Parâmetros.	Intervalos Regulares e Ruins de ANPM.	<i>Facade, Template Method, Strategy, Command, Mediator, Bridge.</i>
Classes com muitos filhos	Evitar Uso Excessivo de Herança	Muitas Sobreescritas de Métodos	Trocar a Herança por uma Agregação.	Intervalos Regulares e Ruins de NOC.	<i>Composite, Prototype, Decorator, Adapter.</i>
Classe com muitos grandes e/ou muitos condicionais	Composição de Métodos, Evitar Estruturas Encadeadas Complexas	Grande Número Efectivo de Linhas de Código	Reducir LOC da Classe e de seus métodos, Reduzir a Complexidade Ciclomática e Quebrar os métodos.	Intervalos Regulares e Ruins de AMLOC, ACCM.	<i>Chain of Responsibilities, Mediator, Flyweight.</i>
Classe com muita Exposição	Parâmetros Privados	Grande Número de Parâmetros Públicos	Reducir o Número de Parâmetros Públicos.	Intervalos Regulares e Ruins de NPA.	<i>Facade, Singleton.</i>
Complexidade Estrutural	Maximização da Coesão	Grande Acoplamento entre Objetos	Reducir a quantidade de responsabilidades dos Métodos.	Intervalos Regulares e Ruins de CBO e LCOM4.	<i>Chain of Responsibilities, Mediator, Strategy.</i>

Tabela 12 – Cenários de Limpeza extraídos de Rêgo (2014)

2.5.1 Verificadores de Erro

Os verificadores estáticos de código, são ferramentas automáticas para a verificação de código, que podem verificar estilos de programação, erros ou ambos. O verificador de erro é uma ferramenta de análise estática voltada para a detecção automática de erros, tomando como base as tendências comuns dos desenvolvedores em atrair defeitos que, em sua maioria, não são visíveis aos compiladores. ([LOURIDAS, 2006](#))

Segundo [Pugh \(2008\)](#), análise Estática de Código é a análise de um sistema de computador que é realizada sem a sua execução. Já a análise realizada com a execução dos programas é conhecida como análise dinâmica. O termo Análise Estática de Código pode se referir à análise automatizada, que é uma das técnicas estáticas de inspeção de software no processo de validação e verificação de software. A possibilidade de automatizar o processo de verificação de programas resultou no desenvolvimento de analisadores estáticos automatizados.

Analisadores estáticos automatizados são ferramentas de software que varrem o texto-fonte. Entretanto os verificadores estáticos de código podem trabalhar diretamente sobre o código-fonte do programa ou trabalhar sobre o código objeto, no caso da linguagem Java, sobre o *bytecode* ([SOMMERVILLE et al., 2008](#)). Segundo [Louridas \(2006\)](#), a intenção da análise estática automatizada é chamar a atenção para anomalias do programa. As anomalias são resultados de erros de programação ou omissões, de tal modo que onde possa ocasionar erros durante a execução do programa é enfatizado, porém nem todas as anomalias são necessariamente defeitos de programa.

[Jelliffe \(2004\)](#), destacou alguns dos benefícios da utilização de analisadores estáticos:

- Encontra erros e códigos de risco;
- Fornece um retorno objetivo aos programadores para ajudá-los a reconhecer onde eles foram precisos ou desatentos;
- Fornece a um líder de projeto uma oportunidade para estudar o código, o projeto e a equipe de uma perspectiva diferente;
- Retira certas classes de defeitos, o que possibilita que a equipe concentre-se mais nas deficiências do projeto.

2.5.2 Ferramentas de verificação de Erro

Além da ferramenta *Analizo*, ferramenta de análise estática de código-fonte, que será apresentada junto ao ambiente de *Data Warehousing* no Capítulo 4, foi adicionado a este ambiente duas ferramentas de verificação de erros, o FindBugs na versão 3.0.1 e o

PMD na versão 5.0.0.

O FindBugs é uma ferramenta de código aberto utilizado pelos desenvolvedores de software para fazer uma inspeção no código de forma automatizada. Esta ferramenta examina as suas classes procurando por possíveis erros em potencial no código durante a fase de desenvolvimento. O FindBugs analisa o código fonte ou mesmo o código objeto, *bytecode* para programas Java. Segundo [Louridas \(2006\)](#), é um popular verificador estático de código para a linguagem Java e considera a ferramenta como um patrocinador na fortificação da qualidade do software. Atualmente, ela pode analisar programas compilados em qualquer versão da linguagem Java.

O PMD é um analisador de código Java que procura em uma base de código por possíveis problemas voltados às más práticas de desenvolvimento, tais como variáveis não utilizadas, código duplicado, trechos de código de elevada complexidade, criação desnecessária de objetos. Segundo ([HOVEMEYER; PUGH, 2004](#)) é uma ferramenta de extrema valia em forçar os desenvolvedores a seguir um estilo de programação e em tornar o código mais fácil para ser entendido pelo desenvolvedores.

As Tabelas [13](#) e [14](#) evidenciam a diferença entre estas três ferramentas que fazem parte do ambiente de *Data Warehousing* que faz parte da solução que será apresentada neste trabalho.

	Findbugs	PMD	Analizo
Versão	3.0.1	5.0.0	1.18
Lisença	Lesser GNU Public License	BSD-style license	GNU General Public License
Propósito	Erros potenciais	Máx Práticas	Extração e cálculo de métricas de código fonte.
Pontos Fortes	-Encontra defeitos reais na maioria das verificações - verificação rápida - Menos de 50% de falsos positivos	Foco em problemas potenciais, possíveis erros, código não utilizado e sobre expressões complicadas no Código fonte Java	Superta diversas métricas de código fonte.
Número de regras	408	234	47
Regra Categorias	-Máx práticas (Bad practice) -Corretude (Correctness) -Erros de concorrência (Multithreaded correctness) - Internacionalização (Internationalization) -Experimental (Experimental)	-Basic rules: Regras básicas gerais. -Braces rules: Regras relacionadas ao uso de chaves. -Code size rules: Regras que avaliam questões relacionadas ao tamanho do código. -Controversial rules: Regras de aplicação geral, mas de aplicação controversa. -Coupling rules: Regras relacionadas ao acoplamento entre objetos e pacotes.	-Global Metrics -Module Metrics

Tabela 13 – Comparativo das ferramentas utilizadas

	<ul style="list-style-type: none"> -Vulnerabilidade de código malicioso (Malicious code vulnerability) -Potenciais problemas de desempenho (Performance) -Segurança (Security) -Código Confuso (Dodgy code) 	<ul style="list-style-type: none"> -Design rules: Regras que avaliam o design do código. -Import statement rules: Regras relacionadas ao uso de import. -Naming rules: Regras que avaliam nomes de variáveis e métodos. -Optimization rules: Regras relacionadas à otimização. -Strict Exception rules: Regras relacionadas ao lançamento e captura de exceções. -String and StringBuffer rules: Regras que verificam o bom uso das classes String e StringBuffer. -Unused code rules: Regras que detectam código não utilizado. 	
--	---	---	--

Tabela 14 – Comparativo das ferramentas utilizadas

2.6 Considerações Finais do Capítulo

Esse capítulo apresentou a fundamentação teórica sobre métricas de código fonte, quais são seus intervalos qualitativos, a maneira como elas foram relacionadas a cenários de limpeza. Também foi apresentada a fundamentação teórica sobre verificadores de erro. O próximo capítulo será responsável por demonstrar uma breve visão acerca da importância da contratação de serviços de TI, tendo em vista ser o assunto de maior importância dentro do contexto de contratos, especificamente no que tange a atividade de aceitação do cliente. Ainda será tratado dos conceitos de contratação de serviços de TI na Instrução Normativa n. 04.

3 Contratações de Fornecedores de Desenvolvimento de Software

O não cumprimento do disposto na legislação de licitações e contratos pode acarretar riscos para a contratação de Tecnologia da Informação, sendo de grande importância o conhecimento da legislação de licitações e contratos para que se possam ser usados como base em processos de contratação de fornecedores de desenvolvimento de software. Assim, neste capítulo será apresentada uma visão sobre a importância da contratação de serviços de TI e de forma resumida, os conceitos de contratação de serviços de TI presentes na Instrução Normativa Nº 04.

3.1 Importância da Contratação de Fornecedores de Desenvolvimento de Software

De acordo com a [ISO/IEC 15939 \(2008\)](#), contrato é o acordo realizado entre duas partes, respaldado pela lei, ou acordo interno similar restrito a uma organização, para o fornecimento de serviços de software ou para o desenvolvimento, produção, operação ou manutenção de um produto de software.

Embora a contratação de serviços de TI tenha papel importante na estratégia organizacional, há muitos riscos que podem frustrar seus resultados. A definição e institucionalização de processos de contratação de serviços de TI, especialmente aqueles relacionados a software, envolvem ações complexas, principalmente no que diz respeito à identificação dos requisitos necessários, a garantia da qualidade dos resultados esperados, os critérios de aceitação, a gestão de mudanças, as transferências de conhecimentos, a legislação pertinente, entre outros. E envolvem também questões de relacionamento entre clientes e fornecedores, o que implica em competências administrativas e jurídicas. Essas complexidades apresentam riscos para as partes envolvidas e, como consequência, é comum a ocorrência de sérios conflitos. Normas e modelos de referência podem ser úteis para resolver esses conflitos ([CRUZ; ANDRADE; FIGUEIREDO, 2011](#)).

Neste trabalho, a aquisição de *software* é o assunto mais relevante dentro do contexto de contratos, mais especificamente da atividade de aceitação pelo cliente. A ([ISO/IEC 15939, 2008](#)) dividi a aquisição de *software* e serviços correlatos em quatro atividades, conforme a Figura 5.

Segundo a [ISO/IEC 15939 \(2008\)](#), o propósito da atividade de aceitação pelo cliente é aprovar o *software* e os serviços correlatos (S&SC) entregues pelo fornecedor

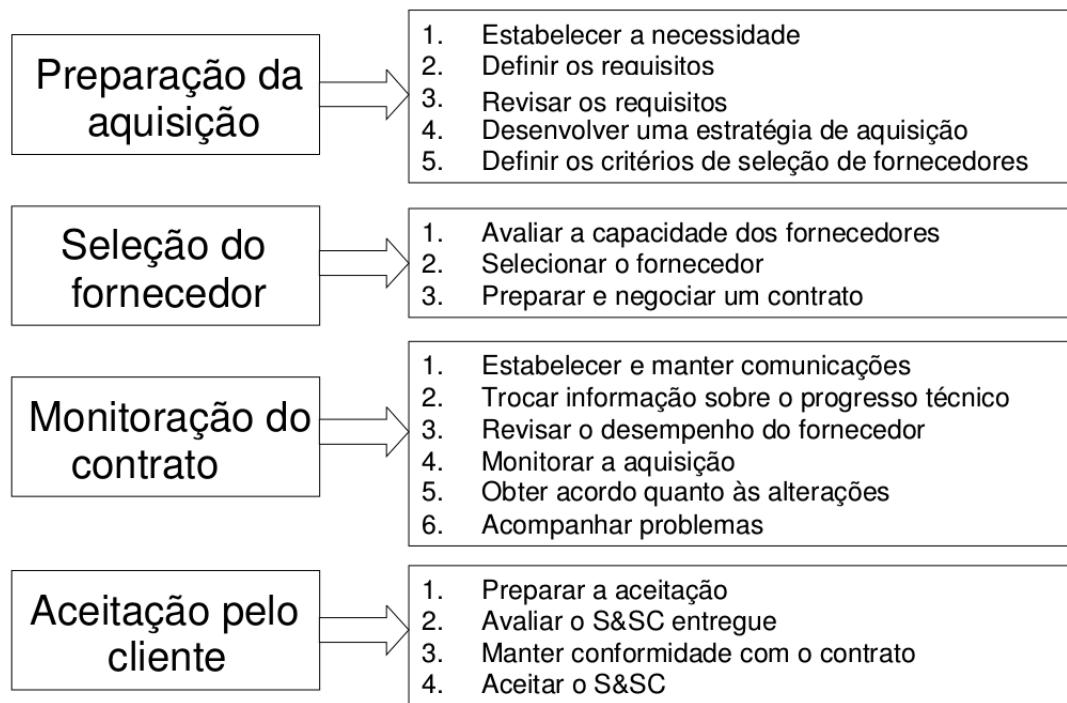


Figura 5 – Atividades de aquisição extraída de [ISO/IEC 15939 \(2008\)](#)

quando todos os critérios de aceitação estiverem satisfeitos. Nesta atividade são refinados os critérios de aceitação que foram definidos no plano de projeto e incorporados no pedido de proposta e no contrato. As avaliações podem ser conduzidas no decorrer do contrato, por uma abordagem envolvendo múltiplas iterações e entregas de produtos, ou por meio de uma entrega única. Os S&SC entregues são analisados para identificar a conformidade aos critérios estabelecidos. As tarefas de avaliação são concebidas de modo a reduzir a interferência com as avaliações executadas pelo fornecedor e a duplicação de esforços de avaliação. Não havendo aprovação do S&SC, e dependendo das cláusulas contratuais, podem ser planejados e implementados ajustes para que o produto seja submetido a uma nova avaliação. Este ciclo ocorre enquanto o produto não é aprovado, ou até que seja definitivamente rejeitado. As tarefas previstas compreendem:

- **Definir critérios de aceitação**
- **Avaliar o produto entregue**
- **Manter conformidade com o contrato**
- **Aceitar o S&SC**

A Instrução Normativa Nº 04 trata do processo de contratação de serviços de TI pela Administração Pública Federal direta, autárquica e fundacional. Esta Instrução Normativa IN (Instrução Normativa) será explicada na próxima seção, onde os artigos mais relevantes no contexto da aquisição de *software* e da aceitação de *software* pelo

cliente serão abordados.

3.2 Instrução Normativa N° 04

Instruções normativas são atos expedidos por autoridades administrativas, normas complementares das leis, dos tratados e das convenções internacionais e dos decretos, e não podem transpor, inovar ou modificar o texto da norma que complementam. As instruções normativas visam regulamentar ou implementar o que está previsto nas leis que, no caso do Brasil, são apreciadas, elaboradas e aprovadas pelo Congresso Nacional, e sancionadas pelo Presidente da República.

A [IN4 \(2014\)](#) dispõe sobre o processo de contratação de Soluções de Tecnologia da Informação pelos órgãos integrantes do Sistema de Administração de Recursos de Tecnologia da Informação e Informática (SISP) do Poder Executivo Federal. Esta IN é a consolidação de um conjunto de boas práticas para Contratação de Solução de TI, que formam o Modelo de Contratações de Soluções de TI (MCTI).

A Instrução Normativa nº 04 [IN4 \(2014\)](#) está dividida em três capítulos:

- **Capítulo 1:** Diz respeito às disposições gerais.
- **Capítulo 2:** Diz respeito ao processo de contratação e é dividido em 3 seções.
 - **Seção 1:** Diz respeito ao Planejamento da Contratação e é dividido em 4 Subseções.
 - **Seção 2:** Diz respeito à Seleção do Fornecedor.
 - **Seção 3:** Diz respeito à Gestão do Contrato e é dividido em 4 Subseções.
- **Capítulo 3:** Apresenta as Disposições Finais.

Os artigos mais relevantes e que serão abordados neste trabalho são os artigos 20 e 34, ambos inseridos no capítulo 2 da referida normativa.

O caput do artigo 20 do [IN4 \(2014\)](#) trata sobre o modelo de gestão do contrato, definido a partir do Modelo de Execução do Contrato, que deverá contemplar as condições para gestão e fiscalização do contrato de fornecimento da Solução de Tecnologia da Informação.

O inciso II versa sobre os procedimentos de teste e inspeção, para fins de elaboração dos Termos de Recebimento Provisório e Definitivo. Ademais, na alínea "a", itens 1,2 e 5 tratam sobre a metodologia, formas de avaliação da qualidade e adequação da solução de Tecnologia da Informação às especificações funcionais e tecnológicas, observando:

- **Item 1:** Definição de mecanismos de inspeção e avaliação da Solução, a exemplo

de inspeção por amostragem ou total do fornecimento de bens ou da prestação de serviços.

- **Item 2:** Adoção de ferramentas, computacionais ou não, para implantação e acompanhamento dos indicadores estabelecidos.
- **Item 5:** Garantia de inspeções e diligências, quando aplicáveis, e suas formas de exercício.

O inciso III versa sobre a fixação dos valores e procedimentos para retenção ou glosa no pagamento, sem prejuízo das sanções cabíveis, que só deverá ocorrer quando a contratada incindir nas alíneas "a" e "b" do mencionado inciso.

- **alíneas "a":** não atingir os valores mínimos aceitáveis fixados nos Critérios de Aceitação, não produzir os resultados ou deixar de executar as atividades contratadas; ou
- **alíneas "b":** deixar de utilizar materiais e recursos humanos exigidos para fornecimento da solução de tecnologia da informação, ou utilizá-los com qualidade ou quantidade inferior à demandada.

Por fim, no inciso IV trata sobre a definição clara e detalhada das sanções administrativas observadas, principalmente, nas alíneas "b" e "c".

- **alíneas "b":** proporcionalidade das sanções previstas ao grau do prejuízo causado pelo descumprimento das respectivas obrigações.
- **alíneas "c":** as situações em que advertências ou multas serão aplicadas, com seus percentuais correspondentes, que obedecerão a uma escala gradual para as sanções recorrentes.

Outrossim, o artigo 34 da referida normativa trata sobre o monitoramento da execução deverá observar o disposto no Plano de Fiscalização da contratada e o disposto no modelo de gestão do contrato, observando os seguintes incisos:

- **II:** Avaliação da qualidade dos serviços realizados ou dos bens entregues e justificativas, a partir da aplicação das Listas de Verificação e de acordo com os Critérios de Aceitação definidos em contrato, a cargo dos Fiscais Técnico e Requisitante do Contrato.
- **III:** Identificação de não conformidade com os termos contratuais, a cargo dos Fiscais Técnico e Requisitante do Contrato.
- **VI:** Encaminhamento das demandas de correção à contratada, a cargo do Gestor do Contrato ou, por delegação de competência, do Fiscal Técnico do Contrato.
- **VII:** Encaminhamento de indicação de glosas e sanções por parte do Gestor do

Contrato para a Área Administrativa.

3.3 Considerações finais do capítulo

Neste capítulo tratou-se acerca da lei de licitações e contratos, evidenciando a importância da contratação de fornecedores de desenvolvimento de software, restringindo o estudo na análise da instrução normativa n. 04 com suas devidas especificações.

Conforme as atividades de aquisição da ([ISO/IEC 15939, 2008](#)), a contratante deve avaliar o S&SC entregue pela contratada, sendo a monitoração de métricas de código-fonte parte da avaliação do software entregue, assim devendo ser monitorada através da observância disposto no Plano de Fiscalização da contratada e o disposto no modelo de gestão do contrato, que deverá conter as condições para a gestão e fiscalização do contrato de fornecimento da solução de Tecnologia da Informação.

No próximo capítulo será apresentado a fundamentação teórica acerca do *Data Warehouse*, assim como sua utilização no monitoramento de métricas, defeitos e violações.

4 Data Warehouse

Neste capítulo será apresentada a fundamentação teórica sobre *Data Warehouse* e como utiliza-lo para o monitoramento de métricas, *bugs* e violações. Também será apresentado o ambiente de *Data Warehouse* utilizado na solução proposta por Rêgo (2014), como ela foi desenvolvida e os incrementos no qual este trabalho buscou aumentar o valor da aferição da qualidade.

4.1 Definição

Na década de 80 as organizações perceberam a importância de não apenas usar dados para propósitos operacionais, mas também para derivar a inteligência por trás deles. Essa inteligência não só justificaria as decisões passadas, mas também ajudaria na tomada de decisões para o futuro. O termo *Business Intelligence* tornou-se cada vez mais popular, e foi durante o final dos anos 1980 que pesquisadores da IBM, Barry Devlin e Paul Murphy, desenvolveram o conceito de *Business data warehouse*. A partir que aplicações de *business intelligence* foram surgindo, foi rapidamente verificado que os dados de bancos transacionais tinham que primeiramente ser transformados e armazenados em outros bancos de dados com um esquema específico para poderem derivar de sua inteligência. Esta base de dados poderia ser usada para arquivamento, e seria maior em tamanho do que as bases de dados transacionais, mas seu *design* seria ideal para executar relatórios que permitem as grandes organizações planejarem e tomarem decisões de forma proativa. Este banco de dados, normalmente armazenando as atividades realizadas no passado e no presente das organizações, foi chamado de *Data Warehouse* (SHARMA, 2011).

Para Inmon (2002), *Data Warehouse* é uma coleção de dados que tem como característica ser orientada a assunto, integrada, não volátil e temporal. Por orientação a assunto, podemos entender como um foco em algum aspecto específico da organização. O fato do ambiente ser integrado remete ao fato dele ser alimentado com dados que têm como origem múltiplas fontes, integrando esses dados de maneira a construir uma única orientação. Como um conjunto não volátil e temporal de dados, é entendido que a informação carregada remete a um determinado momento da aplicação, possibilitando assim acesso a diferentes intervalos de tempo, não havendo como modificá-los atualizando em tempo real.

Segundo Rocha (2000), *Data Warehousing* é a infra-estrutura tecnológica de *hardware* e *software* para a atividade de análise gerencial. Agora que sabemos o que é o *data warehouse* e *Data Warehousing*, serão mostrado os componentes que compõem um ambiente completo de *data warehousing*. É importante entender como os componentes

funcionam individualmente antes de começarem a ser combinados para se criar um *data warehouse*. Cada componente do armazém tem uma função específica. É preciso aprender a importância estratégica de cada componente e como manuseá-los efetivamente para fazer uso do *data warehousing*. Uma das maiores ameaças ao sucesso no *data warehousing* é confundir os papéis e as funções dos componentes (KIMBALL; ROSS, 2002). A Figura 6 descreve uma arquitetura geral de um ambiente de *Data Warehousing*, os componentes do ambiente serão esclarecidos no decorrer deste capítulo.

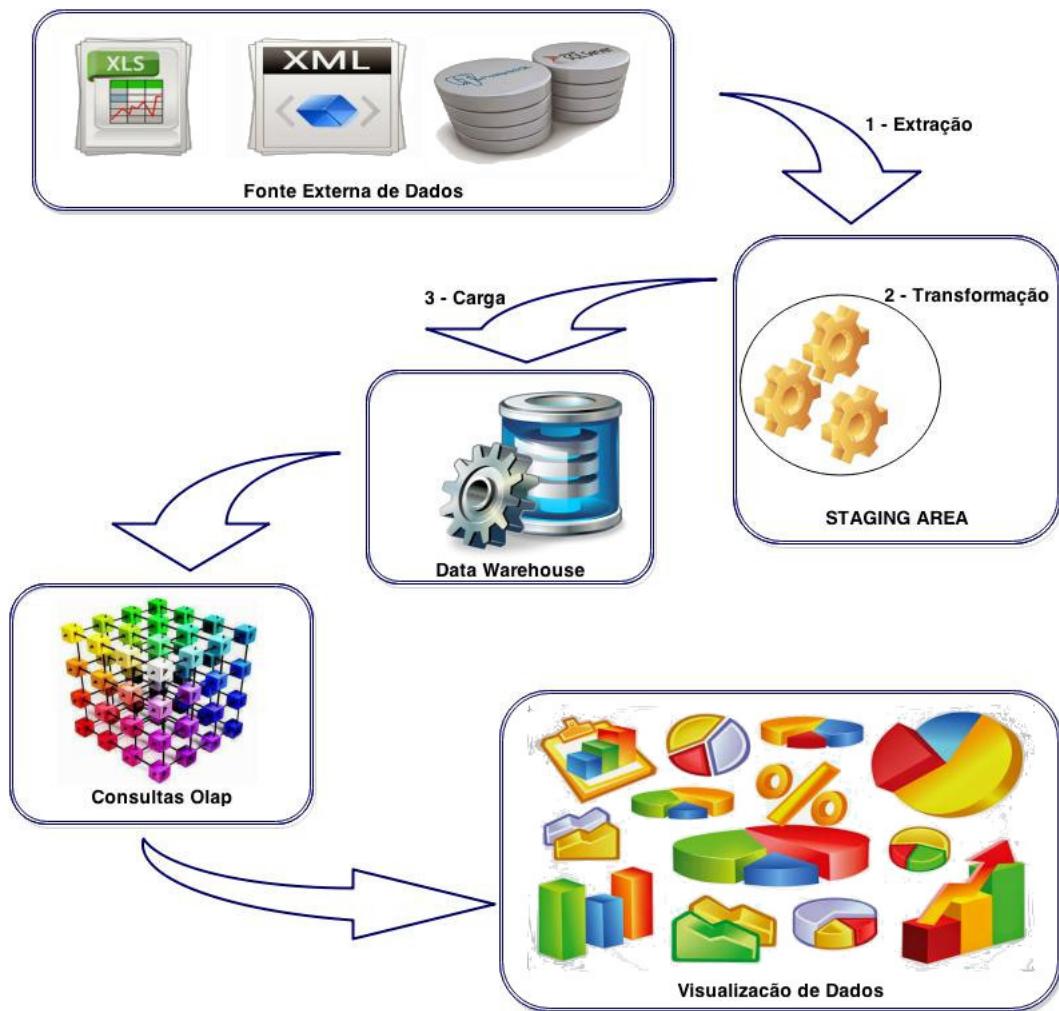


Figura 6 – Arquitetura de um ambiente de Data Warehousing

4.2 Extraction-Transformation-Load

A *Staging Area* é uma área de armazenamento onde acontece o processo de ETL. Na Figura 6, as etapas 1- Extração, 2- Transformação e 3- Carga formam o processo de *Extraction-Transformation-Load* (ETL). Cada uma das etapas recebe a seguinte descrição:

- **Extração:** Primeira etapa do processo de ETL, consiste na leitura e entendimento da fonte dos dados, copiando os que são necessários para futuros trabalhos (KIM-

[BALL; ROSS, 2002](#)).

- **Transformação:** Após a etapa de extração ter sido feita, os dados podem receber diversos tipos de transformações, que incluem correções de conflitos, conversão de formatos, remoção de campos que não são úteis, combinação entre dados de diversas fontes, entre outros ([KIMBALL; ROSS, 2002](#)).
- **Carga:** Após ter sido realizado o processo de transformação, os dados já estão prontos para serem carregados no *Data Warehouse*, tornando possível que todos os dados visualizados após esse processo reflitam a informação que passou pelos processos de extração e transformação ([SHARMA, 2011](#)).

4.3 Modelagem Dimensional

Modelagem dimensional é um novo nome para uma velha técnica para deixar os bancos de dados simples e compreensíveis. No início dos anos 70 as organizações de TI, consultores, usuários finais e fornecedores tiveram que migrar para uma estrutura dimensional simples que combinasse com a necessidade humana pela simplicidade ([KIMBALL; ROSS, 2002](#)).

Segundo [Kimball e Ross \(2002\)](#), a modelagem dimensional tem sido amplamente aceita como a técnica dominante para a apresentação do *data warehouse*. Os profissionais e especialistas de *data warehouse* reconhecem que a apresentação do *data warehouse* deve ser fundamentada na simplicidade. A simplicidade é a chave fundamental que permite que os usuários entendam facilmente as bases de dados e naveguem de forma eficiente no bancos de dados do *software*.

Para facilitar na difusão do conceito de modelagem dimensional [Kimball e Ross \(2002\)](#) utiliza como exemplo um diretor geral que descreve seus negócios como "Nós vendemos produtos em várias áreas de negócio e medimos nosso desempenho ao longo do tempo". Assim, designers dimensionais colocariam em ênfase o produto, as áreas de negócio e o tempo, pensando intuitivamente neste negócio como um cubo de dados, onde as bordas estariam marcadas como produto, negócio e tempo. Pontos dentro do cubo são onde as medições que combinam produtos, áreas de negócio e tempo são salvas. A capacidade de visualizar algo tão abstrato como um conjunto de dados de uma forma concreta e tangível é o segredo da compreensão.

Os conceitos básicos da modelagem dimensional são: fatos, dimensões e medidas. Um fato é uma coleção de itens de dados relacionados, que consiste em medidas e dados de contexto. Ele representa tipicamente itens de negócios ou transações de negócio. A dimensão é uma coleção de dados que descrevem uma dimensão negócio. Dimensões determinam o contextual a fundo para os fatos; eles são os parâmetros nos quais queremos

realizar a *On-Line Analytic Processing (OLAP)*. A medida é um atributo numérico de um fato, que representa o desempenho ou comportamento do negócio em relação às dimensões (GUTIÉRREZ; MAROTTA, 2000).

Considerando o contexto relacional, existem dois modelos básicos que são utilizados na modelagem dimensional: modelo de estrela e modelo de floco de neve. O modelo estrela é a estrutura básica de um modelo dimensional. Ele tem uma grande tabela central (tabelas fato) e um conjunto de pequenas tabelas (tabelas dimensão) dispostos em um padrão radial ao redor da tabela central, como é mostrado na Figura 7. O modelo de floco de neve é o resultado da decomposição de uma ou mais das dimensões (GUTIÉRREZ; MAROTTA, 2000).

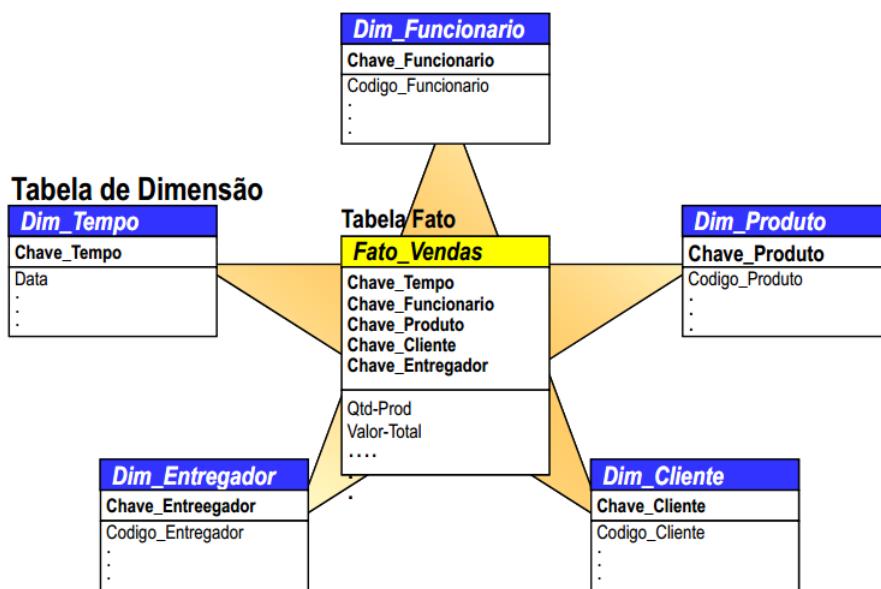


Figura 7 – Esquema estrela extraído de Times (2012)

4.3.1 OLAP (*On-Line Analytic Processing*)

A atividade que consiste em buscar e apresentar os dados de um *Data Warehouse*, sendo essa busca quase sempre baseada em um cubo multidimensional de dados, é chamada de *On-Line Analytic Processing (OLAP)* (KIMBALL; ROSS, 2002). Segundo Sharma (2011), *On-Line Analytic Processing(OLAP)* refere-se a cargas onde grandes quantidades de históricos de dados são processados para gerar relatórios e executar análise de dados. Normalmente, bancos de dados que utilizam OLAP são alimentados a partir de bancos de dados *On-Line Transaction Processing (OLTP)* que são alterados para gerir cargas OLAP. Um banco de dados OLAP armazena um grande volume com os mesmos dados transacionais que um banco de dados OLTP, mas estes dados são transformados pelo processo de ETL para permitir um melhor desempenho para geração de relatórios e para facilitar as análises. Geralmente sistemas OLTP são ajustados para inserções, atualiza-

ções e exclusões bem rápidas, enquanto os sistemas OLAP estão ajustados para consultas rápidas.

A tabela 15 evidencia as diferenças entre aplicações OLTP e OLAP extraídas do trabalho de ([NERI, 2002](#)):

OLPT	OLAP
Controle operacional	Tomada de decisão
Atualização de dados	Análise de dados
Pequena complexidade das operações	Grande complexidade das operações
Não armazena dados históricos	Armazena dados históricos
Voltada ao pessoal operacional	Voltada aos gestores do negócio

Tabela 15 – Diferenças entre OLTP e OLAP extraídas de [Neri \(2002\)](#)

No modelo multidimensional, os dados são organizados em múltiplas dimensões, e cada dimensão contém vários níveis de abstração definidos pelas hierarquias. Esta organização fornece aos usuários a flexibilidade para visualizar os dados a partir de diferentes perspectivas. Existe uma série de operações de cubo de dados OLAP para materializar esses diferentes pontos de vista, permitindo consultas interativas e análises de dados. Assim, OLAP fornece um ambiente amigável para análise de dados interativos.

Entre as operações OLAP estão:

- **Drill Down:** Busca aumentar o nível de detalhamento, partindo de um certo nível de dados para um nível mais detalhado ([SHARMA, 2011](#)).
- **Drill Up:** Ao contrário da operação *Drill Down*, a *Roll Up* parte de um nível mais detalhado para um nível menos detalhado ([SHARMA, 2011](#)).

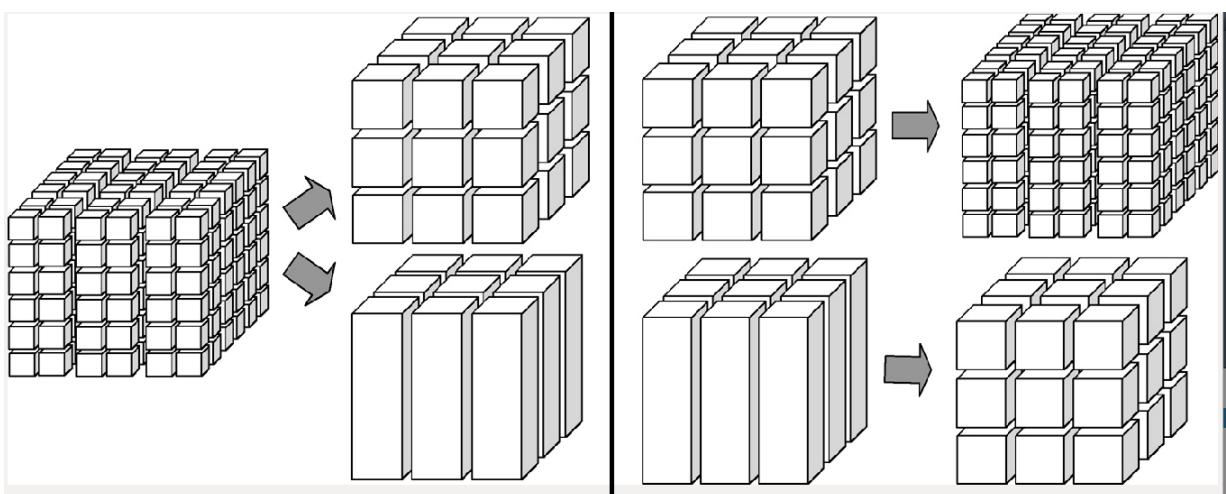


Figura 8 – Exemplo de operações *Drill Down*(direita) e *Drill up*(esquerda) extraídos de [Golfarelli \(2009\)](#)

- **Slice and Dice:** Técnica com filosofia parecida à cláusula *where* usada em *SQL*. Permite que sejam criadas restrições na análise dos dados (TIMES, 2012). O *Slice* faz restrição de um valor ao longo de uma dimensão, já o *Dice* faz restrições de valores em várias dimensões. Semelhante ao *Slice*, só que mais complexo.

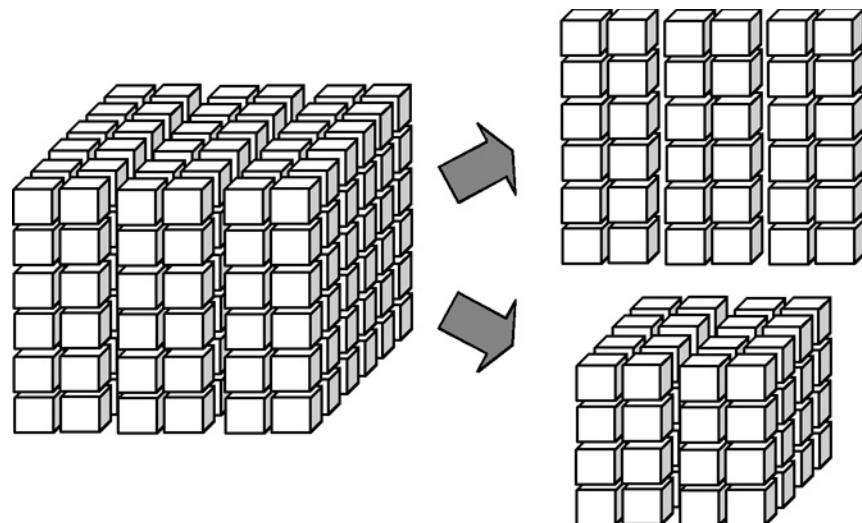


Figura 9 – Exemplo de operações *Slice*(acima) e *Dice*(embaixo) extraídos de [Golfarelli \(2009\)](#)

- **Drill Across:** Permite que diferentes cubos sejam concatenados (NERI, 2002). Uma operação do tipo *Drill Across* irá simplesmente unir diferentes tabelas fato através de dimensões correspondentes (KIMBALL, 1998).

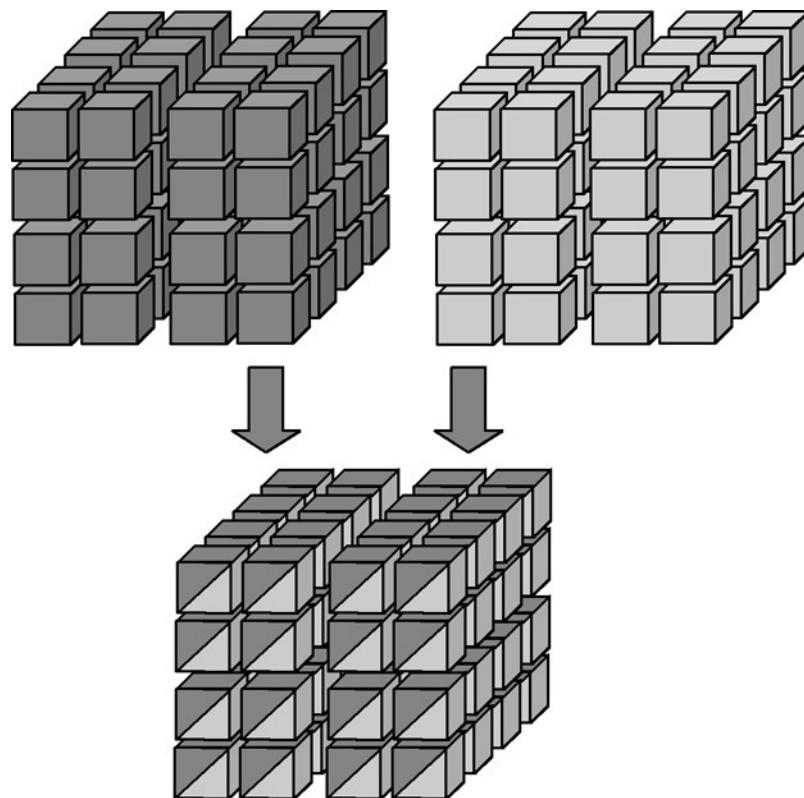


Figura 10 – Exemplo da operação *Drill Across* extraído de [Golfarelli \(2009\)](#)

- **Pivoting:** Metaforicamente, significa rotacionar o cubo. Essa técnica altera a ordenação das tabelas dimensionais ([NERI, 2002](#)).

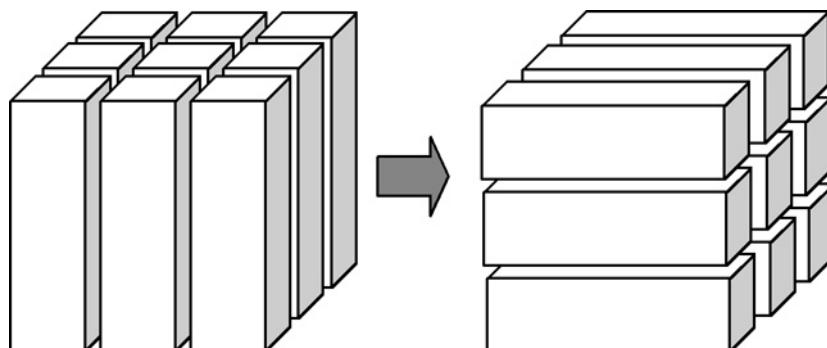


Figura 11 – Exemplo da operação *Pivoting* extraído de [Golfarelli \(2009\)](#)

4.4 Visualização de Dados

O processo de monitorização visual envolve uma série de passos sequenciais que o *dashboard* deve ser concebido para apoio. O usuário ao começar a obter uma visão geral do que está acontecendo deve rapidamente identificar o que precisa de atenção, para que em seguida, o usuário olhe mais de perto cada uma dessas áreas que precisam de atenção para entendê-las bem o suficiente para determinar se algo deve ser feito sobre eles. O

monitoramento é uma atividade cognitiva que recebe a entrada principalmente através do canal visual, porque este é o sentido mais poderoso, que é capaz de trabalhar em altas velocidades de entrada, capaz de detectar diferenças sutis e complexas.(Few, 2006)

A orientação visual dos *dashboards* é importante devido à velocidade da percepção de que é geralmente necessária para monitorar informações. Quanto mais rápido se deseja avaliar o que está acontecendo, mais deve-se confiar no meio gráfico para mostrar a informação. O texto deve ser lido, o que envolve um processo relativamente lento, sendo que certas propriedades visuais, no entanto, pode ser percebida de relance, sem pensamento consciente.

Segundo Few (2006), um *dashboard* é um display visual das informações mais importantes necessárias para alcançar um ou mais objetivos, consolidados e organizados em uma única tela para que a informação possa ser monitorada em um piscar de olhos. Few (2006) categorizou diversos tipos de *dashboards* sendo estratégicas, analíticas, ou operacionais, e as características do design no que tange à sugestão de organização variam para dar suporte às necessidades de cada categoria:

- fins estratégicos - O uso primários de *dashboards* nos dias de hoje é para propósitos estratégicos, oferecem uma rápida visão que os tomadores de decisão precisam para monitorar a saúde e as oportunidades de um negócio.
- fins analíticos - Mais sofisticação para as mídias de exibição, para que os analistas possam examinar melhor dados complexos e relacionamentos. *Dashboards* analíticos devem suportar interações com os dados, como aprofundamentos em camadas detalhadas, não apenas para ver o que está acontecendo, mas para examinar as causas.
- fins operacionais - *Dashboards* que monitorem operações devem manter consciência das atividades e eventos que estão mudando constantemente e podem demandar atenção e resposta.

Além de categorizar os tipos de *dashboards*, Few (2006) também evidencia o primordial para se obter um *dashboard* de qualidade:

- Disponibilizar a informação diretamente relacionada num único ecrã, ou seja, evitar partilhar a informação por várias páginas;
- Evitar a necessidade de *scrolling*;
- Contextualizar a informação disponibilizada;
- Incluir fatores de comparação e sugerir ações na visualização dos indicadores;
- Utilizar escalas adequadas, que devem dar uma perspectiva real das quantidades

apresentadas e não podem iludir os utilizadores;

- Utilizar níveis de precisão adequados nos indicadores, pois evita perdas de tempo com leituras e interpretações de informação desnecessárias e pouco relevantes;
- Escolher os indicadores mais adequados, que facilitem e acelerem a interpretação da informação disponibilizada;
- Escolher soluções gráficas flexíveis e adequadas que facilitem e acelerem a interpretação da informação disponibilizada;
- Uniformizar a leitura ao longo do *dashboard*;
- Facilitar a interpretação da informação disponível para acelerar a sua leitura. Por exemplo, evitar cores berrantes, muito próximas, muito apagadas ou um número muito elevado de cores;
- Apresentar a informação de forma equilibrada, dado que o espaço utilizado num *dashboard* desce de importância do canto superior esquerdo para o canto inferior direito, e por esta razão, a informação que se destaca na visualização deverá ser a mais importante;
- Os títulos não devem ser mais apelativos que os indicadores;
- Destacar a informação mais importante e não cair no erro de chamar a atenção para tudo;
- Aproveitar bem o espaço disponível, ou seja evitar decorações desnecessárias e ainda evitar soluções de pesada implementação para responder a pormenores visuais;
- Utilizar cores de forma ponderada, ou seja, utilizar cores apelativas apenas para a informação mais importante, podendo utilizar contrastes;
- Manter as cores para os mesmos indicadores ao longo do *dashboard* ou para o mesmo tipo de indicador associado;
- Podem ser utilizadas figuras geométricas para além das cores, tais como o círculo, triângulo ou quadrado como forma de ajudar utilizadores que sofram de daltonismo;
- Criar uma apresentação apelativa, baseando-se no nossa intuição e naquilo que consideramos que a maioria das pessoas aceita e tolera positivamente.

4.4.1 Ferramenta para criação de *Dashboards*

Neste trabalho foi utilizado o CDE (*community dashboard editor*) na versão 14.12.10.1 para a criação dos *dashboards*. O CDE é um *plugin* para o *Pentaho Business Intelligence*.

O CDE permite o desenvolvimento e a implantação de *dashboards* no *Pentaho Business Intelligence*. O CDE nasceu para simplificar a criação e edição de *dashboards* e é uma ferramenta muito poderosa e completa, combinando *front end* com fontes de dados e componentes personalizados de uma forma perfeita. ([CDE... , 2014](#))

O CDE possui algumas tecnologia por trás dela:

- CDF (*Community Dashboard Framework*) - É um *framework* HTML/javascript que permite criar páginas com relatórios, gráficos e tabelas.
- CDA (*Community Dashboard Access*) - São vários componentes que dão acesso à diferentes tipos de fontes de dados.
- CCC (*Community Chart Components*) - É uma biblioteca de gráficos, que possui um poderoso conjunto de ferramentas de visualização livre e de código aberto. O objetivo do CCC é fornecer aos desenvolvedores o caminho para incluir em seus *dashboards* os tipos de gráficos básicos, sem perder o princípio fundamental: a extensibilidade.

Para o design do *dashboard*, o CDE oferece três perspectivas:

- *Layout* - Para projetar o layout do seu dashboard a partir do zero ou de um template, ao definir o layout é possível aplicar estilos e adicionar elementos HTML para descrever páginas Web, CSS para controlar o estilo do layout, *JavaScript* para adicionar interatividade e *jQuery* para simplificar todas essas tarefas.
- *Components* - Para adicionar e configurar os diferentes componentes que compõem o seu *dashboard*: Componentes Visuais (caixas de texto, tabelas e gráficos, seletores e relatórios), parâmetros que representam os valores que são partilhados pelos componentes e *Scripts*, que permitem personalizar a aparência ou o comportamento de outros componentes.
- *Datasources* - Define os dados usados pelos componentes. *Dashboards* podem ser povoados por uma variedade de fontes como: Bases de dados, cubos do *Mondrian*, metadados do Pentaho , arquivos XML, *ad-hoc datasource* e transformações do *Kettle*.

A descrição completa da implementação do *dashboard* foi descrita no Apêndice [B](#).

4.5 Ambiente de *Data Warehousing* para Métricas, bugs e violações de Código-Fonte

Para a implementação do ambiente de Data Warehousing para métricas de código-fonte, [Rêgo \(2014\)](#) definiu a arquitetura tal como mostra a Figura 6.

A ferramenta de análise estática de código-fonte escolhida por Rêgo (2014) foi a Analizo. A Analizo possibilita emitir saídas das métricas em CSV que detalham nome da classe e as respectivas métricas, o que permitiu a seu trabalho incorporar a análise das métricas ANPM, AMLOC, CBO, NPA. As ferramentas FindBugs e PMD, descritas no capítulo 2.5.1, foram adicionadas ao ambiente de *Data Warehousing* como pode ser visto na Figura 18.

Rêgo (2014) seguiu a metodologia de (KIMBALL; ROSS, 2002), apresentada na Figura 12, para o seu projeto de *data warehouse*. Seguindo esta metodologia, Rêgo (2014) definiu os seguintes requisitos de negócio que a sua solução deveria suportar:

- **Requisito 1:** Visualizar o intervalo qualitativo obtido para cada métrica de código-fonte em uma determinada *release* do projeto para a configuração *Open JDK8 Metrics*.
- **Requisito 2:** Comparar o intervalo qualitativo obtido para cada métrica de código-fonte ao longo de todas as *releases* de um projeto para a configuração *Open JDK8 Metrics*.
- **Requisito 3:** Visualizar o valor percentil obtido para cada métrica de código-fonte em uma determinada *release* do projeto para a configuração *Open JDK8 Metrics*.
- **Requisito 4:** Comparar o valor percentil de cada métrica de código-fonte ao longo de todas as *releases* para a configuração *Open JDK8 Metrics*.
- **Requisito 5:** Visualizar o intervalo qualitativo obtido para cada métrica de código-fonte em uma determinada *release* do projeto para a configuração *Tomcat Metrics*.
- **Requisito 6:** Comparar o intervalo qualitativo obtido para cada métrica de código-fonte ao longo de todas as *releases* de um projeto para a configuração *Tomcat Metrics*.
- **Requisito 7:** Visualizar a medida obtida para cada métrica de código-fonte em uma determinada *release* do projeto para a configuração *Tomcat Metrics*.
- **Requisito 8:** Comparar o valor percentil obtido para cada métrica de código-fonte ao longo de todas as *releases* para a configuração *Tomcat Metrics*.
- **Requisito 9:** Visualizar a quantidade de cenários de limpeza identificados por tipo de cenários de limpeza de código-fonte em cada classe ao longo de cada *release* de um projeto.
- **Requisito 10:** Comparar a quantidade de cenários de limpeza por tipo de cenários de limpeza de código-fonte em uma *release* de um projeto.

- **Requisito 11:** Visualizar o total de cenários de limpeza em uma determinada *release* de um projeto.
- **Requisito 12:** Visualizar cada uma das classes com um determinado cenário de limpeza de código-fonte ao longo das *releases* do projeto.
- **Requisito 13:** Visualizar as 10 classes de um projeto com menor número de cenários de limpeza identificados.
- **Requisito 14:** Visualizar as 10 classes de um projeto com maior número de cenários de limpeza identificados.
- **Requisito 15:** Acompanhar a Taxa de Aproveitamento de Oportunidades de Melhoria de Código-Fonte que é a divisão do total de cenários de limpeza identificados em uma *release* e o o número total de classes da mesma *release* de um projeto



Figura 12 – Metodologia de Projeto de *Data Warehouse* proposta por [Kimball e Ross \(2002\)](#) extraída de [Rêgo \(2014\)](#)

Utilizando a mesma metodologia que o [Rêgo \(2014\)](#), foi definido novos requisitos de negócio que a solução deveria suportar:

- Requisito 16: Visualizar o total de *bugs* em uma determinada *release* de um projeto.
- Requisito 17: Visualizar o total de violações em uma determinada *release* de um projeto.
- Requisito 18: Visualizar a quantidade de *bugs* por severidade em uma determinada *release* de um projeto.
-
- Requisito 19: Visualizar a quantidade de *bugs* por tipo em uma determinada *release* de um projeto.
- Requisito 20: Visualizar a quantidade de *bugs* por classe em uma determinada *release* de um projeto.
- Requisito 21: Visualizar a quantidade de violações por severidade em uma determinada *release* de um projeto.
- Requisito 22: Visualizar a quantidade de violações por tipo em uma determinada

release de um projeto.

- Requisito 23: Visualizar a quantidade de violações por classe em uma determinada *release* de um projeto.

Ainda seguindo a metodologia de (KIMBALL; ROSS, 2002), Rêgo (2014) identificou fatos e dimensões no contexto de monitoramento de métricas, como mostra a Tabela 16.

Fato	Dimensões
Valor Percentil	<ul style="list-style-type: none"> • Projeto • Métrica • Configuração • Qualidade • <i>Release</i> • Tempo
Quantidade de Cenários de Limpeza	<ul style="list-style-type: none"> • Projeto • Cenário de Limpeza • Classe • <i>Release</i>
Taxa de Aproveitamento de Oportunidade de Melhoria de Código-Fonte	<ul style="list-style-type: none"> • Projeto • <i>Release</i>

Tabela 16 – Fatos e dimensões identificadas por Rêgo (2014)

Dando prosseguimento a metodologia de (KIMBALL; ROSS, 2002), foram identificadas mais fatos e dimensões em complemento à solução de Rêgo (2014), como mostra a Tabela 17.

Fato	Dimensões
<i>FindBugs</i>	<ul style="list-style-type: none"> • Projeto • <i>Release</i> • Tempo • Classe do <i>Bug</i> • Tipo do <i>Bug</i> • Prioridade do <i>Bug</i>
PMD	<ul style="list-style-type: none"> • Projeto • Tempo • <i>Release</i> • Classe da Violação • Regras • Severidade

Tabela 17 – Fatos e dimensões identificadas neste trabalho.

Após a identificação dos fatos e das dimensões, [Rêgo \(2014\)](#) construiu o projeto físico do *Data Warehouse* que pode ser visto na Figura 13. A Tabela 18 facilita a interpretação do projeto físico.

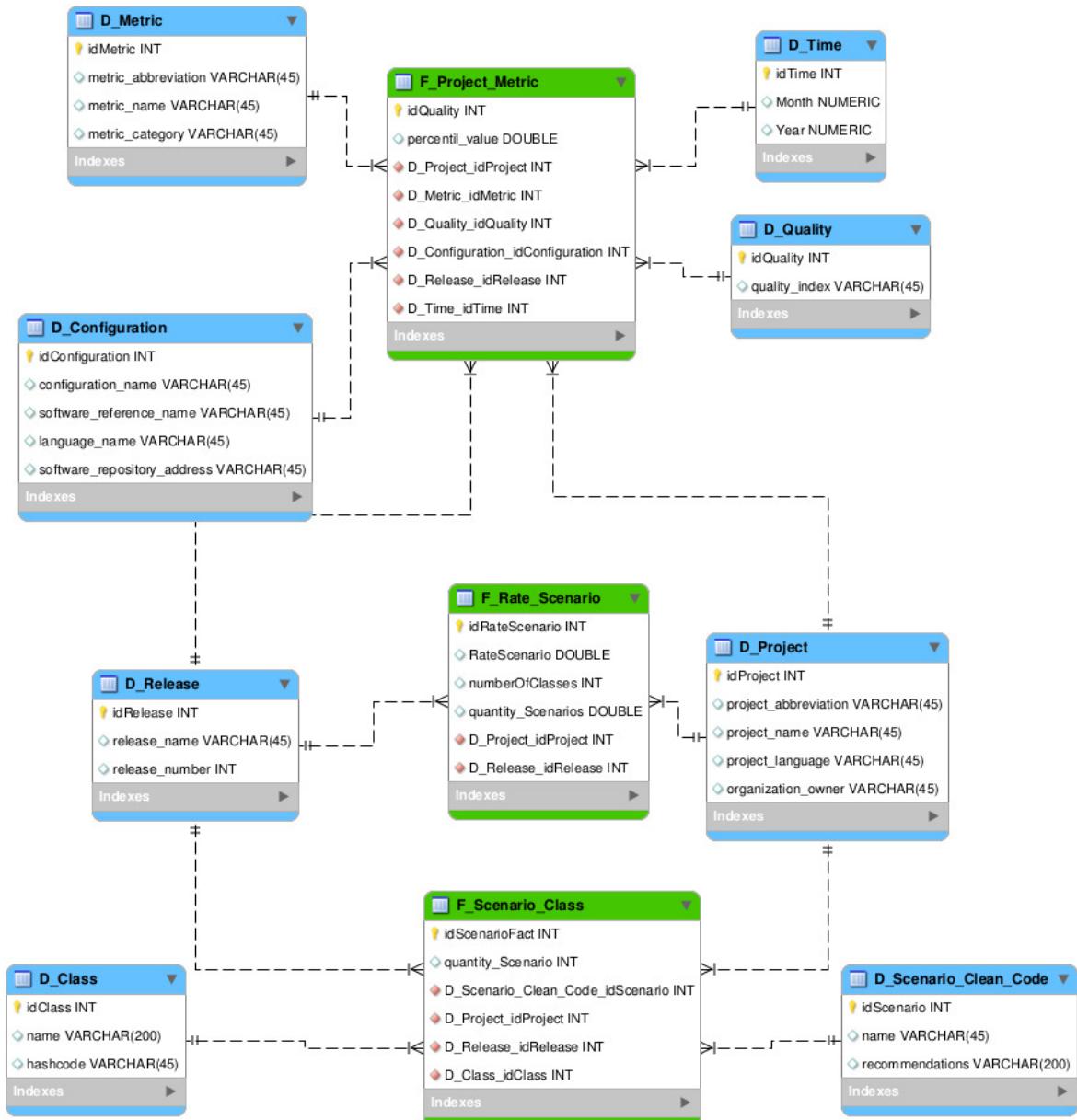
Figura 13 – Projeto físico do *Data Warehouse* extraído de Rêgo (2014)

Tabela Fato	Tabelas Dimensões
F_Project_Metric	<ul style="list-style-type: none">• D_Project• D_Metric• D_Configuration• D_Quality• D_Release• D_Time
F_Scenario_Class	<ul style="list-style-type: none">• D_Project• D_Scenario_Clean_Code• D_Class• D_Release
F_Rate_Scenario	<ul style="list-style-type: none">• D_Project• D_Release

Tabela 18 – Tabelas fatos e tabelas dimensões elaboradas por [Rêgo \(2014\)](#)

Visando facilitar o processo de ETL principalmente na etapa da transformação, Rêgo (2014) criou uma área de metadados mostrada na Figura 14 com intuito de tratar os dados que representam os próprios dados dos processos de negócio. A Tabela 19 descreve as tabelas contidas na área de metadados do *Data Warehouse*.

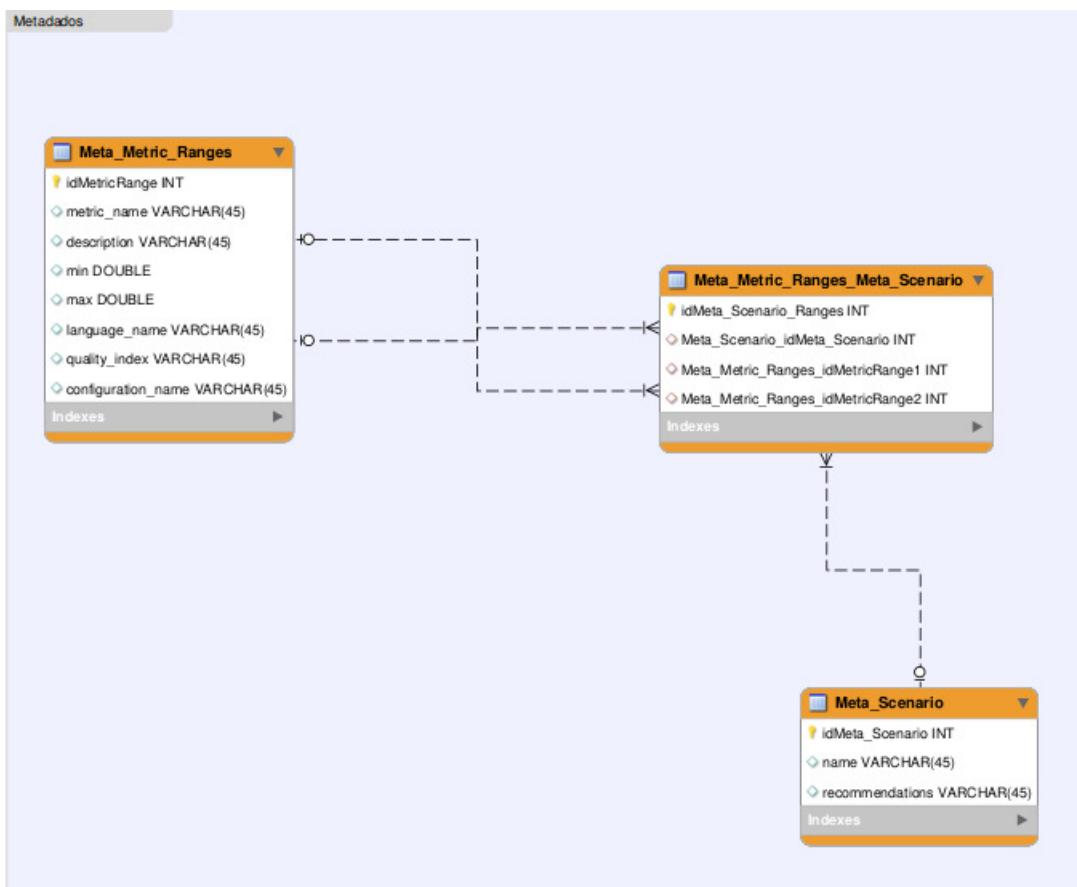


Figura 14 – Projeto físico do *Data Warehouse* extraído de Rêgo (2014)

Tabelas	Descrição
<i>Meta_Metric_Ranges</i>	Contém cada configuração de intervalo qualitativo para cada métrica de código fonte.
<i>Meta_Scenario</i>	Contém os cenários de limpeza de código e suas recomendações.
<i>Meta_Metric_Ranges_Meta_Scenario</i>	Como a cardinalidade entre as tabelas <i>Meta_Scenario</i> e <i>Meta_Metric_Ranges</i> seria de n para n, essa tabela foi criada contendo os registros únicos dessas duas tabelas.

Tabela 19 – Descrição das Tabelas do Metadados do *Data Warehouse*

Após a identificação dos fatos e das dimensões deste trabalho, o projeto físico do *Data Warehouse* construído por Rêgo (2014) foi alterado. Foram adicionadas as Tabelas *F_Project_Bug*, *D_Class_Bug*, *D_Type*, *D_Priority*, *F_Project_Violation*, *D_Rules*, *D_Severite* e *D_Class_Violation* como pode ser visto nas Figuras 15 e 16. A Figura 17

mostra como ficou o projeto físico finalizado e as Tabelas 18, 20 e 19 facilitam a sua interpretação.

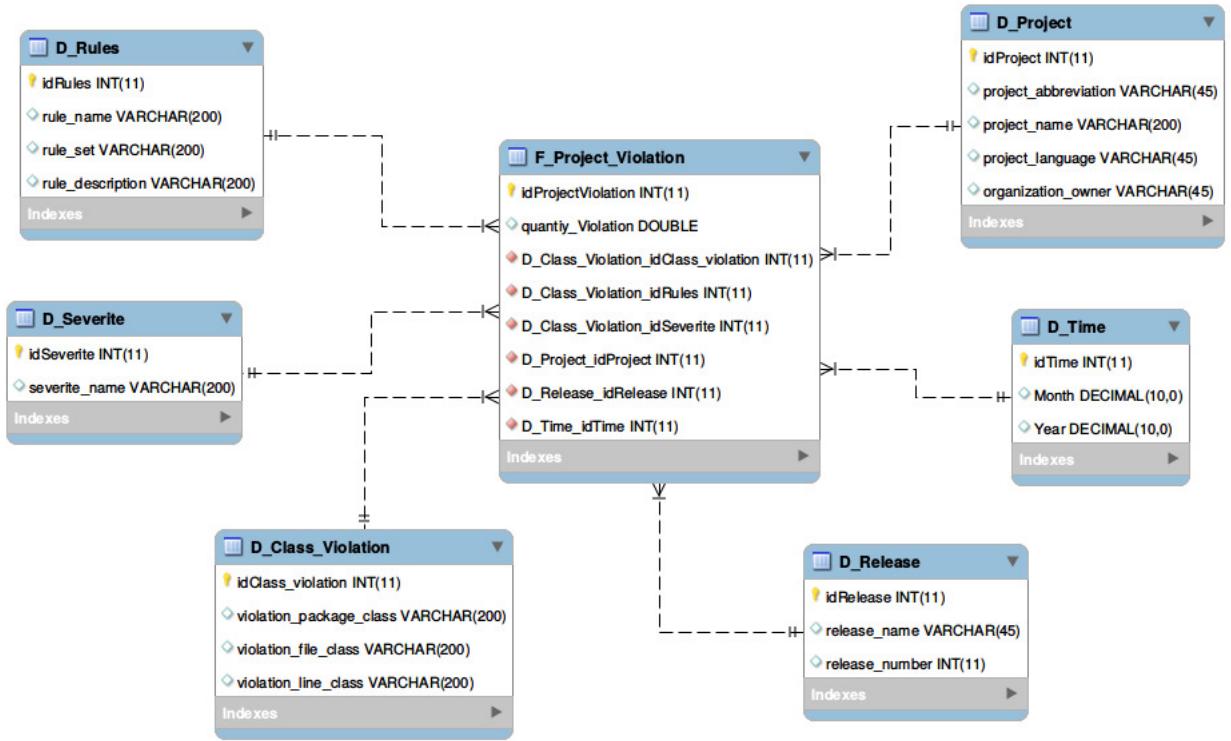
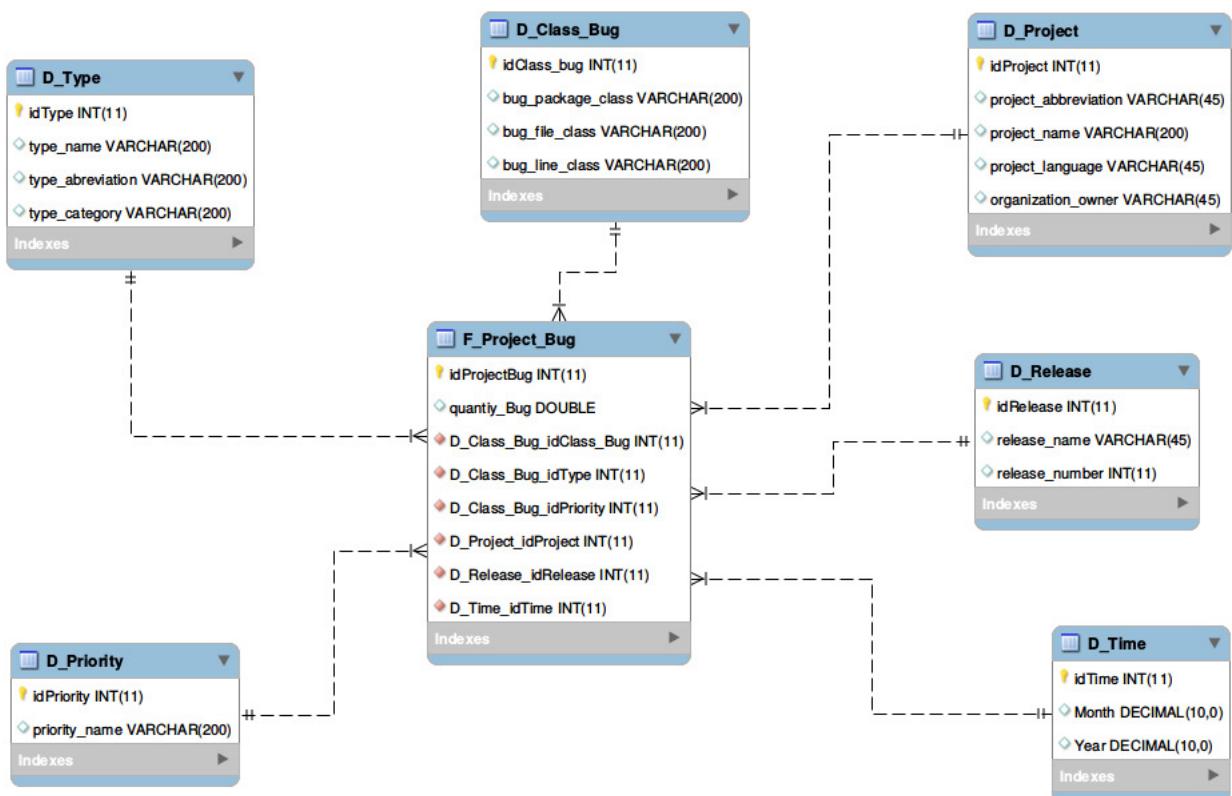


Figura 15 – Parte do projeto físico contendo as tabelas relacionados ao PMD

Figura 16 – Parte do projeto físico contendo as tabelas relacionados ao *FindBugs*

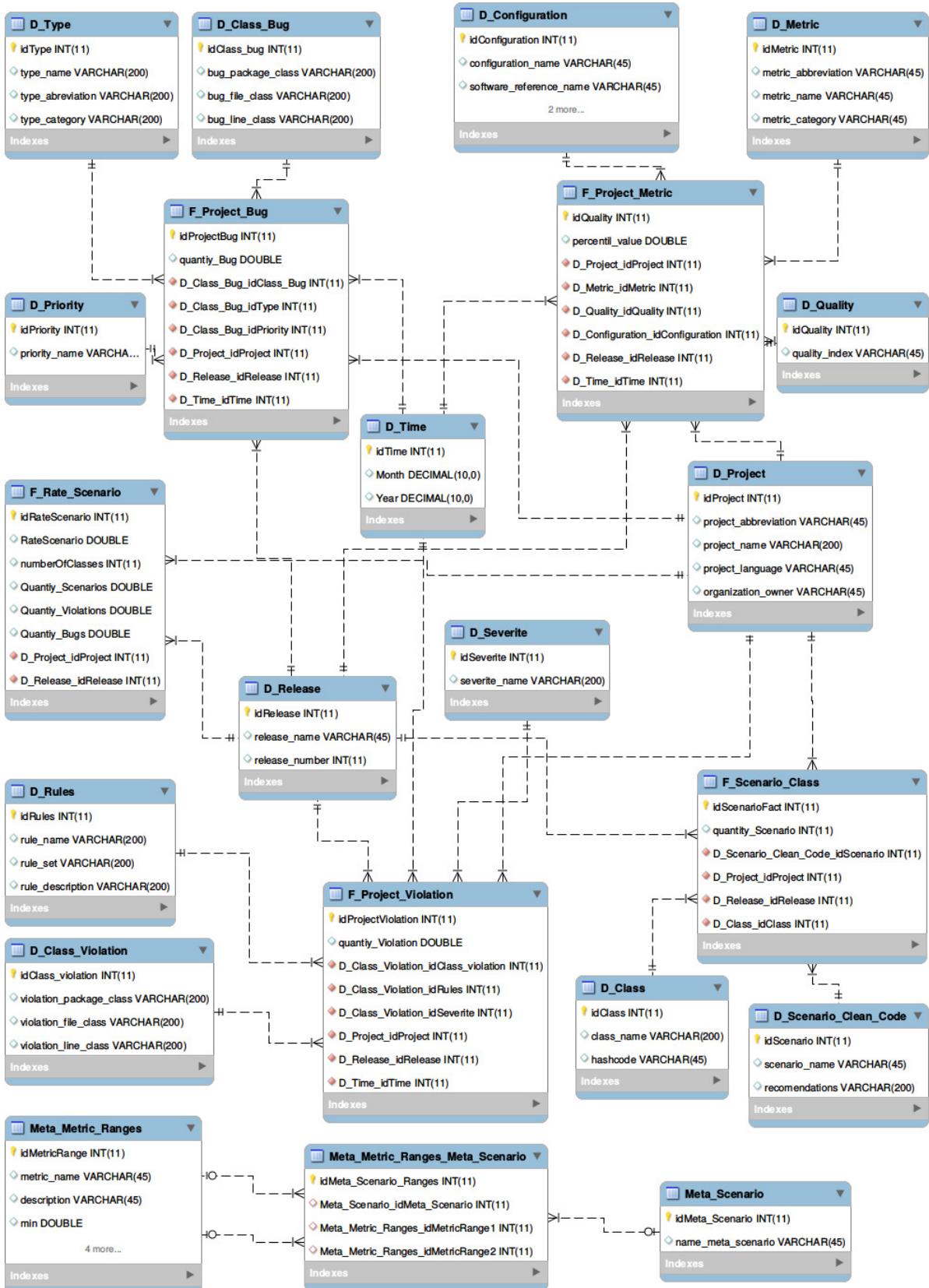
Figura 17 – Projeto físico do *Data Warehouse* finalizado.

Tabela Fato	Tabelas Dimensões
F_Project_Bug	<ul style="list-style-type: none"> • D_Project • D_Release • D_Time • D_Class_Bug • D_Type • D_Priority
F_Project_Violation	<ul style="list-style-type: none"> • D_Project • D_Release • D_Time • D_Class_Violation • D_Rules • D_Severite

Tabela 20 – Tabelas fatos e tabelas dimensões adicionadas à solução de Rêgo (2014)

4.5.1 Ferramentas de *Data Warehousing*

Entre as alternativas de código aberto que suportam um modelo dimensional, Rêgo (2014) utilizou o *Pentaho Business Analytics Community Edition*. Esta alternativa livre apresenta soluções que cobrem as áreas de ETL, *reporting*, *OLAP* e mineração de dados. A Figura 18, apresenta o modo como cada uma das ferramentas está disposta na arquitetura do ambiente de *Data Warehousing* para Métricas, *Bugs* e Violações de Código-Fonte.

Segue abaixo as ferramentas utilizadas em cada etapa do *Data Warehouse*:

- **ETL:** Pentaho Data Integration Community Edition ou Kettle.
- **Implementação das Consultas OLAP:** *Pentaho BI Platform 8*, provê uma arquitetura e a infraestrutura para soluções de *Business Intelligence*, *Data Mining*.
- **Visualização de Dados:** Como visto na subseção 4.4 foi utilizado o *plugin* do *Pentaho BI CDE* para a visualização de dados. O *plugin* *Saiku Analytics* também pode ser utilizado para visualização de dados, ela oferece serviços de apoio a operações OLAP e à visualização de dados. Na arquitetura do *Saiku Analytics*, está incorporado outro *software* livre que realiza consulta, chamado de *Mondrian OLAP*.
- **Análise estática de código-fonte:** *Analizo*, possibilita emitir saídas das métricas em CSV que detalham o nome da classe e as suas respectivas métrica.

- **Verificadores de erro:** Como visto no Capítulo 2.5.1 foram utilizados as ferramentas *FindBugs* e *PMD* que possibilitam uma análise de código voltadas para bugs e violações de código.

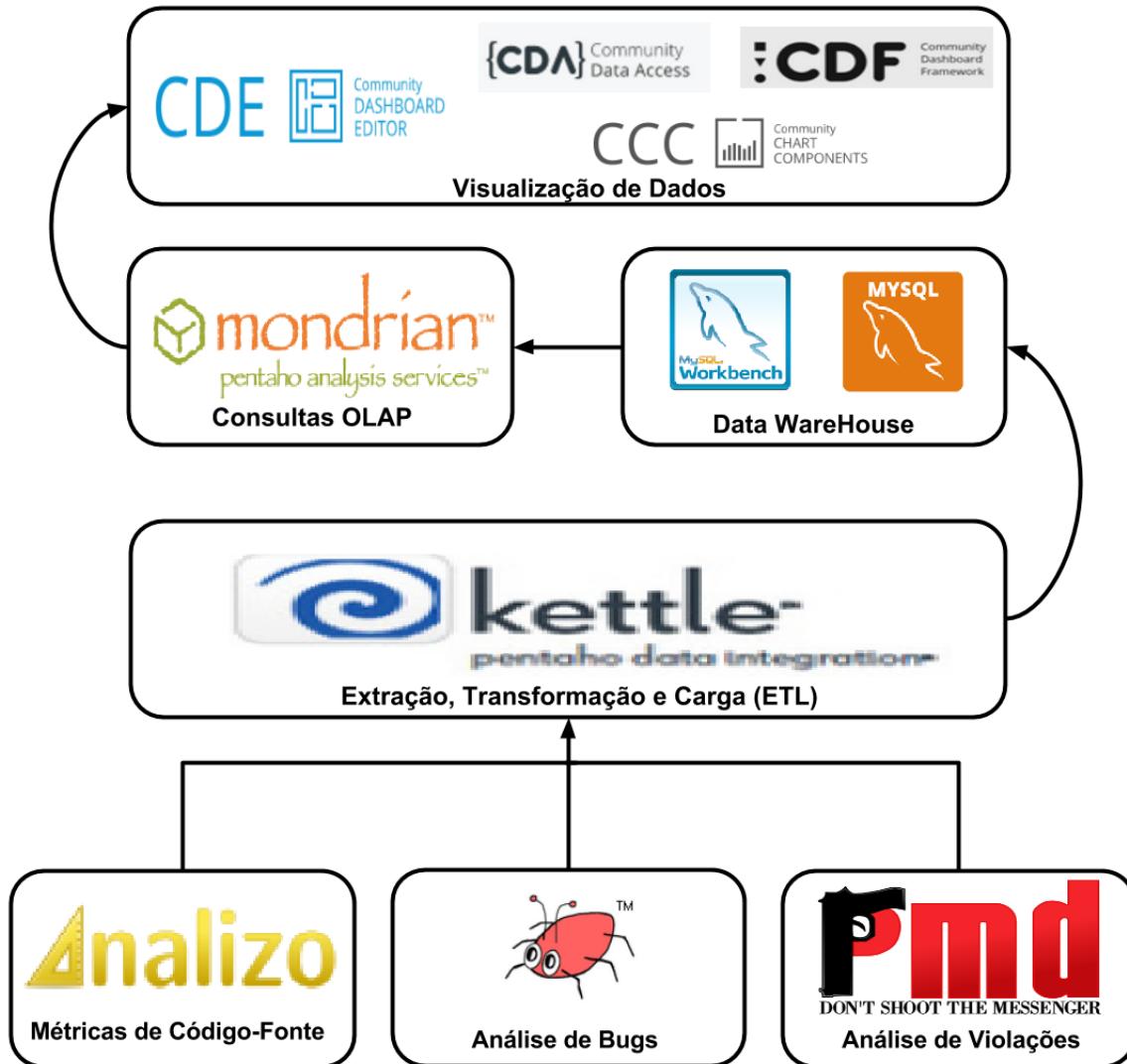


Figura 18 – Arquitetura Ambiente de *Data Warehousing* para Métricas de Código-Fonte extraído de Rêgo (2014) e adaptado.

A descrição completa da implementação do processo de *Extraction-Transformation-Load* na ferramenta *Kettle* foi descrita no Apêndice A.

4.6 Considerações finais do capítulo

Nesse capítulo foi apresentada a solução proposta no trabalho de Rêgo (2014) e o que foi acrescentado visando suprir os novos requisitos adicionados, bem como a base teórica para a compreensão de ambos. No próximo capítulo será apresentado o projeto de

estudo de caso que visa analisar se a solução de DW proposta nesse capítulo pode assistir ao processo de aferição da qualidade de *software*.

5 Estudo de Caso

Neste capítulo são apresentados o projeto e a execução do estudo de caso, as análises dos dados quantitativos e qualitativos e a discussão dos resultados, realizado neste trabalho de conclusão de curso. Isso consistiu na elaboração de um protocolo para o estudo de caso, na identificação do problema e na definição das questões e objetivos de pesquisa. O método para coleta dos dados e como eles foram analisados também são apresentados neste capítulo.

5.1 Planejamento do Estudo de Caso

Segundo [Yin \(2001\)](#), o estudo de caso é um conjunto de procedimentos pré-especificados para se realizar um estudo experimental que investiga um fenômeno contemporâneo dentro de seu contexto da vida real, especialmente quando os limites entre o fenômeno e o contexto não estão claramente definidos. Uma grande vantagem do estudo de caso é a sua capacidade de lidar com uma ampla variedade de evidências, documentos, artefatos, entrevistas e observações. Além disso, em algumas situações, como na observação participante, pode ocorrer manipulação informal.

Algumas perguntas norteadoras auxiliam no entendimento do estudo de caso deste trabalho:

- Qual o escopo do estudo de caso?
- Qual o problema a ser tratado?
- Qual a questão de pesquisa relacionada a esse problema?
- Quais são os objetivos a serem alcançados nessa pesquisa?
- Como foi a seleção do estudo de caso?
- Qual é a fonte dos dados coletados nessa pesquisa e qual o método de coleta?
- Qual o método de análise dos dados coletados?
- Quais as ameaças à validade do estudo de caso?

Com o objetivo da elucidação do escopo do estudo de caso proposto neste trabalho, bem como ilustrado na [Figura 19](#), foram apresentados, nos capítulos anteriores: i) um estudo teórico relacionado à métricas de software; ii) verificadores de erro; iii) contratação de serviços de TI por parte da Administração Pública Federal brasileira; iv) uso de Data

Warehouse, construído para aferição de qualidade interna de produto de software v) a importância do uso de *dashboard*. Adicionalmente, foi apresentada uma solução para o coleta, monitoramento e interpretação de Métricas de Código-Fonte com suporte de um ambiente de Data *Warehousing*.

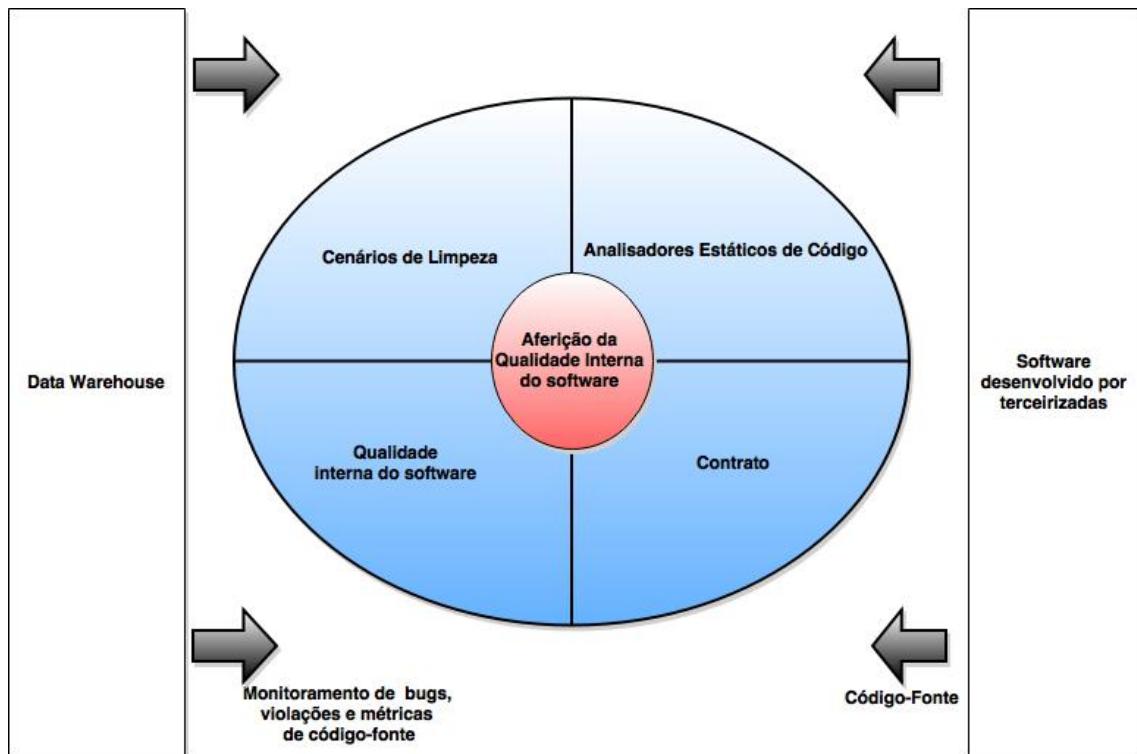


Figura 19 – Escopo do Estudo de Caso

Para as demais perguntas a serem respondidas, foi estruturado um protocolo de estudo de caso baseado em [Brereton, Kitchenham e Budgen \(2008\)](#) que é dividido da seguinte maneira:

- **Background - Seção 5.2:** Identificar outros estudos acerca do tópico, definir a questão de pesquisa principal e suas proposições derivadas que serão abordadas por este estudo.
- **Design - Seção 5.3:** Identificar se o projeto de pesquisa é um caso único ou múltiplo bem como seu propósito geral. O estudo de caso único envolve a estratégia de pesquisa aplicada à compreensão de várias dimensões do fenômeno com foco em um caso singular enquanto o estudo de caso múltiplo envolve o estudo das mesmas dimensões do fenômeno em mais de um caso simultaneamente.
- **Seleção - Seção 5.4:** Apresentar critérios para a seleção do caso e descrição do objeto de estudo a ser analisado.
- **Fonte e Método de Coleta de Dados - Seção 5.6:** Identificar os dados que serão

coletados, definindo um plano para a coleta e como a informação será armazenada.

- **Processo de Análise dos Dados - Seção 5.7:** Identificar os critérios para interpretação dos resultados do estudo de caso, relacionar os dados com a questão de pesquisa e elaborar a explicação do encontrado.
- **Ameaças a validade do estudo de caso - Seção 5.8:** Elicitar tipos de validades aplicáveis a um estudo de caso, baseando-se no trabalho desenvolvido por [Yin \(2001\)](#), sendo elas: constructo, interna, externa e confiabilidade.

5.2 Background

Esta seção contém referências sobre os trabalhos que antecederam esse estudo de caso dentro de um contexto similar ao que foi apresentado, assim como a própria questão geral de pesquisa a ser respondida com todos os elementos necessários para respondê-la.

5.2.1 Trabalhos Antecedentes e Relacionados

O principal trabalho que antecede essa ideia foi desenvolvido por [Rêgo \(2014\)](#), no qual a solução para monitoramento de métricas de código fonte utilizando *Data Warehouse* foi desenvolvida e utilizada para aferir a qualidade interna de um produto de software contratado por uma Autarquia Federal.

Anteriormente ao trabalho realizado por [Rêgo \(2014\)](#), [Marinescu \(2005\)](#) mostrou em seu trabalho que a utilização de métricas isoladas dificulta a interpretação de anomalias do código, reduzindo a eficácia da medição. Além disso, o autor ainda afirma que a métrica por si só não contém informação suficiente para motivar uma transformação no código que melhore sua qualidade. Buscando trabalhar nesse contexto, [Marinescu \(2005\)](#) apresentou o conceito de interpretação das métricas em um nível de abstração maior que o adquirido ao observar apenas o valor da métrica.

Outro trabalho antecedente a ser destacado é a tese desenvolvida por [Meirelles \(2013\)](#), em que se buscou responder como métricas de código-fonte podem influir na atratividade de projetos de software livre e quais métricas devem ser controladas ao longo do tempo. Além disso, [Meirelles \(2013\)](#) também inseriu como questão de pesquisa se as métricas de código-fonte melhoram com o amadurecimento dos projetos. Durante a execução de seu trabalho, foi observado que, a partir dos resultados obtidos para cada métrica de código-fonte em uma análise realizada no código-fonte de 38 projetos de software livre, para uma determinada linguagem de programação, é possível definir um intervalo qualitativo a partir de um determinado percentil ao observar o conjunto de valores que são frequentemente obtidos.

O trabalho de Machini et al. (2010) fortemente relaciona-se com este trabalho por ter como objetivo fazer com que as métricas de código-fonte sejam mais facilmente incorporadas no cotidiano dos programadores, apresentando uma maneira de interpretar os valores das métricas através de cenários problemáticos que ocorrem frequentemente durante o desenvolvimento.

Já o trabalho de Ruiz et al. (2005) apresenta um ambiente de *data warehousing* para apoiar a implementação de um programa de medição em uma organização certificada como CMM Nível 2. Além disso, foi concebido um experimento que valida sua arquitetura, onde foram usados dados organizacionais reais. Ruiz et al. (2005) aborda três aspectos essenciais:

- A captura de dados, considerando os vários tipos de heterogeneidade.
- A integração, transformação e representação de dados quantitativos do projeto de acordo com a visão organizacional unificada e centralizada.
- A funcionalidade de análise que permite o monitoramento do processo.

Novello (2006) também utilizou *data warehouse* para monitoramento de métricas no processo de desenvolvimento de software, porém essas métricas não eram de código fonte.

Outros trabalhos relacionados ao conceito de métricas de software utilizados nessa revisão são Fenton e Pfleeger (1998), Kan (2002), McCabe (1976). Outros trabalhos que possuem como tema *data warehousing* foram utilizados durante a revisão bibliográfica, entre eles estão Inmon (2002), Kimball e Ross (2002), Sharma (2011).

5.2.2 Questão de Pesquisa

A questão de pesquisa a seguir foi proveniente do problema da falta de capacidade da administração pública em aferir, de forma sistemática e automatizada, a qualidade interna dos produtos de software adquiridos e desenvolvidos por organizações contratadas, que foi identificado na seção 1.3 do capítulo 1.

O uso de um ambiente de *Data Warehousing* para aferição da qualidade interna de software, apoiado por ferramentas de análise estática de código-fonte, pode assistir ao processo de aferição da qualidade interna em uma organização pública federal?

Para estruturar a questão de pesquisa foi utilizada a técnica *goal-question-metrics* conforme ilustrado na figura 20.

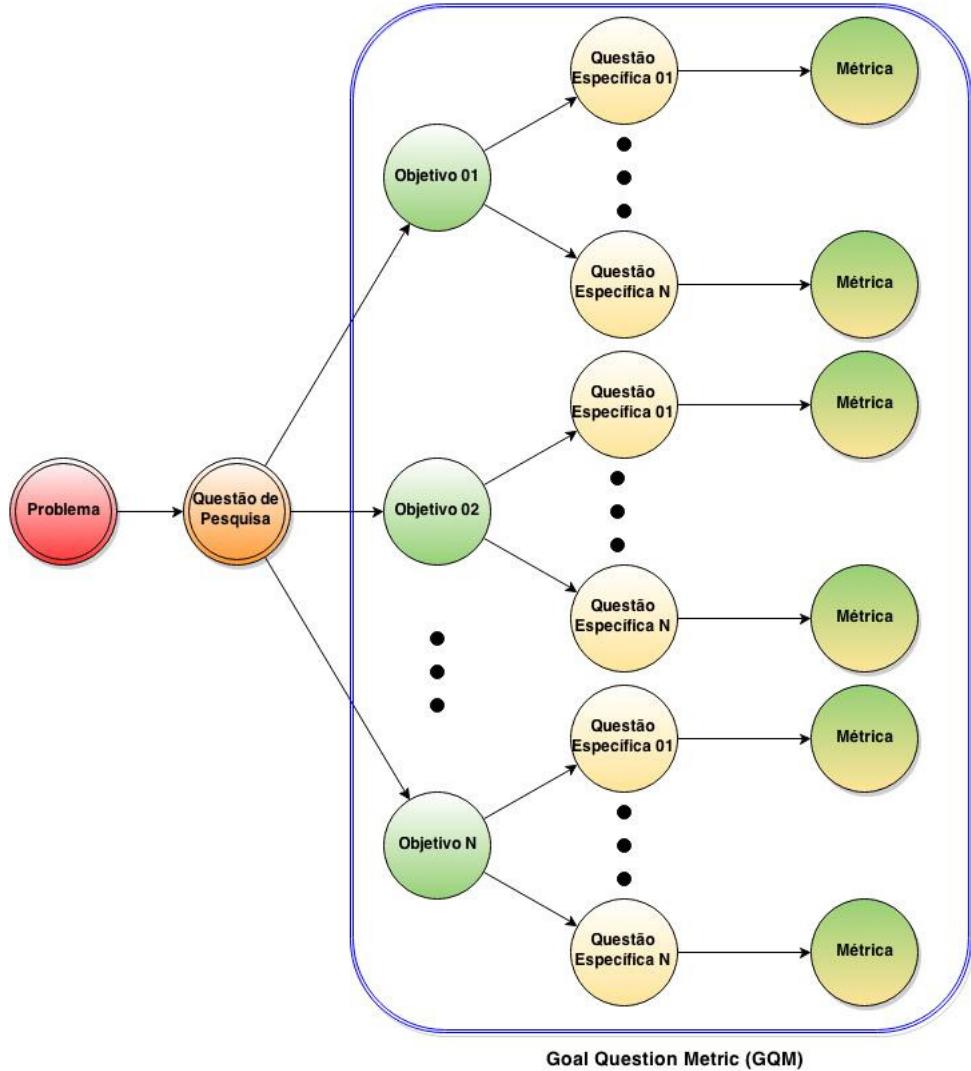


Figura 20 – Estrutura do Estudo de Caso

Segundo ([BASILI; CALDIERA; ROMBACH, 1996](#)), o resultado da aplicação do GQM é a especificação de um sistema de medida destinada a um conjunto específico de questões e um conjunto de regras para a interpretação dos dados da medição. O GQM é uma estrutura hierárquica que começa com um objetivo que especifica o propósito da medição, objeto a ser medido, questão a ser medida e o ponto de vista de quem a observa. O objetivo é refinado em várias questões, onde para cada questão é definida uma métrica(objetiva ou subjetiva)([BASILI; CALDIERA; ROMBACH, 1996](#)). A estrutura do GQM está a seguir.

Objetivo 01: Realizar um estudo de caso na CAIXA, onde foi investigada a oportunidade de uso da solução construída neste trabalho, apresentada na seção ??, para auxiliar as equipes da CAIXA em aferir a qualidade interna do produto de software contratado de um fornecedor, de forma a auxiliar à tomada de decisões.

QE01: Qual a opinião da equipe do CETEC/GITECGO quanto a detecção de

cenários de limpeza de código através da solução DW construída neste trabalho?

Fonte: Entrevista com membros das equipes da CETEC e da GITECGO.

Métrica: Ótimo, Muito Bom, Bom, Ruim, Muito Ruim, Não possui opinião.

QE02: Qual a opinião da equipe do CETEC/GITECGO quanto a detecção de bugs no código(*findbugs*) através da solução DW construída neste trabalho?

Fonte: Entrevista com membros das equipes da CETEC e da GITECGO.

Métrica: Ótimo, Muito Bom, Bom, Ruim, Muito Ruim, Não possui opinião.

QE03: Qual a opinião da equipe do CETEC/GITECGO quanto a detecção de violações de código(*PMD*) através da solução DW construída neste trabalho?

Fonte: Entrevista com membros das equipes da CETEC e da GITECGO.

Métrica: Ótimo, Muito Bom, Bom, Ruim, Muito Ruim, Não possui opinião.

QE04: Qual a opinião da equipe do CETEC/GITECGO quanto a visualização dos cenários de limpeza através do *dashboard*?

Fonte: Entrevista com membros das equipes da CETEC e da GITECGO.

Métrica: Ótimo, Muito Bom, Bom, Ruim, Muito Ruim, Não possui opinião.

QE05: Qual a opinião da equipe da CETEC/GITECGO quanto a visualização de violações de código(*PMD*) através do *dashboard*?

Fonte: Entrevista com membros das equipes da CETEC e da GITECGO.

Métrica: Ótimo, Muito Bom, Bom, Ruim, Muito Ruim, Não possui opinião.

QE06: Qual a opinião da equipe da CETEC/GITECGO quanto a visualização de bugs(*findbugs*) através do *dashboard*?

Fonte: Entrevista com membros das equipes do CETEC e da GITECGO.

Métrica: Ótimo, Muito Bom, Bom, Ruim, Muito Ruim, Não possui opinião.

QE07: Em relação a visualizações de erros e violações, qual a opinião da equipe da CETEC/GITECGO do uso do DW em comparação à solução do sonarQube?

Fonte: Entrevista com membros das equipes do CETEC e da GITECGO.

Métrica: Muito Melhor, Melhor, Imparcial, Pior, Muito Pior.

QE08: Existe alguma correlação entre *bugs*, violações e cenários analisados?

Fonte: Total de *bugs*, violações e cenários de limpeza encontrados pela solução de DW.

Métrica: Coeficiente de correlação Linear.

QE09: É possível provar a corretude dos resultados das análises?

Fonte: Valores das análises apresentados pela solução de DW e valores das análises apresentados pelas ferramentas FindBugs e PMD utilizadas individualmente.

Métrica: Erro quadrático médio.

5.3 Design

Segundo ([WOHLIN et al., 2012](#)), umas das formas de classificar um estudo de caso é com base na identificação de seus propósitos gerais: exploratórios, descritivos, explicativos ou avaliativos. Este estudo de caso é classificado como exploratório, pois não foi esperado obter uma resposta definitiva para o problema proposto. A ideia é obter uma visão mais acurada do problema para posteriormente realizar uma pesquisa mais aprofundada. Tal escolha foi feita porque o tema escolhido foi pouco explorado até o momento, constituindo apenas a primeira etapa de uma investigação mais ampla e sistemática.

Uma segunda forma de classificação é relacionada a quantidade de casos: caso único ou casos múltiplos. Considerando as restrições de tempo para realização deste estudo de caso, foi definido o estudo de um caso único, uma organização, e holístico com uma unidade de análise, um sistema.

5.4 Seleção

A organização pública selecionada para este estudo de caso foi a CAIXA Econômica Federal, cuja personalidade jurídica é de empresa pública. O estudo aconteceu mais especificamente na Centralizadora Nacional de Tecnologia da Informação (CETEC) e na Gerência de Filial de suporte Tecnológico de Brasília (GITECBR).

Atividades atribuídas à CETEC:

- **Suporte para o ambiente descentralizado**
- **Inventário de recurso no ambiente descentralizado**

- Desenvolvimento e produção de soluções no ambiente descentralizado
- Gestão de incidentes e mudanças no ambiente descentralizado

Atividades atribuídas à GITECBR:

- Gestão de incidentes tecnológicos de *software* e *hardware*
- Suporte tecnológico para canais e unidades da CAIXA
- Desenvolvimento de soluções e serviços tecnológicos de *software* e *hardware*
- Comunicação no ambiente descentralizado e externo.

A homologação dos serviços e a aceitação dos S&SC é facultado à CAIXA, submetendo os produtos de software produzidos pela CONTRATADA, à testes em software especializados para avaliação do desempenho dos mesmos. Atualmente, na CETEC, a solução em homologação, que é utilizada nas unidades da CAIXA no processo de monitoração de métricas de código-fonte adquiridos por empresas terceirizadas, faz uso da ferramenta SonarQube¹. Apesar do processo existente estar em fase de homologação, a definição de quais as métricas, indicadores e os seus valores de referência para interpretação, ainda não foi finalizado. Diante disso, as medidas que realmente são utilizadas para o ateste da qualidade interna do produto de *software* entregue estão relacionadas ao número de defeitos e ao índice aceitável de defeitos por ponto de função, conforme pode ser observado na Figura 21. O número de pontos de defeitos pode gerar multas para a CONTRATADA conforme o índice de defeitos referenciado na tabela 21. Contudo, a multa não exime a CONTRATADA das obrigações de corrigir os erros encontrados, onde ainda as alterações propostas pela CONTRATADA deverão ser efetuadas sem qualquer tipo de ônus financeiro ou a outro projeto para a CAIXA.

¹ <http://www.sonarqube.org/>

Fórmula de Índice de Defeitos	
N Tipos de defeitos	
Pd = $\left[\sum_{1}^{N} (\text{PSE} \times \text{Qtd Ocorrências}) + (\text{PRE} \times \text{Qtd Reincidência}) \right] / \text{Ts}$	
Onde:	
Pd — Pontos de Defeitos	
PSE — Peso da Severidade dos Erros	
PRE — Peso de Reincidência de Erros	
Ts — Tamanho do Serviço em Ponto de Função	

Severidade	Peso	Severidade na Reincidência	Peso
Altíssima	5	Altíssima	7
Alta	3	Alta	4
Média	2	Média	3
Baixa	1	Baixa	1

Figura 21 – Fórmula de Índice de Defeitos

Regras relacionadas à Fórmula de Índice de Defeitos:

- Para arredondamento do valor de “Pd” aplicar-se-á a seguinte regra: se o número constante na segunda casa decimal for superior ou igual a 5, o algarismo da primeira casa decimal será acrescido de 1. Caso contrário, o valor da primeira casa decimal permanece inalterado. (ex: se o resultado do cálculo for igual a 0,18, o valor passará a ser 0,2. Se o resultado do cálculo for igual a 0,13, o valor passará a ser 0,1).
- Caso o índice de defeitos fique superior ao aceitável, que é de 0,2 erro por ponto de função, sensibilizará o indicador de desempenho e aplicará o respectivo percentual de redução, conforme a tabela 21.

Percentual de Redução	Incidência
0.05%	Por erro gerado, a partir do limite tolerável de 0.2 erro por ponto de função.
0.10%	Por erro gerado, quando houve a devolução para acertos e a demanda foi entregue novamente com erros, independentemente do limite tolerável.

Tabela 21 – Multa por defeitos

Para o estudo de caso deste trabalho foi analisado o Sistema de Gestão Financeira de Bens e Serviços de Tecnologia da Informação (SIGET) um sistema do portfólio da filial GITECBR desenvolvido na linguagem java. Em cada *release* do projeto a GITECBR recebe o código-fonte do sistema da contratada. Antes da nova versão do sistema entregue ser implantada em ambiente de homologação de sistemas, ela deve ser aferida pela solução disponibilizada pela CETEC que faz uso do sonarQube. O sonarQube utilizado pela CETEC dispõe de *plugins* configurados de outras ferramentas para poder apresentar os defeitos contidos no código-fonte, dentre os *plugins* utilizados estão o Findbugs e o PMD. Com base nos defeitos e no índice aceitável de defeitos, que é de 0,2 erro por ponto de função, o gestor define se a *release* deve ser rejeitada e se existe alguma multa aplicável conforme a tabela 21.

5.5 O Software Analisado

O Projeto Gestão Financeira de Bens e Serviços de Tecnologia da Informação (SIGET) é um sistema web e foi criado a partir de uma necessidade de negócio da Gerência Nacional Gestão de Ativos de TI (GEGAT) para fornecer informações gerenciais e monitorar as atividades de execução físico-financeira dos contratos e contratação de Tecnologia da Informação da CAIXA.

O SIGET possui as funcionalidades de cadastramento de dados sobre as demandas e contratos de bens e serviços de TI e consulta a relatórios para acompanhamento dessas demandas e desses contratos. O acesso deve ser restrito a usuários autorizados e há níveis de permissão diferenciados.

Os subprodutos gerados constituem-se como artefatos do projeto e do sistema. Tal documentação é importante para a conformidade com os normativos e os controles institucionais da CAIXA e formalização dos acordos entre o gestor negocial e a área responsável pela solução tecnológica.

O SIGET é uma ferramenta auxiliar no aprimoramento da gestão de contratos de ativos de TI, reduzindo a utilização de planilhas e controles paralelos, que aumentam o risco operacional. A GEGAT solicitou o desenvolvimento do sistema em módulos para um melhor controle:

- Cadastros Auxiliares: Abrange as funcionalidades relacionadas à manutenção das informações gerais das demandas (cadastro principal);
- Demandas: Abrange as funcionalidades correspondentes à manutenção da demanda, sua reserva orçamentária e suas fases;
- Orçamento: Abrange as funcionalidades que tratam da solicitação, aprovação e autorização de orçamento;

- Rotina Automática: Referente à funcionalidade de concessão automática de alguns perfis de acesso a determinados empregados CAIXA.

5.6 Fonte dos Dados Coletados e Método de Coleta

Os dados deste estudo de caso foram coletados através de questionários, que está presente no Apêndice C, entrevistas e resultados gerados pela própria solução de *Data Warehousing*, construída neste trabalho, utilizando o código-fonte do sistema SIGET desenvolvido por uma empresa contratada pela CAIXA.

Os registros oriundos do questionários foram utilizados principalmente para dados qualitativos. O questionário traz indagações a respeito da solução de *Data Warehousing* pelo ponto de vista dos membros das equipes da CETEC e da GITECGO, no que diz respeito: à detecção de cenários de limpeza de código, detecção de *bugs*, detecção de violações, ao nível de satisfação da solução, ao nível de satisfação da visualização do *dashboard*.

Ao longo das análises das *releases* do SIGET foram coletados dados quantitativos utilizando a solução, tais como, a quantidade de Cenários de Limpeza total, por tipo e por *release*, a quantidade de *bugs* total, por tipo, por *index* e por *release* e a quantidade de violações total, por tipo, por *index* e por *release*. Além destes dados quantitativos também foram calculados o erro quadrático médio e o coeficiente de correlação, cuja as análises estão presentes na Seção 5.9.

5.7 Processo de análise dos dados

A análise de dados coletados durante o estudo de caso foi feita através de 4 etapas:

- **Categorização:** Organização dos dados em duas categorias - qualitativos e quantitativos. Os dados qualitativos referem-se aos questionários e entrevistas realizados. Os dados quantitativos, por sua vez, referem-se aos valores numéricos da solução de DW.
- **Exibição:** Consiste na organização dos dados coletados para serem exibidos através de gráficos, tabelas e texto para poderem ser analisados.
- **Verificação:** Atestar padrões, tendências e aspectos específicos dos significados dos dados. Procurando assim gerar uma discussão e interpretação de cada dado exibido.
- **Conclusão:** Agrupamento dos resultados mais relevantes das discussões e interpretações dos dados anteriormente apresentados.

5.8 Ameaças a validade do estudo de caso

Yin (2001) descreve como principais ameaças relacionadas à validade do estudo de caso aquelas relacionadas à validade de construção, à validade interna, à validade externa e à confiabilidade. As quatro ameaças definidas por ele, bem como a forma usada nesse trabalho para preveni-las, são descritas da seguinte maneira:

- **Validade de construção:** A validade de construção está presente na fase de coleta de dados, quando devem ser evidenciadas as múltiplas fontes de evidência e a coleta de um conjunto de métricas para que se possa saber exatamente o que medir e quais dados são relevantes para o estudo, de forma a responder as questões de pesquisa (YIN, 2001). O uso do GQM mitiga a validade de construção, uma vez que estabelece uma lógica entre a questão de pesquisa e as métricas que serão analisadas para respondê-la.
- **Validade interna:** Para Yin (2001), o uso de várias fontes de dados e métodos de coleta permite a triangulação, uma técnica para confirmar se os resultados de diversas fontes e de diversos métodos convergem. Dessa forma é possível aumentar a validade interna do estudo e aumentar a força das conclusões. A triangulação de dados se dará pelas medidas extraídas do código-fonte por meio da solução de *Data Warehousing* (explicada no capítulo 4), pela análise de questionários e pelos dados coletados através de entrevistas.
- **Validade externa:** Yin (2001) recomenda a replicação do estudo em múltiplos casos. Por esse ser um caso único, a generalização não poderá ser alcançada. Este trabalho é o primeiro a analisar a solução para o estudo de caso na CAIXA, portanto não há como correlacionar os resultados obtidos a nenhum outro estudo.
- **Confiabilidade:** Com relação a confiabilidade, Yin (2001) associa à repetibilidade, desde que seja usada a mesma fonte de dados. Nesse trabalho, o protocolo de estudo de caso apresentado nessa seção, além da disponibilização da base de dados coletados e analisados, que estão dispostos no repositório do *github*², e a corretude dos dados validada na Seção 5.9.1, garantem a repetibilidade desse trabalho e, consequentemente, a validade relacionada à confiabilidade.

5.9 Execução do Estudo de Caso e Análise dos Dados

Para cada uma das *releases* do SIGET, coletou-se o código-fonte, conforme a disponibilidade, no repositório da CAIXA. Após a obtenção do código-fonte, foi realizado a análise estática cada *release* utilizando as ferramentas analizo, findbugs e PMD. Pos-

² https://github.com/NiltonAraruna/tcc_dw_metricas

teriormente, os resultados das análises foram extraídos, transformados e carregados no ambiente de Data Warehousing proposto no Capítulo 4. Dessa forma, foi possível obter:

- O intervalo qualitativo para cada uma das Métricas de Código-Fonte por *index* e por *release*;
- Quantidade de Cenários de Limpeza total, por tipo e por *release*;
- Classe, nome e recomendação para cada cenário de limpeza;
- Quantidade de *bugs* total, por tipo, por *index* e por *release*;
- Classe, tipo, nome e linha para cada *bug*;
- Quantidade de violações total, por tipo, por *index* e por *release*;
- Classe, tipo, nome e linha para cada violação;

Como demonstrado nas Figuras 22, 23, 24 e 25.

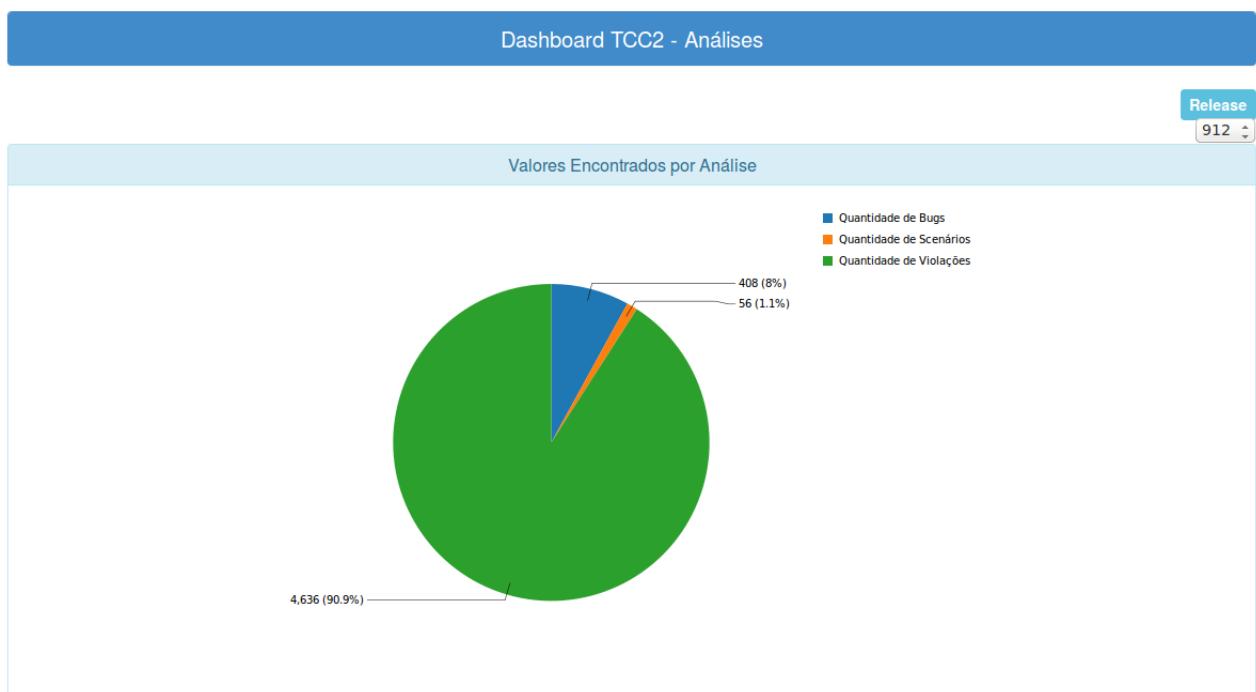


Figura 22 – Dashboard Quantidade Total



Figura 23 – Dashboard PMD

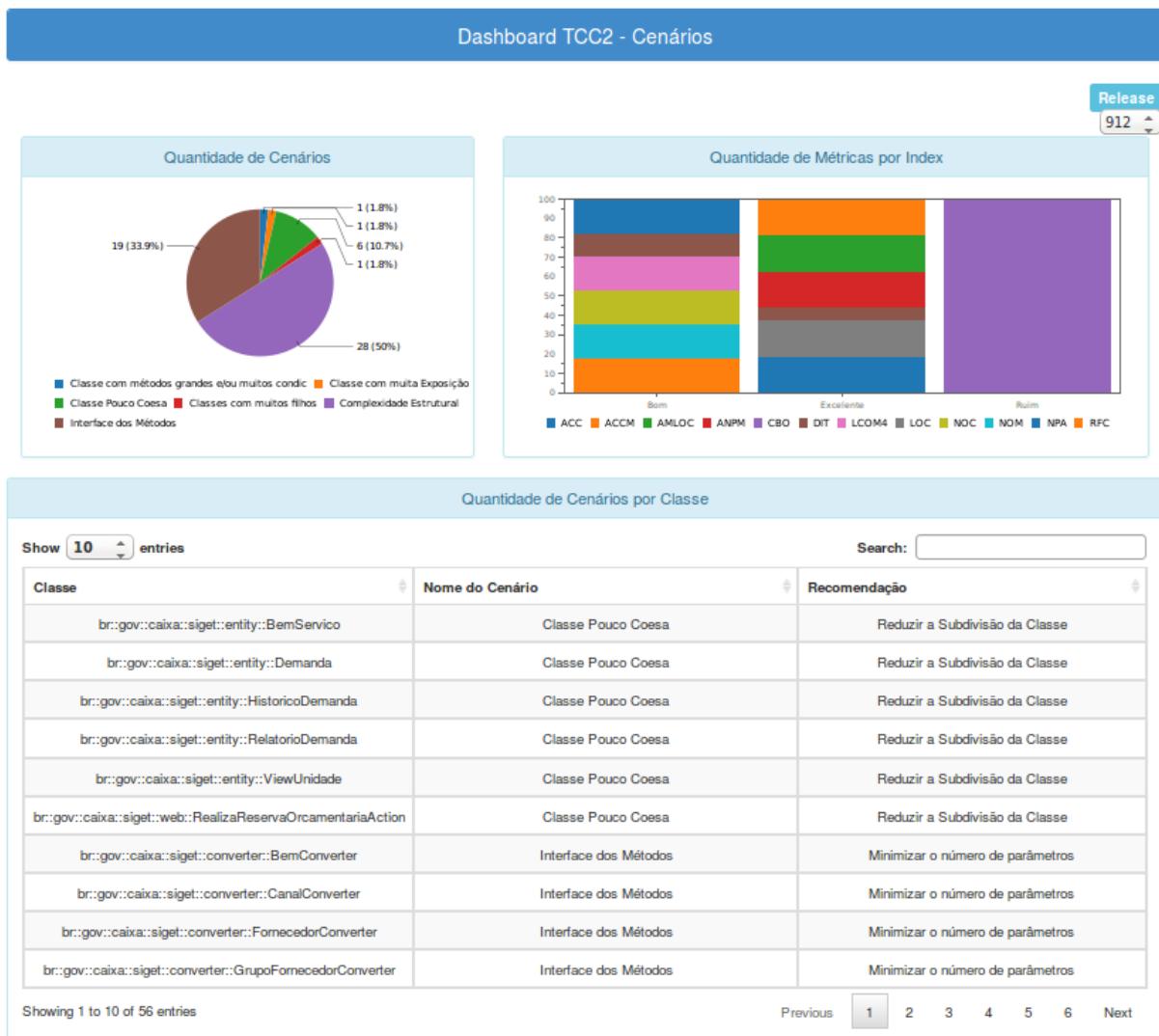


Figura 24 – Dashboard Cenários

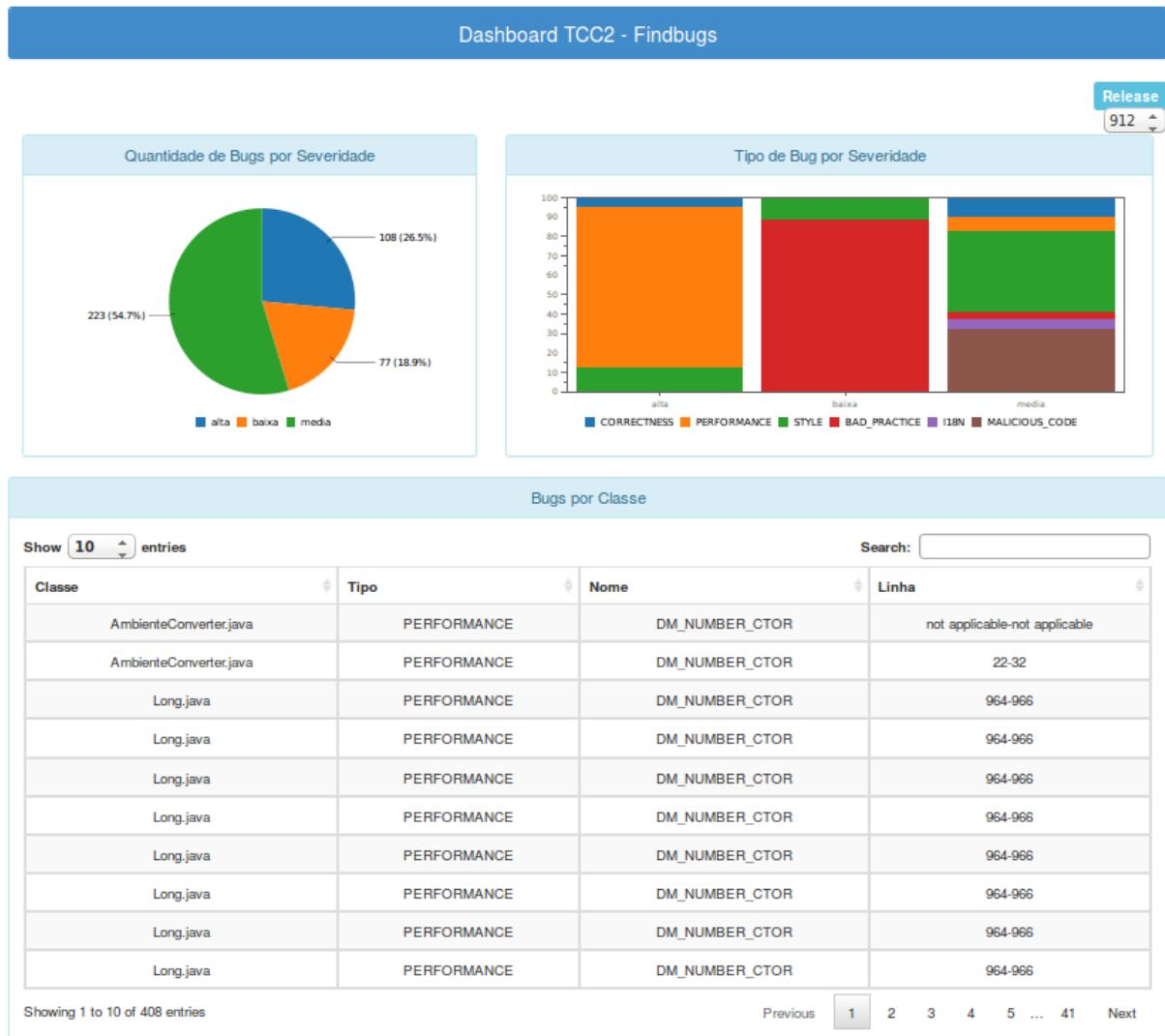


Figura 25 – Dashboard FindBugs

5.9.1 Análise do Erro Quadrático Médio

Com o objetivo de validar a corretude dos dados apresentados pela solução de DW, contruída neste trabalho, foi realizado o cálculo do EQM (Erro Quadrático Médio). EQM é a soma das diferenças entre o valor estimado e o valor real dos dados, ponderados pelo número de termos:

$$EQM = \sum_i \frac{(x_i - y_i)^2}{n}$$

Quanto mais próximo o valor do EQM de 0, maior a corretude e menor é a diferença entre os valores estimados e os valores reais.

Tomando como referência o valor estimado como os valores apresentados pela solução de DW e o valor real como os valores apresentados pelas ferramentas *FindBugs* e PMD utilizadas individualmente. Foi calculado o EQM de cada conjunto de regra do *FindBugs* e do PMD de todas as *releases*, as Tabelas 22, 23, 24 e 25 mostram os resultados

do EQM da maioria das *releases*, a tabela completa se encontra no repositório do *github*³. Os valores apresentados pelas ferramentas utilizadas individualmente foram coletados de arquivos do tipo csv por elas geradas, que estão disponíveis para consulta no repositório deste trabalho no *github*⁴. Os valores da solução de DW foram coletados a partir dos *Dashboards* da própria solução.

³ https://github.com/NiltonAraruna/tcc_dw_metricas/tree/master/ARQUIVOS%20/Atualizada/

⁴ https://github.com/NiltonAraruna/tcc_dw_metricas/tree/master/ARQUIVOS%20/Atualizada/analises

Releases	Dados	Por Tipo										Total		
		correct ness	<-EQM	per form	<-EQM	style	<-EQM	bad practice	<-EQM	118n	<-EQM			
912	FindBugs	28	0	104	0	117	0	76	0	11	0	72	0	408
	DW	28		104		117		76		11		72		408
914	FindBugs	28	0	104	0	112	0	76	0	11	0	72	0	403
	DW	28		104		112		76		11		72		403
915	FindBugs	35	0	104	0	120	0	76	0	11	0	72	0	418
	DW	35		104		120		76		11		72		418
916	FindBugs	35	0	104	0	123	0	80	0	11	0	72	0	425
	DW	35		104		123		80		11		72		425
919	FindBugs	39	0	114	0	125	0	82	0	11	0	72	0	443
	DW	39		114		125		82		11		72		443
927	FindBugs	39	0	118	0	121	0	82	0	11	0	72	0	443
	DW	39		118		121		82		11		72		443
928	FindBugs	39	0	118	0	121	0	82	0	11	0	72	0	443
	DW	39		118		121		82		11		72		443
931	FindBugs	39	0	118	0	123	0	82	0	11	0	72	0	445
	DW	39		118		123		82		11		72		445
938	FindBugs	39	0	104	0	119	0	80	0	11	0	72	0	425
	DW	39		104		119		80		11		72		425
940	FindBugs	39	0	107	0	118	0	89	0	11	0	72	0	436
	DW	39		107		118		89		11		72		436
967	FindBugs	39	0	121	0	140	0	79	0	11	0	72	0	462
	DW	39		121		140		79		11		72		462
968	FindBugs	61	0	152	0	154	0	84	0	17	0	72	0	540
	DW	61		152		154		84		17		72		540

Tabela 22 – Resultados do EQM do Findbugs

Releases	Dados	Por Tipo					
		SE	<-EQM	Design	<-EQM	Optimization	<-EQM
912	PMD	56	0	235	0	1708	0
	DW	56	235	235	1708	605	605
914	PMD	56	0	235	0	1708	0
	DW	56	235	235	1708	605	605
915	PMD	59	0	246	0	1774	0
	DW	59	246	246	1774	622	622
916	PMD	59	0	245	0	1791	0
	DW	59	245	245	1791	633	633
919	PMD	60	0	272	0	1931	0
	DW	60	272	272	1931	684	684
927	PMD	59	0	272	0	1933	0
	DW	59	272	272	1933	684	684
928	PMD	60	0	272	0	1932	0
	DW	60	272	272	1932	684	684
931	PMD	60	0	272	0	1933	0
	DW	60	272	272	1933	684	684
938	PMD	59	0	247	0	1851	0
	DW	59	247	247	1851	656	656
940	PMD	67	0	277	0	2011	0
	DW	67	277	277	2011	764	764
941	PMD	67	0	277	0	2011	0
	DW	67	277	277	2011	765	765
967	PMD	67	0	284	0	2074	0
	DW	67	284	284	2074	765	765
968	PMD	73	0	291	0	2295	0
	DW	73	291	291	2295	788	788
SIGLAS							

SE=Strict Exceptions ; SaS=String and StringBuffer ; JB=JavaBeans ; CS=Code Size

Tabela 23 – Resultados do EQM do PMD

Releases	Dados	Por Tipo					
		Basic	<-EQM	SE	<-EQM	Controversial	<-EQM
912	PMD	18	0	192	0	948	0
	DW	18		192		948	283
914	PMD	18	0	192	0	948	283
	DW	18		192		948	283
915	PMD	24	0	190	0	961	0
	DW	24		190		961	282
916	PMD	24	0	193	0	963	0
	DW	24		193		963	284
919	PMD	29	0	198	0	1051	0
	DW	29		198		1051	286
927	PMD	29	0	198	0	1051	0
	DW	29		198		1051	286
928	PMD	29	0	199	0	1051	0
	DW	29		199		1051	286
931	PMD	29	0	199	0	1052	0
	DW	29		199		1052	286
938	PMD	25	0	195	0	984	0
	DW	25		195		984	284
940	PMD	35	0	201	0	1061	0
	DW	35		201		1061	288
941	PMD	35	0	201	0	1063	0
	DW	35		201		1063	289
967	PMD	35	0	201	0	1109	0
	DW	35		201		1109	300
968	PMD	41	0	237	0	1170	0
	DW	41		237		1170	318
SIGLAS	SE=Strict Exceptions ; SaS=String and StringBuffer ; JB=JavaBeans ; CS=Code Size						

Tabela 24 – Continuação dos resultados do EQM do PMD

Releases	Dados	Por Tipo						<-EQM	TOTAL
		EC	<-EQM	Braces	<-EQM	IS	<-EQM		
912	PMD	10	0	113	0	25	0	200	4636
	DW	10		113		25		200	4636
914	PMD	10	0	113	0	25	0	200	4636
	DW	10		113		25		200	4636
915	PMD	18	0	114	0	25	0	206	4761
	DW	18		114		25		206	4761
916	PMD	18	0	115	0	33	0	207	4807
	DW	18		115		33		207	4807
919	PMD	18	0	124	0	30	0	209	5182
	DW	18		124		30		209	5182
927	PMD	18	0	124	0	31	0	211	5183
	DW	18		124		31		211	5183
928	PMD	18	0	124	0	31	0	211	5184
	DW	18		124		31		211	5184
931	PMD	18	0	124	0	31	0	210	5189
	DW	18		124		31		210	5189
938	PMD	18	0	120	0	25	0	208	4930
	DW	18		120		25		208	4930
940	PMD	22	0	124	0	21	0	209	5385
	DW	22		124		21		209	5385
941	PMD	22	0	124	0	121	0	209	5489
	DW	22		124		121		209	5489
967	PMD	22	0	124	0	21	0	209	5521
	DW	22		124		21		209	5521
968	PMD	20	0	124	0	38	0	216	5960
	DW	20		124		38		216	5960
SIGLAS									

EC=Empty Code; IS=Import Statements

Tabela 25 – Continuação dos resultados do EQM do PMD

A Figura 26 e as Tabelas 26, 27 e 28 exemplificam como os dados foram coletados para o cálculo do EQM. A Figura 26 demonstra a quantidade de violações do tipo *Basic* da release 914 no dashboard do PMD, já as Tabelas 26, 27 e 28 demonstram a quantidade de violações do tipo *Basic* da mesma release no arquivo gerado pela ferramenta PMD.

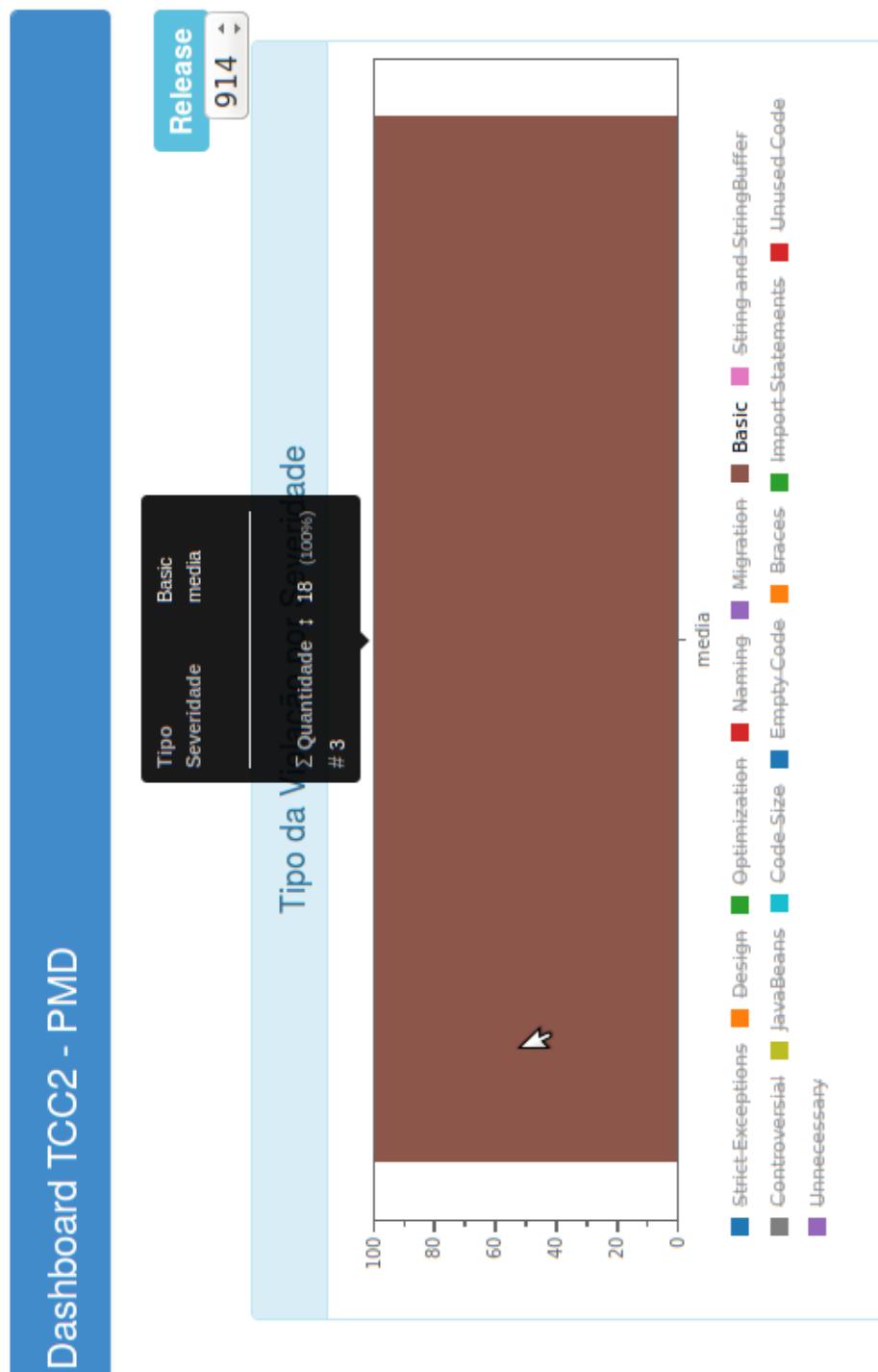


Figura 26 – Exemplo dos dados coletados a partir do *dashboard*

Package	File	Prio- rity	Line	Description	Rule set	Rule
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/AmbienteConverter.java	3	15	Each class should declare at least one constructor	Basic	AtLeastOneConstructor
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/AmbienteConverter.java	3	18	Avoid excessively long variable names like manterAmbienteBean	Basic	LongVariable
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/AmbienteConverter.java	3	18	Found non-transient, non-static member. Please mark as transient or provide accessors.	Basic	BeanMembersShouldSerialize
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/AmbienteConverter.java	3	21	Parameter 'facesContex' is not assigned and could be declared final	Basic	MethodArgumentCouldBeFinal
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/AmbienteConverter.java	3	21	Parameter 'ui-Component' is not assigned and could be declared final	Basic	MethodArgumentCouldBeFinal
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/AmbienteConverter.java	3	21	Parameter 'valor' is not assigned and could be declared final	Basic	MethodArgumentCouldBeFinal
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/AmbienteConverter.java	2	25	Avoid instantiating Long objects.Call Long.valueOf() instead	Basic	LongInstantiation

Tabela 26 – Exemplo dos dados coletados a partir do arquivo gerado pela ferramenta PMD

br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/Ambiente-Converter.java	3	27	A method should have only one exit point, and that should be the last statement in the method	Basic	OnlyOneReturn
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/Ambiente-Converter.java	3	36	Parameter 'facesContext' is not assigned and could be declared final	Basic	MethodArgumentCouldBeFinal
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/Ambiente-Converter.java	3	36	Parameter 'ui-Component' is not assigned and could be declared final	Basic	MethodArgumentCouldBeFinal
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/Ambiente-Converter.java	3	36	Parameter 'valor' is not assigned and could be declared final	Basic	MethodArgumentCouldBeFinal
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/Ambiente-Converter.java	3	38	Local variable 'ambiente' could be declared final	Basic	LocalVariableCouldBeFinal
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/Ambiente-Converter.java	3	39	A method should have only one exit point, and that should be the last statement in the method	Basic	OnlyOneReturn

Tabela 27 – Continuação exemplo dos dados coletados a partir do arquivo gerado pela ferramenta PMD

br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/BemConverter.java	3	12	Each class should declare at least one constructor	Basic	AtLeastOneConstructor
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/BemConverter.java	3	14	Found non-transient, non-static member. Please mark as transient or provide accessors.	Basic	BeanMembersShouldSerialize
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/BemConverter.java	3	14	This final field could be made static	Basic	FinalFieldCouldBeStatic
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/BemConverter.java	3	17	Parameter 'facesContex' is not assigned and could be declared final	Basic	MethodArgumentCouldBeFinal
br.gov.caixa.siget.converter	siget_914/src/br/gov/caixa/siget/converter/BemConverter.java	3	18	Parameter 'ui-Component' is not assigned and could be declared final	Basic	MethodArgumentCouldBeFinal

Tabela 28 – Continuação exemplo dos dados coletados a partir do arquivo gerado pela ferramenta PMD

Como pode ser observado os resultados obtidos do EQM foram todos igual a 0, significando a corretude entre os valores obtidos através do *FindBugs* e PMD e os valores apresentados pela solução de DW. A corretude dos valores dos cenários de limpeza não foram validados porque algumas correções e adaptações tiveram que ser feitas na solução de DW de ([RêGO, 2014](#)) para que possilitasse as análises do SIGET, como a adição das função *ROUND* no step "Inserindo os Fatos na *FprojectMetric*" da transformação "Percentiles Analizo".

5.9.2 Análise do Coeficiente de Correlação

Segundo ([WASSERMAN, 2010](#)), o coeficiente de correlação linear r , proposto por Karl Pearson, é calculado a partir de uma amostra de n pares de observações de X e Y , e mede a intensidade e a direção da relação linear entre duas variáveis quantitativas e o grau de correlação entre as variáveis:

$$r = \frac{\sum(X-X')(Y-Y')}{\sqrt{[\sum(X-X')^2][\sum(Y-Y')^2]}}$$

Correlação entre duas variáveis é quando uma delas está, de alguma forma, relacionada com a outra. A Tabela 30 mostra a significância de índice de Correlação segundo ([WASSERMAN, 2010](#)).

Índice de correlação r	Significância
-1	Máxima correlação inversa
-0,99 a >0,7	Correlação inversa muito forte
-0,7 a <0,0	Correlação inversa fraca
-0,0	Inexistência de correlação
>0,0 a 0,7	Correlação direta fraca
>0,7 a -0,9	Correlação direta muito forte
+1	Máxima correlação direta

Tabela 29 – Significância de índice de Correlação

Um diagrama de dispersão também demonstra a relação entre duas variáveis quantitativas, medidas sobre os mesmos indivíduos. Se a reta do gráfico for mais próxima de uma reta de 45 graus, maior será a correlação positiva, se a reta apresentada for mais próxima de 135 graus, maior será a correlação negativa. [Assis \(2009\)](#)

Não tivemos a pretensão generalizar o resultado de uma possível correlação entre *bugs*, violações e cenários de limpeza por se tratar de um estudo de apenas um estudo de caso, uma instituição e um sistema. Entretanto, queríamos, para este caso, investigar a correlação linear entre *bugs* e violações, entre *bugs* e cenários de limpeza e entre cenários de limpeza e violações. O resultado do cálculo do coeficiente de correlação linear pode ser visto na Tabela 30.

Releases	Total de Cenários	Total de Bugs	Total de Violações	Correlação entre Cenários e Bugs	Correlação entre Cenários e Violações	Correlação entre Violações e Bugs
912	56	408	4636			
914	56	403	4636			
915	58	418	4761			
916	57	425	4807			
917	57	425	4799			
918	57	443	5159			
919	62	443	5182			
927	62	443	5183			
928	62	443	5184			
931	62	445	5189			
938	60	425	4930			
940	63	436	5385			
941	63	438	5489			
953	65	462	5521			
967	65	462	5521			
968	66	540	5960			

Tabela 30 – Resultado do cálculo do coeficiente de correlação linear

Analizando os resultados do coeficiente de correlação linear em conformidade com a Tabela 30 podemos notar que a correlação entre cenários de limpeza e violações, entre violações e *bugs* e correlação entre cenários e *bugs* é uma correlação direta e muito forte. Por meio dos gráficos de dispersão observados nas figuras 27, 28 e 29 também é possível evidenciar uma correlação direta e muito forte pois as retas dos gráficos se aproximam de 45 graus.

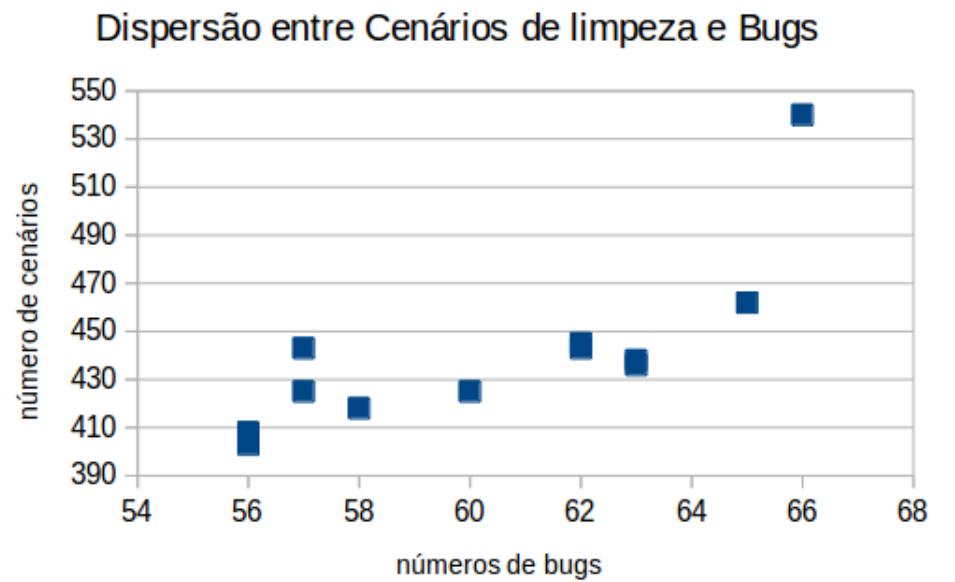


Figura 27 – Gráfico de Dispersão entre Cenários de Limpeza e *Bugs*

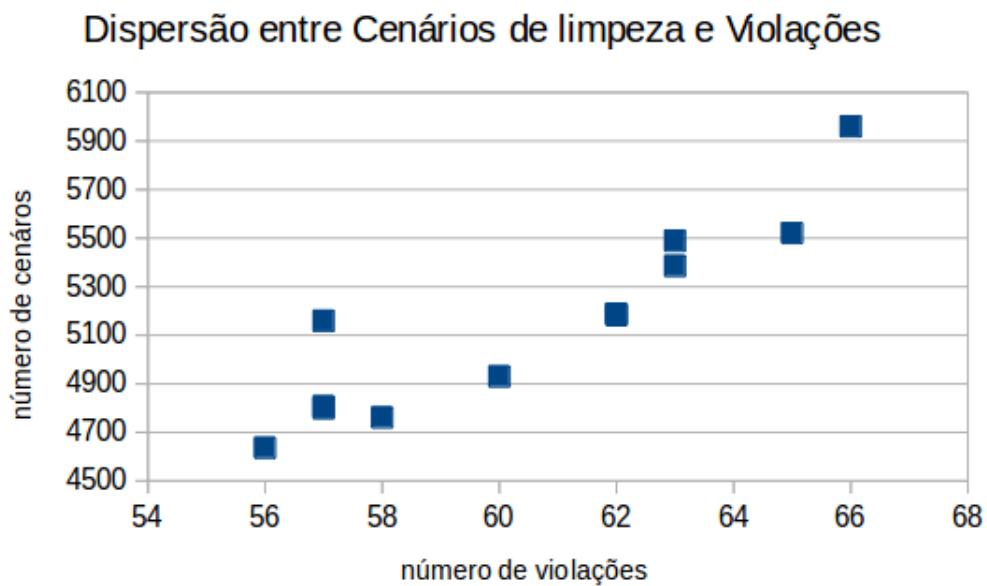


Figura 28 – Gráfico de Dispersão entre Cenários de Limpeza e Violações

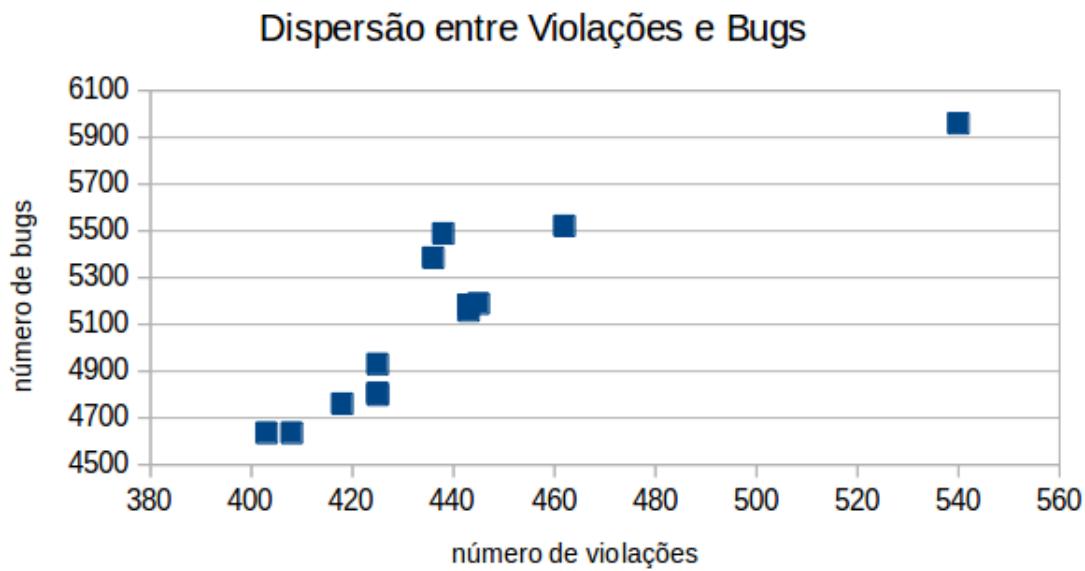


Figura 29 – Gráfico de Dispersão entre Violações e *Bugs*

5.9.3 Analise do Questionário

O questionário aplicado pode ser encontrado no Apêndice C. Vale ressaltar que esse questionário foi aplicado não só para caracterização do conhecimento sobre qualidade dos envolvidos do estudo de caso na CAIXA como também para coleta de dados qualitativos necessários para responder questões específicas do GQM.

Das 15 pessoas que realizaram o questionário todas trabalham na CETEC, sendo 4 do núcleo de inovação, 2 do núcleo qualidade, 3 do núcleo de governança, 3 do núcleo de banco de dados, 1 analista, 1 coordenador e 1 gerente de projetos. Do total 46,7% trabalham na mesma área de atuação entre 3 a 5 anos, 33,3% a mais de 5 anos e 20% a menos de 2 anos.

Em relação a QE01 e QE04: 20% definiram como Ótimo, 73,3% como Muito Bom e 6.7% como sem opinião. Sendo que 80% dos entrevistados entenderam o conceito sobre cenários de limpeza.

Em relação a QE02: 53.3% definiram como imparcial, 26.7% Melhor e 20% Muito Melhor a solução de DW em relação ao FindBugs. Sendo que 66.7% nunca utilizaram a ferramenta FindBugs.

Em relação a QE03: 80% definiram como imparcial, 13.3% Melhor e 6.7% Muito Melhor a solução de DW em relação ao PMD. Sendo que 99.3% nunca utilizaram a ferramenta PMD.

Em relação a QE05 e QE06: 40% definiram como Ótimo, 46.7% Muito Bom e 13.3% Bom a opinião quanto a visualização das análises através do *dashboard* da solução

de DW.

Em relação a QE07: 26.7% definiram como Melhor e 76.3% como Imparcial. Sendo que 73.3% nunca utilizaram a ferramenta sonarQube.

Segundo os respondentes, onde houve consideração aberta, para os sujeitos do estudo, os pontos fortes da solução de DW, levantados via questionário, foram:

- A forma prática na apresentação dos *bugs*;
- Apresentação de Cenários de limpeza;
- Apresentação de informações de forma direcionada;
- Simplicidade na visualização;
- Apresentar de forma gráfica os *bugs*, violações e cenários de limpeza detectados e auxiliar na tomada de decisões;
- Possibilidade de trabalhar com grande diversidade de dados;
- Concentrar os dados em um local proporcionando novas análises;
- Permitir a prevenção dos sistemas já em produção e agregar qualidade nas aferições por parte da empresa pública;
- Uma visão gerencial, Ferramental que auxilia na tomada de decisões;
- Traduz de forma simples os resultados das análises, facilitando o dia a dia do usuário;
- Dar visibilidade quanto a qualidade dos produtos entregues e auxiliar na tomada de decisões.

Pontos fracos da solução de DW levantados pelo questionário:

- Caso precise, a manutenção seria custosa e teria que ter uma equipe com conhecimentos técnicos sobre a solução;
- Recomendações para violações e *bugs*;
- Layout mais atrativo e recomendações conceituais;
- Tempo necessário para visualização das análises;
- Detalhar melhor as recomendações para os cenários de limpeza;
- Necessidade de outras ferramentas para fazer as análises.

De todos que responderam o questionário, 100% definiram que a solução de DW traria benefícios para a CAIXA, benefícios apontados:

- Com a visualização de *bugs*, violações e cenários de limpeza a Caixa teria maior visibilidade da qualidade dos softwares que são entregues pelas tercerizadas;
- Agilidade na validação e refatoração do código;
- Melhora na identificação do nível de qualidade dos produtos entregues pelas fábricas de software;
- Apoio na devolução de entregas com inconsistências para a fábrica e na aplicação de multas.
- Visualização do Percentual de *bugs*, violações e cenários de limpeza encontrados nas aplicações;
- Apresenta maior quantidade de informações que outras ferramentas;
- A solução apoia a melhoria contínua da qualidade;
- Automação da validação de código-fonte;
- O ganho da qualidade do software entregue
- Agregar qualidade nos recebimentos.
- Facilidade na visualização e acesso a dados importantes para a refatoração;
- Apoio na aferição de qualidade dos projetos desenvolvidos pelas fábricas que hoje está abaixo da média esperada.

5.10 Considerações finais do capítulo

Esse capítulo teve como objetivo apresentar o protocolo de estudo de caso que foi adotado neste trabalho, sua execução, as análises dos dados quantitativos e qualitativos e a discussão dos resultados. No próximo capítulo será apresentada a conclusão, limitações e trabalhos futuros.

6 Conclusão

Neste trabalho, foi apresentado o problema do processo de aferição da qualidade interna dos produtos de *software* desenvolvidos por terceirizadas em órgãos públicos brasileiros. Para realizar uma investigação empírica deste problema, com vistas à proposição de uma solução, foi realizado um estudo do uso de uma solução de *Data Warehouse*, construída neste trabalho, para o monitoramento de métricas, cenários de limpeza, bugs e violações de código-fonte no contexto das unidades da CETEC e GITECBR da CAIXA Econômica Federal.

A solução de DW proposta por [Rêgo \(2014\)](#) foi utilizada, adaptada e evoluída neste trabalho com o objetivo de aumentar os domínios de métricas de código-fonte, utilizadas na aferição da qualidade interna de *software* monitorando métricas, cenários de limpeza, bugs e violações de código-fonte. Também foi construído um *dashboard* gerencial visando dar maior visibilidade das medidas analisadas pela solução.

Antes da solução de DW ser apresentada, foi realizada uma revisão bibliográfica a respeito de qualidade interna de software, de métricas, de *Data Warehouse* e acerca da definição de estudo de caso. Os conceitos levantados na revisão bibliográfica foram apresentados com o objetivo de facilitar na compreensão da solução adotada. Foi apresentada a plataforma da solução, quer sejam: o ambiente e as ferramentas utilizadas na solução de DW.

Também foi exibido o projeto de pesquisa do estudo realizado. Conceitos fundamentais do projeto de pesquisa, como qual o problema a ser resolvido e a questão que o caracteriza, foram identificados e apresentados. Em seguida, utilizou-se da técnica GQM para, através de objetivos, questões específicas e métricas, responder a questão de pesquisa. As etapas pelas quais o estudo de caso atravessou foram descritas e modeladas de forma sequencial, de tal forma a obedecer a revisão bibliográfica sobre o estudo de caso.

Por fim, foi exibido a parte referente a execução do estudo de caso, coleta e análise dos dados obtidos. A execução, coleta e a análise dos dados serviram para evidenciarmos a corretude e a correlação dos dados apresentados pela solução de *Data Warehousing* monitorando as métricas, cenários de limpeza, bugs e violações de código-fonte do sistema SIGET, adquirido pela CAIXA, com o objetivo de responder a questão de pesquisa definida.

Para responder a questão de pesquisa foi elaborado uma análise da corretude e correlação dos dados apresentados pela solução de DW. O erro quadrático entre os dados apresentados pela solução e os dados apresentados pelas ferramentas findbugs e PMD quando utilizadas individualmente. O valor do erro quadrático foi igual a zero para

todas as análises, indicando a corretude dos dados apresentados pela solução de DW. As ferramentas findbugs e PMD são ferramentas reconhecidas e já validadas pela indústria de software. A corretude entre os dados apresentados por estas ferramentas e a solução de DW foi uma das validações da solução construída neste trabalho.

A análise do coeficiente de correlação, visou investigar a existência de uma possível correlação entre os cenários de limpeza e bugs, bugs e violações e entre cenários de limpeza e violações. Os resultados do coeficiente de correlação indicaram uma correlação forte entre todos, porém, por se tratar de uma análise realizado em apenas uma organização e em apenas um sistema, esta análise não pode ser generalizada.

Por a solução não ter sido utilizada em uma situação real em um processo de desenvolvimento da CAIXA, foi elaborado um questionário visando dar um maior suporte para a validação científica da solução, além de servir como fonte das métricas das questões específicas do GQM. O questionário foi respondido por quinze funcionários da equipe CETEC da CAIXA, envolvendo membros de todos os núcleos: inovação, qualidade, governança, banco de dados, analista, um coordenador e um gerente de projetos. Todos os quinze funcionários responderam que a solução traria benefícios para a CAIXA e destacaram pontos fortes da solução de DW.

Após todas estas análises ficou mais claro os indícios dos benefícios que a solução traria para CAIXA, podendo ser utilizada como apoio para a homologação dos serviços e a aceitação dos S&SC fornecidos pelas contratadas.

Quanto aos trabalhos futuros, a possibilidade imediata seria a utilização real da solução de DW apresentada neste trabalho, durante todo o processo de desenvolvimento da CAIXA, objetivando-se na validação e refinamentos do protocolo deste estudo de caso, e em seguida, a ampliação deste estudo para outras organizações públicas brasileiras.

6.1 Limitações

A maior limitação deste trabalho foi o fato da solução de DW apresentada não ter sido utilizada em uma situação real devido ao tempo gasto para a implementação da solução.

Como apresentado anteriormente, na Seção 5.8, Yin (2001) recomenda a replicação do estudo em múltiplos casos. Por esse ser um caso único, a generalização não poderá ser alcançada. Este trabalho é o primeiro a analisar a solução para o estudo de caso na CAIXA, portanto há a limitação por não ter como correlacionar os resultados obtidos a nenhum outro estudo.

Também cabe destacar que o ambiente de Data Warehousing, especificado neste trabalho, é dependente das ferramentas de análise estática de código-fonte utilizadas.

Portanto, é dependente dos dados providos por essas ferramentas. O FindBugs, que foi uma das ferramentas escolhida, limita-se a linguagem de programação Java, por exemplo.

Referências

- ASSIS, P. O. A. de. *Análise da Correlação entre Métricas de Evolução e Qualidade de Design de Software*. Universidade Federal de Campina Grande: [s.n.], 2009. Citado na página 100.
- BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. *The Goal Question Metric Approach*. [S.l.]: Encyclopedia of Software Engineering, 1996. Citado na página 78.
- BASILI, V. R.; ROMBACH, H. D. *TAME: Integrating Measurement into Software Environments*. 1987. Disponível em: <<http://drum.lib.umd.edu//handle/1903/7517>>. Citado na página 30.
- BECK, K. *Extreme Programming Explained*. [S.l.]: Addison Wesley, 1999. Citado na página 18.
- BECK, K. *Implementation Patterns*. 1. ed. [S.l.]: Addison-Wesley Professional, 2007. Citado na página 35.
- BRERETON, P.; KITCHENHAM, B.; BUDGEN, D. Using a protocol template for case study planning. In: *Proceedings of EASE 2008*. [S.l.]: BCS-eWiC, 2008. Citado na página 75.
- BUDD, T. *An introduction to object-oriented programming*. 3rd ed. ed. Boston: Addison-Wesley, 2002. ISBN 0201760312. Citado na página 30.
- CDE The Community Dashboard Editors. 2014. <<http://www.webdetails.pt/ctools/cde>>. Accessed: 2015-02-10. Citado na página 60.
- CRUZ, C.; ANDRADE, E.; FIGUEIREDO, R. *Processo de contratação de Serviços de Tecnologia da Informação para Organizações Públicas*. [S.l.]: PBPQ Software, 2011. Citado na página 46.
- FENTON, N. E.; PFLEGER, S. L. *Software Metrics: A Rigorous and Practical Approach*. 2 edition. ed. [S.l.]: Course Technology, 1998. 656 p. Citado 2 vezes nas páginas 29 e 77.
- FEW, S. *Information dashboard design: the effective visual communication of data*. 1st ed. ed. Beijing ; Cambride [MA]: O'Reilly, 2006. ISBN 0596100167 9780596100162. Citado na página 58.
- FOWLER, M. *Refactoring: improving the design of existing code*. [S.l.]: Addison-Wesley Professional, 1999. Citado na página 18.
- GIL, A. C. *Como elaborar projetos de pesquisa*. [S.l.]: Atlas, 2002. ISBN 9788522431694 8522431698. Citado na página 20.
- GOLFARELLI, M. *Data warehouse design: modern principles and methodologies*. New York: McGraw-Hill, 2009. ISBN 9780071610391. Citado 4 vezes nas páginas 10, 55, 56 e 57.

- GUTIÉRREZ, A.; MAROTTA, A. An overview of data warehouse design approaches and techniques. 2000. Citado na página 54.
- HARMAN, M. Why source code analysis and manipulation will always be important. In: IEEE. *Source Code Analysis and Manipulation (SCAM), 2010 10th IEEE Working Conference on*. [S.I.], 2010. Citado na página 29.
- HONGLEI, T.; WEI, S.; YANAN, Z. The research on software metrics and software complexity metrics. In: . IEEE, 2009. ISBN 978-1-4244-5422-8, 978-0-7695-3930-0. Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5385114>>. Citado na página 29.
- HOVEMEYER, D.; PUGH, W. Finding bugs is easy. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 39, n. 12, p. 92–106, dez. 2004. ISSN 0362-1340. Disponível em: <<http://doi.acm.org/10.1145/1052883.1052895>>. Citado na página 43.
- IN4. Instrução normativa número 4. 2014. Citado 2 vezes nas páginas 18 e 48.
- INMON, W. H. *Building the Data Warehouse*. 3rd. ed. New York, NY, USA: John Wiley & Sons, Inc., 2002. Citado 2 vezes nas páginas 51 e 77.
- ISO/IEC 15939. *ISO/IEC 15939: Software Engineering - Software Measurement Process*. [S.I.], 2002. Citado 2 vezes nas páginas 18 e 27.
- ISO/IEC 15939. *ISO/IEC 12207: System and Software Engineering - Software Life Cycle Processes*. [S.I.], 2008. Citado 5 vezes nas páginas 10, 27, 46, 47 e 50.
- ISO/IEC 9126. *ISO/IEC 9126-1: Software Engineering - Product Quality*. [S.I.], 2001. Citado 2 vezes nas páginas 10 e 28.
- JELLIFFE, R. Mini-review of java bug finders. O'Reilly Developer Weblogs, mar. 2004. Disponível em: <<http://www.oreillynet.com/pub/wlg/4481>>. Citado na página 42.
- KAN, S. H. *Metrics and models in software quality engineering*. [S.I.]: Addison Wesley, 2002. Citado 3 vezes nas páginas 29, 31 e 77.
- KIMBALL, R. *The data warehouse lifecycle toolkit: expert methods for designing, developing, and deploying data warehouses*. [S.I.]: Wiley. com, 1998. Citado na página 56.
- KIMBALL, R.; ROSS, M. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. 2nd. ed. New York, NY, USA: John Wiley & Sons, Inc., 2002. ISBN 0471200247, 9780471200246. Citado 8 vezes nas páginas 10, 52, 53, 54, 61, 62, 63 e 77.
- LAIRD, M. C. B. L. M. *Software measurement and estimation: A practical approach*. [S.I.]: Wiley-IEEE Computer Society Press, 2006. Citado 2 vezes nas páginas 30 e 31.
- LEITE, J. C. Terceirização em informática sob a ótica do prestador de serviços. *Revista de Administração de Empresas*, v. 37, n. 4, 1997. Disponível em: <<http://www.scielo.br/pdf/rae/v37n4/a08v37n4>>. Citado na página 18.
- LOURIDAS, P. Static code analysis. *IEEE Softw.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 23, n. 4, p. 58–61, jul. 2006. ISSN 0740-7459. Disponível em: <<http://dx.doi.org/10.1109/MS.2006.114>>. Citado 2 vezes nas páginas 42 e 43.

- MACHINI, J. a. et al. *Código Limpo e seu Mapeamento para Métricas de Código-Fonte*. [S.l.]: Universidade de São Paulo, 2010. Citado 8 vezes nas páginas 12, 35, 36, 37, 38, 39, 40 e 77.
- MARINESCU, R. Measurement and quality in object-oriented design. In: IEEE. *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*. [S.l.], 2005. p. 701–704. Citado 2 vezes nas páginas 35 e 76.
- MARTIN, R. C. *Clean Code: A Handbook of Agile Software Craftsmanship*. [s.n.], 2008. 464 p. ISBN 9780132350884. Disponível em: <<http://portal.acm.org/citation.cfm?id=1388398>>. Citado na página 35.
- MCCABE, T. J. A Complexity Measure. *IEEE Transactions Software Engineering*, v. 2, n. 4, p. 308–320, December 1976. Citado 2 vezes nas páginas 29 e 77.
- MEIRELLES, P. R. M. *Monitoramento de métricas de código-fonte em projetos de software livre*. Tese (Doutorado) — Instituto de Matemática e Estatística – Universidade de São Paulo (IME/USP), 2013. Citado 7 vezes nas páginas 12, 27, 30, 31, 32, 33 e 76.
- MILLS, E. E. Metrics in the software engineering curriculum. *Ann. Softw. Eng.*, J. C. Baltzer AG, Science Publishers, Red Bank, NJ, USA, v. 6, n. 1-4, abr. 1999. ISSN 1022-7091. Citado 2 vezes nas páginas 28 e 29.
- NERI, H. et al. Aferição da qualidade do código-fonte com apoio de um ambiente de data warehousing na gestao de contrato ágil: um estudo de caso preliminar em uma autarquia da administracão plública federa. *WBMA*, n. 5, Novembro 2014. Citado 3 vezes nas páginas 8, 9 e 18.
- NERI, H. R. *Análise, Projeto e Implementação de um Esquema MOLAP de Data Warehouse utilizando SGBD-OR Oracle 8.1*. Universidade Federal da Paraíba - UFPB: [s.n.], 2002. Citado 4 vezes nas páginas 12, 55, 56 e 57.
- NOVELLO, T. C. *Uma abordagem de Data Warehouse para análise de processos de desenvolvimento de software*. phdthesis — Pontifícia Universidade Católica do Rio Grande do Sul, 2006. Disponível em: <<http://tardis.pucrs.br/dspace/handle/10923/1570>>. Citado na página 77.
- PUGH, W. Using static analysis to find bugs. *IEEE Softw.*, v. 25, n. 5, 2008. Disponível em: <<http://doi.ieeecomputersociety.org/10.1109/MS.2008.130>>. Citado na página 42.
- RêGO, G. B. Monitoramento de métricas de código-fonte com suporte de um ambiente de data warehousing: um estudo de caso em uma autarquia da administração pública federal. 2014. Disponível em: <<http://bdm.unb.br/handle/10483/8069>>. Citado 25 vezes nas páginas 10, 12, 33, 34, 36, 37, 38, 39, 40, 41, 51, 60, 61, 62, 63, 64, 65, 66, 67, 71, 72, 76, 99, 106 e 113.
- ROCHA, A. B. *Guardando Históricos de Dimensões em Data Warehouses*. Dissertação (Mestrado), Universidade Federal da Paraíba - Centro de Ciências e Tecnologia, 2000. Citado na página 51.
- RUIZ, D. D. A. et al. A data warehousing environment to monitor metrics in software development processes. In: *16th International Workshop on Database and Expert Systems Applications (DEXA 2005), 22-26 August 2005, Copenhagen, Denmark*. [S.l.]: IEEE Computer Society, 2005. p. 936–940. ISBN 0-7695-2424-9. Citado na página 77.

- SHARMA, N. *Getting started with data warehousing*. [S.l.]: IBM Redbooks, 2011. Citado 5 vezes nas páginas 51, 53, 54, 55 e 77.
- SILVA, E.; MENEZES, E. Metodologia da pesquisa e elaboração de dissertação. 2005. Disponível em: <https://projetos.inf.ufsc.br/arquivos/Metodologia_de_pesquisa_e_elaboracao_de_teses_e_dissertacoes_4ed.pdf>. Citado 3 vezes nas páginas 12, 22 e 23.
- SOFTEX. MPS. *BR-Guia de Aquisição*. [S.l.], 2013. Disponível em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_de_Implementacao_SV_Parte_2_20132.pdf>. Citado na página 18.
- SOFTEX. MPS BR-guia de implementação – parte 2: Fundamentação para implementação do nível f do mr-mps-sv:2012. 2013. Citado na página 27.
- SOMMERVILLE, I. et al. *Engenharia de software*. São Paulo: Pearson Prentice Hall, 2008. ISBN 9788588639287 8588639289. Disponível em: <http://www.worldcat.org/search?qt=worldcat_org_all&q=9788588639287>. Citado na página 42.
- TCU. *Acórdão-381/2011-TCU-Plenário*. [S.l.], 2011. Disponível em: <<https://contas.tcu.gov.br/juris/SvlHighLight?key=ACORDAO-LEGADO-89657&texto=2b4e554d41434f5244414f2533413338312b414e442b2b4e554d414e4f41434f5244414f25334132303131&sort=&ordem=&bases=ACORDAO-LEGADO;RELACAO-LEGADO;DECISAO-LEGADO;SIDOC;ACORDAO-RELACAO-LEGADO;>>. Citado na página 19.
- TIMES, V. C. *Sistemas de DW*. 2012. Universidade Federal de Pernambuco - UFPE. Disponível em: <www.cin.ufpe.br/~if695/bda_dw.pdf>. Citado 3 vezes nas páginas 10, 54 e 56.
- WASSERMAN, L. *All of Statistics: A Concise Course in Statistical Inference*. [S.l.]: Springer Publishing Company, Incorporated, 2010. ISBN 1441923225, 9781441923226. Citado na página 100.
- WILLCOCKS, L.; LACITY, M. *Global information technology outsourcing*. [S.l.: s.n.], 2001. Citado na página 18.
- WOHLIN, C. et al. *Experimentation in software engineering*. [S.l.]: Springer, 2012. Citado 2 vezes nas páginas 23 e 80.
- YIN, R. *Estudo de caso: planejamento e métodos*. [S.l.]: Bookman, 2001. Citado 5 vezes nas páginas 23, 74, 76, 86 e 107.

APÊNDICE A – Descrição do Processo de ETL no Kettle

Neste apêndice, será apresentado a implementação do ETL no Kettle, onde se utilizou dos arquivos CSV resultantes da análise de métricas de código-fonte do Analizo, FindBugs e PMD. No processo de ETL proposto por Rêgo (2014), os arquivos do tipo CSV obtidos do Analizo foram convertidos para JSON, neste trabalho apenas os arquivos obtidos pelo Analizo e PMD foram convertidos para JSON, os arquivos obtidos do FindBugs foram mantidos no formato CSV. Visando realizar a conversão, foi escrito uma pequena aplicação *web* na linguagem *Ruby*, disponível para download no repositório do *github*¹.

A.1 Implementações das *Transformations*

Como explicado anteriormente, o Kettle utiliza o componente de *Transformation* para realizar cálculos, consultas em tabelas, inserções em tabelas, leitura de dados e entre outros. Os componentes que foram utilizados na solução de DW são mostrados na Figura 30.

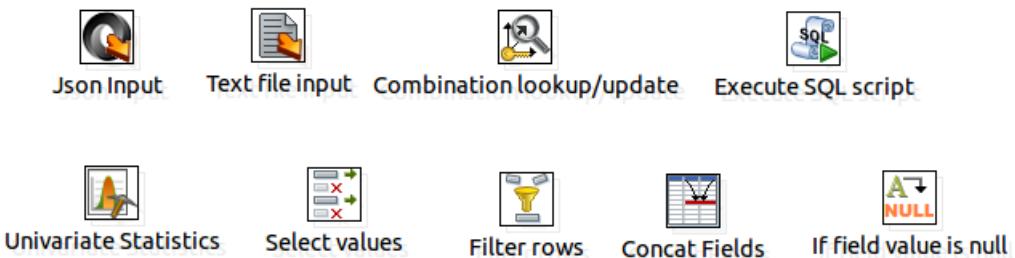


Figura 30 – Componentes do Kettle que foram utilizadas nas Transformações

O componente *JSON Input* serve para ler os dados provenientes de um arquivo JSON, em que o conteúdo de cada variável pode ser lido como: `..@nome_da_variavel`. O componente *Text file input* serve para o mesmo, porém para arquivos de outros tipos como o CSV. O componente *Univariate Statistics* é utilizado para se calcular estatísticas como média, mediana, número de amostras e percentis, que foram utilizados no cálculo dos intervalos percentis das métricas de código-fonte; O componente *Execute SQL Script* foi utilizado para recuperação de dados no Metadados e Dimensões e também para inserção de nas Tabelas Fatos; O componente *Combination Lookup* foi utilizado para se

¹ https://github.com/matheustristao/csvto_json

verificar se um determinado campo já existia em uma Dimensão, caso existisse, apenas se retornava o id da tupla, se não o inseria na Dimensão; O componente *Select Values* foi utilizado para filtrar os dados proveniente de outros componentes em uma *Transformation*; O componente *Filter rows* foi utilizado para que as linhas vazias dos arquivos fossem omitidas; O componente *Concat Fields* foi utilizado para juntar os campos *start* e *end* da análise do FindBugs para formar o campo *line* que é padrão das outras análises; Por fim o componente *if field is null* para rejeitar os valores de *line* em branco.

Na primeira transformação, como se mostra na Figura 31, obtém-se os dados provenientes do arquivo JSON com componente *JSON Input*. Em uma primeira etapa, coletava-se sobre dados: nome do projeto, release do arquivo, data de lançamento da release. Após a coleta dos dados do arquivo JSON, se inseria, conforme as verificações do componente *Combination Lookup*, nas dimensões correspondentes.

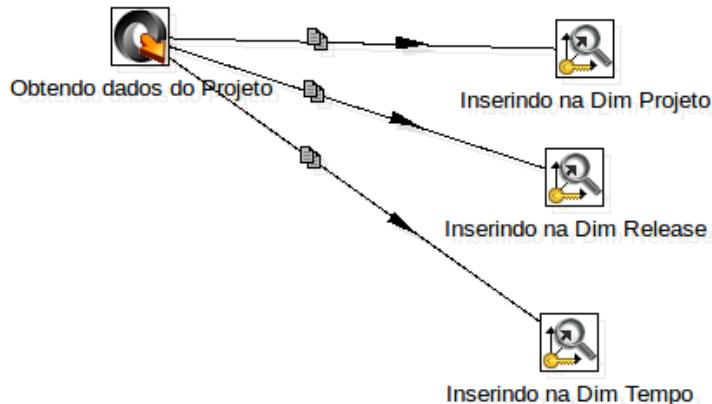


Figura 31 – Primeira Transformação realizada no Kettle

Na segunda transformação, como se mostra na Figura 32, cobre-se o processo de negócio de avaliação dos valores percentis das métricas de código-fonte do projeto em uma determinada *release* do software. Para tal, foi coletado os valores das métricas de código-fonte de cada classe com o componente *JSON Input*. Após a coleta dos valores das métricas, esses eram direcionados ao componente *Univariate Statistics* que realiza cálculos estatísticos. Após a realização dos cálculos, foi realizado um filtro com *Select Values* a fim de se obter apenas os percentis obtidos para cada uma das métricas.

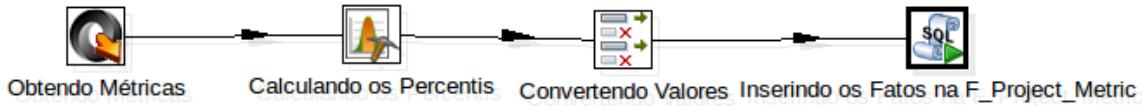


Figura 32 – Segunda Transformação realizada no Kettle

Por fim, foi realizado a avaliação dos valores percentis, em intervalos qualitativos nos metadados com o componente *Execute SQL Script*. Neste componente, recebeu-se o código-fonte descrito no Código-Fonte 1, onde cada ? foi substituído por uma variável dentro da *Transformation*.

```

1
2
3 SET @idProject = (SELECT max(idProject) from D_Project);
4
5 SET @idTime = (SELECT max(idTime) from D_Time);
6
7 SET @idRelease = (SELECT max(idRelease) from D_Release);
8
9 SET @Project_Language = (SELECT project_language from D_Project where
    idProject = @idProject);
10
11 SET @Best_Configuration = (SELECT idConfiguration FROM D_Configuration
    where configuration_name like '%Open JDK8 Metrics%' and
language_name LIKE CONCAT('%', @Project_Language, '%'));
12
13 SET @Worst_Configuration = (SELECT idConfiguration FROM D_Configuration
    where configuration_name like '%Tomcat Metrics%' and
language_name LIKE CONCAT('%', @Project_Language, '%'));
14
15
16
17
18
19 SET @idLOC = (SELECT idMetric FROM D_Metric where metric_abbreviation='
    LOC');
20
21
22
23 SET @qualityBestLOC = (SELECT idQuality FROM Meta_Metric_Ranges
    INNER JOIN D_Quality
ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
    language_name = 'java'
24 and metric_name='LOC' and ? <= max AND ? >= min and configuration_name
like '%Open JDK8 Metrics%');
25
26
27

```

```
28
29
30
31 SET @qualityWorstLOC = (SELECT idQuality FROM Meta_Metric_Ranges
32 INNER JOIN D_Quality
33 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index WHERE
34 language_name = @Project_Language
35 and metric_name='LOC' and ? <= max AND ? >= min and configuration_name
36 like '%Tomcat Metrics%');
37
38 INSERT INTO 'F_Project_Metric'
39 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
40 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
41 ')
42 VALUES
43 (?, @idProject, @idLOC, @qualityBestLOC, @Best_Configuration, @idRelease
44 , @idTime);
45
46
47
48 INSERT INTO 'F_Project_Metric'
49 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
50 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
51 ')
52 VALUES
53 (?, @idProject, @idLOC, @qualityWorstLOC, @Worst_Configuration,
54 @idRelease, @idTime);
55
56
57 SET @idACCM = (SELECT idMetric FROM D_Metric WHERE metric_abbreviation='
58 ACCM');
59
60
61 SET @qualityBestACCM = (SELECT idQuality FROM Meta_Metric_Ranges
62 INNER JOIN D_Quality
63 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index WHERE
64 language_name = @Project_Language
65 and metric_name='ACCM' and ? <= max AND ? >= min and configuration_name
```

```
like '%Open JDK8 Metrics%');

65
66
67
68
69 SET @qualityWorstACCM = (SELECT idQuality FROM Meta_Metric_Ranges
70 INNER JOIN D_Quality
71 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index WHERE
    language_name = @Project_Language
72 AND metric_name='ACCM' AND ? <= max AND ? >= min AND configuration_name
    like '%Tomcat Metrics%');

73
74
75 INSERT INTO 'F_Project_Metric'
76 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', '
    D_Quality_idQuality',
77 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
        ')
78 VALUES
79 (?, @idProject, @idACCM, @qualityBestACCM, @Best_Configuration,
    @idRelease, @idTime);

80
81
82
83
84
85 INSERT INTO 'F_Project_Metric'
86 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', '
    D_Quality_idQuality',
87 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
        ')
88 VALUES
89 (?, @idProject, @idACCM, @qualityWorstACCM, @Worst_Configuration,
    @idRelease, @idTime);

90
91
92
93 SET @idAMLOC = (SELECT idMetric FROM D_Metric WHERE metric_abbreviation=
    'AMLOC');

94
95
96
97 SET @qualityBestAMLOC = (SELECT idQuality FROM Meta_Metric_Ranges
98 INNER JOIN D_Quality
99 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index WHERE
    language_name = @Project_Language
100 AND metric_name='AMLOC' AND ? <= max AND ? >= min AND configuration_name
```

```
        like '%Open JDK8 Metrics%');

101
102
103
104
105 SET @qualityWorstAMLOC = (SELECT idQuality FROM Meta_Metric_Ranges
106 INNER JOIN D_Quality
107 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
108 language_name = @Project_Language
109 and metric_name='AMLOC' and ? <= max AND ? >= min and configuration_name
110 like '%Tomcat Metrics%');

111
112 INSERT INTO 'F_Project_Metric'
113 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', '
114     D_Quality_idQuality',
115 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
116 ')
117 VALUES
118 (?, @idProject, @idAMLOC, @qualityBestAMLOC, @Best_Configuration,
119 @idRelease, @idTime);

120
121
122 INSERT INTO 'F_Project_Metric'
123 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', '
124     D_Quality_idQuality',
125 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
126 ')
127 VALUES
128 (?, @idProject, @idAMLOC, @qualityWorstAMLOC, @Worst_Configuration,
129 @idRelease, @idTime);

130 SET @idACC = (SELECT idMetric FROM D_Metric where metric_abbreviation='
131 ACC');

132
133
134 SET @qualityBestACC = (SELECT idQuality FROM Meta_Metric_Ranges
135 INNER JOIN D_Quality
136 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
language_name = @Project_Language
```

```
137 and metric_name='ACC' and ? <= max AND ? >= min and configuration_name
      like '%Open JDK8 Metrics%');

138
139
140
141
142 SET @qualityWorstACC = (SELECT idQuality FROM Meta_Metric_Ranges
143 INNER JOIN D_Quality
144 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
      language_name = @Project_Language
145 and metric_name='ACC' and ? <= max AND ? >= min and configuration_name
      like '%Tomcat Metrics%');

146
147
148
149 INSERT INTO 'F_Project_Metric'
150 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
151 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
      ')
152 VALUES
153 (?, @idProject, @idACC, @qualityBestACC, @Best_Configuration, @idRelease
      , @idTime);

154
155
156
157
158
159 INSERT INTO 'F_Project_Metric'
160 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
161 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
      ')
162 VALUES
163 (?, @idProject, @idACC, @qualityWorstACC, @Worst_Configuration,
      @idRelease, @idTime);

164
165
166
167
168 SET @idANPM = (SELECT idMetric FROM D_Metric where metric_abbreviation='
      ANPM');

169
170
171
172 SET @qualityBestANPM = (SELECT idQuality FROM Meta_Metric_Ranges
173 INNER JOIN D_Quality
```

```
174 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
      language_name = @Project_Language
175 and metric_name='ANPM' and ? <= max AND ? >= min and configuration_name
      like '%Open JDK8 Metrics%');
176
177
178
179
180 SET @qualityWorstANPM = (SELECT idQuality FROM Meta_Metric_Ranges
181 INNER JOIN D_Quality
182 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
      language_name = @Project_Language
183 and metric_name='ANPM' and ? <= max AND ? >= min and configuration_name
      like '%Tomcat Metrics%');
184
185
186
187 INSERT INTO 'F_Project_Metric'
188 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
189 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
      ')
190 VALUES
191 (?, @idProject, @idANPM, @qualityBestANPM, @Best_Configuration,
      @idRelease, @idTime);
192
193
194
195
196
197 INSERT INTO 'F_Project_Metric'
198 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
199 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
      ')
200 VALUES
201 (?, @idProject, @idANPM, @qualityWorstANPM, @Worst_Configuration,
      @idRelease, @idTime);
202
203
204
205 SET @idCBO = (SELECT idMetric FROM D_Metric where metric_abbreviation='
      CBO');
206
207
208
209 SET @qualityBestCBO = (SELECT idQuality FROM Meta_Metric_Ranges
```

```
210 INNER JOIN D_Quality
211 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
      language_name = @Project_Language
212 and metric_name='CBO' and round(?) <= max AND round(?) >= min and
      configuration_name like '%Open JDK8 Metrics%');
213
214
215
216
217 SET @qualityWorstCBO = (SELECT idQuality FROM Meta_Metric_Ranges
218 INNER JOIN D_Quality
219 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
      language_name = @Project_Language
220 and metric_name='CBO' and round(?) <= max AND round(?) >= min and
      configuration_name like '%Tomcat Metrics%');
221
222
223
224 INSERT INTO 'F_Project_Metric'
225 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', '
      D_Quality_idQuality',
226 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
      ')
227 VALUES
228 (round(?), @idProject, @idCBO, @qualityBestCBO, @Best_Configuration,
      @idRelease, @idTime);
229
230
231
232
233
234 INSERT INTO 'F_Project_Metric'
235 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', '
      D_Quality_idQuality',
236 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
      ')
237 VALUES
238 (round(?), @idProject, @idCBO, @qualityWorstCBO, @Worst_Configuration,
      @idRelease, @idTime);
239
240
241
242 SET @idDIT = (SELECT idMetric FROM D_Metric where metric_abbreviation='
      DIT');
243
244
245
```

```
246 SET @qualityBestDIT = (SELECT idQuality FROM Meta_Metric_Ranges
247 INNER JOIN D_Quality
248 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index WHERE
language_name = @Project_Language
249 AND metric_name='DIT' AND ? <= max AND ? >= min AND configuration_name
like '%Open JDK8 Metrics%');

250
251
252
253
254 SET @qualityWorstDIT = (SELECT idQuality FROM Meta_Metric_Ranges
255 INNER JOIN D_Quality
256 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index WHERE
language_name = @Project_Language
257 AND metric_name='DIT' AND ? <= max AND ? >= min AND configuration_name
like '%Tomcat Metrics%');

258
259
260
261 INSERT INTO 'F_Project_Metric'
262 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
263 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime')
264 VALUES
265 (?, @idProject, @idDIT, @qualityBestDIT, @Best_Configuration, @idRelease,
, @idTime);

266
267
268
269
270
271 INSERT INTO 'F_Project_Metric'
272 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
273 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime')
274 VALUES
275 (?, @idProject, @idDIT, @qualityWorstDIT, @Worst_Configuration,
@idRelease, @idTime);

276
277
278
279
280
281
282 SET @idLCOM4 = (SELECT idMetric FROM D_Metric WHERE metric_abbreviation=
```

```
    'LCOM4') ;  
283  
284  
285  
286 SET @qualityBestLCOM4 = (SELECT idQuality FROM Meta_Metric_Ranges  
287 INNER JOIN D_Quality  
288 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index WHERE  
language_name = @Project_Language  
289 AND metric_name='LCOM4' AND ? <= max AND ? >= min AND configuration_name  
like '%Open JDK8 Metrics%');  
290  
291  
292  
293  
294 SET @qualityWorstLCOM4 = (SELECT idQuality FROM Meta_Metric_Ranges  
295 INNER JOIN D_Quality  
296 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index WHERE  
language_name = @Project_Language  
297 AND metric_name='LCOM4' AND ? <= max AND ? >= min AND configuration_name  
like '%Tomcat Metrics%');  
298  
299  
300  
301 INSERT INTO 'F_Project_Metric'  
302 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', '  
D_Quality_idQuality',  
303 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime  
' )  
304 VALUES  
305 (?, @idProject, @idLCOM4, @qualityBestLCOM4, @Best_Configuration,  
@idRelease, @idTime);  
306  
307  
308  
309  
310  
311 INSERT INTO 'F_Project_Metric'  
312 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', '  
D_Quality_idQuality',  
313 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime  
' )  
314 VALUES  
315 (?, @idProject, @idLCOM4, @qualityWorstLCOM4, @Worst_Configuration,  
@idRelease, @idTime);  
316  
317  
318
```

```
319
320
321
322
323
324 SET @idNOC = (SELECT idMetric FROM D_Metric WHERE metric_abbreviation='
325   NOC');
326
327
328 SET @qualityBestNOC = (SELECT idQuality FROM Meta_Metric_Ranges
329 INNER JOIN D_Quality
330 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index WHERE
331   language_name = @Project_Language
332   and metric_name='NOC' and ? <= max AND ? >= min and configuration_name
333     like '%Open JDK8 Metrics%');
334
335
336 SET @qualityWorstNOC = (SELECT idQuality FROM Meta_Metric_Ranges
337 INNER JOIN D_Quality
338 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index WHERE
339   language_name = @Project_Language
340   and metric_name='NOC' and ? <= max AND ? >= min and configuration_name
341     like '%Tomcat Metrics%');
342
343 INSERT INTO 'F_Project_Metric'
344 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_
345   _Quality_idQuality',
346   'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
347   ')
348
349
350
351
352
353 INSERT INTO 'F_Project_Metric'
354 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_
355   _Quality_idQuality',
356   'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
357   ')
```

```
356 VALUES
357 (?, @idProject, @idNOC, @qualityWorstNOC, @Worst_Configuration,
      @idRelease, @idTime);
358
359
360
361
362 SET @idNOM = (SELECT idMetric FROM D_Metric where metric_abbreviation='
      NOM');
363
364
365
366 SET @qualityBestNOM = (SELECT idQuality FROM Meta_Metric_Ranges
367 INNER JOIN D_Quality
368 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
      language_name = @Project_Language
369 and metric_name='NOM' and ? <= max AND ? >= min and configuration_name
      like '%Open JDK8 Metrics%');
370
371
372
373
374 SET @qualityWorstNOM = (SELECT idQuality FROM Meta_Metric_Ranges
375 INNER JOIN D_Quality
376 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
      language_name = @Project_Language
377 and metric_name='NOM' and ? <= max AND ? >= min and configuration_name
      like '%Tomcat Metrics%');
378
379
380
381 INSERT INTO 'F_Project_Metric'
382 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', '
      D_Quality_idQuality',
383 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
      ')
384 VALUES
385 (?, @idProject, @idNOM, @qualityBestNOM, @Best_Configuration, @idRelease
      , @idTime);
386
387
388
389
390
391 INSERT INTO 'F_Project_Metric'
392 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', '
      D_Quality_idQuality',
```

```
393 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
      ')
394 VALUES
395 (?, @idProject, @idNOM, @qualityWorstNOM, @Worst_Configuration,
      @idRelease, @idTime);
396
397
398
399
400 SET @idNPA = (SELECT idMetric FROM D_Metric where metric_abbreviation='
      NPA');
401
402
403
404 SET @qualityBestNPA = (SELECT idQuality FROM Meta_Metric_Ranges
405 INNER JOIN D_Quality
406 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
      language_name = @Project_Language
407 and metric_name='NPA' and ? <= max AND ? >= min and configuration_name
      like '%Open JDK8 Metrics%');
408
409
410
411
412 SET @qualityWorstNPA = (SELECT idQuality FROM Meta_Metric_Ranges
413 INNER JOIN D_Quality
414 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
      language_name = @Project_Language
415 and metric_name='NPA' and ? <= max AND ? >= min and configuration_name
      like '%Tomcat Metrics%');
416
417
418
419 INSERT INTO 'F_Project_Metric'
420 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric',
      'D_Quality_idQuality',
421 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime
      ')
422 VALUES
423 (?, @idProject, @idNPA, @qualityBestNPA, @Best_Configuration, @idRelease
      , @idTime);
424
425
426
427
428
429 INSERT INTO 'F_Project_Metric'
```

```
430 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
431 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime')
432 VALUES
433 (?, @idProject, @idNPA, @qualityWorstNPA, @Worst_Configuration,
434     @idRelease, @idTime);
435
436
437 SET @idRFC = (SELECT idMetric FROM D_Metric where metric_abbreviation='
438     RFC');
439
440
441 SET @qualityBestRFC = (SELECT idQuality FROM Meta_Metric_Ranges
442 INNER JOIN D_Quality
443 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
444     language_name = @Project_Language
445 and metric_name='RFC' and round(?) <= max AND round(?) >= min and
446     configuration_name like '%Open JDK8 Metrics%');
447
448
449 SET @qualityWorstRFC = (SELECT idQuality FROM Meta_Metric_Ranges
450 INNER JOIN D_Quality
451 ON Meta_Metric_Ranges.quality_index=D_Quality.quality_index where
452     language_name = @Project_Language
453 and metric_name='RFC' and round(?) <= max AND round(?) >= min and
454     configuration_name like '%Tomcat Metrics%');
455
456 INSERT INTO 'F_Project_Metric'
457 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
458 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime')
459 VALUES
460 (round(?), @idProject, @idRFC, @qualityBestRFC, @Best_Configuration,
461     @idRelease, @idTime);
462
463
464
465 INSERT INTO 'F_Project_Metric'
```

```

466 ('percentil_value', 'D_Project_idProject', 'D_Metric_idMetric', 'D_Quality_idQuality',
467 'D_Configuration_idConfiguration', 'D_Release_idRelease', 'D_Time_idTime')
468 VALUES
469 (round(?), @idProject, @idRFC, @qualityWorstRFC, @Worst_Configuration,
    @idRelease, @idTime);

```

Código-Fonte 1 – Script SQL de Avaliação dos Valores Percentis das Métricas de Código-Fonte

Na terceira transformação, como se mostra na Figura 33, foram cobertos os processos de negócio de avaliar os cenários de limpeza de código-fonte em cada classe do Projeto em uma determinada *release* e o cálculo da Taxa de Aproveitamento de Oportunidades de Melhoria de Código-Fonte. Para tal, foram obtidos os valores cada métrica para cada classe utilizando o componente *JSON Input*. Após a coleta das métricas, foram inseridas, utilizando o componente *Combination Lookup* a fim de evitar duplicações ao longo das *releases*, as classes do projeto.

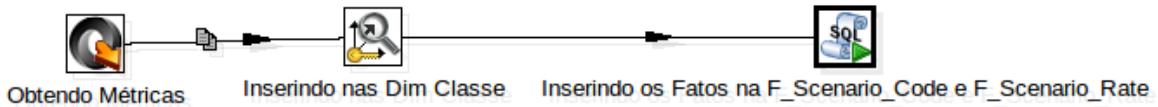


Figura 33 – Terceira Transformação realizada no Kettle

Por fim, foram identificados os Cenários de Limpeza de Código-Fonte com auxílio dos metadados utilizando o componente *Execute SQL Script*. Ainda neste componente, foi calculada a Taxa de Aproveitamento de Oportunidades de Melhoria de Código-Fonte a partir a soma de todos cenários de limpeza identificados em uma determinada *release* e a soma de todas as classes. O código-fonte, que foi colocado no componente *Execute SQL Script* é descrito no Código-Fonte 2, onde cada ? foi substituído por uma variável dentro da *Transformation*.

```

1
2 SET @release_number = (SELECT substring_index('?', '.', 1));
3
4 SET @idRelease = (SELECT max(idRelease) from D_Release);
5
6 SET @Project_Language = (SELECT project_language from D_Project where
    idProject = @idProject);
7
8 SET @idProject = (SELECT max(idProject) from D_Project);
9
10

```

```
11 SET @idACCM = (SELECT idMetricRange FROM Meta_Metric_Ranges
12 where language_name = @Project_Language
13 and metric_name='ACCM' and ? <= max AND ? >= min and configuration_name
14 like '%Open JDK8 Metrics%');
15 SET @idAMLOC = (SELECT idMetricRange FROM Meta_Metric_Ranges
16 where language_name = @Project_Language
17 and metric_name='AMLOC' and ? <= max AND ? >= min and configuration_name
18 like '%Open JDK8 Metrics%');
19 SET @idANPM = (SELECT idMetricRange FROM Meta_Metric_Ranges
20 where language_name = @Project_Language
21 and metric_name='ANPM' and ? <= max AND ? >= min and configuration_name
22 like '%Open JDK8 Metrics%');
23 SET @idCBO = (SELECT idMetricRange FROM Meta_Metric_Ranges
24 where language_name = @Project_Language
25 and metric_name='CBO' and ? <= max AND ? >= min and configuration_name
26 like '%Open JDK8 Metrics%');
27 SET @idLCOM4 = (SELECT idMetricRange FROM Meta_Metric_Ranges
28 where language_name = @Project_Language
29 and metric_name='LCOM4' and ? <= max AND ? >= min and configuration_name
30 like '%Open JDK8 Metrics%');
31 SET @idNPA = (SELECT idMetricRange FROM Meta_Metric_Ranges
32 where language_name = @Project_Language
33 and metric_name='NPA' and ? <= max AND ? >= min and configuration_name
34 like '%Open JDK8 Metrics%');
35 SET @idNOC = (SELECT idMetricRange FROM Meta_Metric_Ranges
36 where language_name = @Project_Language
37 and metric_name='NOC' and ? <= max AND ? >= min and configuration_name
38 like '%Open JDK8 Metrics%');
39
40 SET @idRFC = (SELECT idMetricRange FROM Meta_Metric_Ranges
41 where language_name = @Project_Language
42 and metric_name='RFC' and ? <= max AND ? >= min and configuration_name
43 like '%Open JDK8 Metrics%');
44
45
46
47 SET @idClassePoucoCoesa = (SELECT 'Meta_Scenario'.'idMeta_Scenario'
48 FROM 'Meta_Metric_Ranges_Meta_Scenario' INNER JOIN Meta_Scenario ON
Meta_Scenario.idMeta_Scenario = Meta_Metric_Ranges_Meta_Scenario.
```

```
    Meta_Scenario_idMeta_Scenario where
49 Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange1 =
    @idLCOM4
50 and Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange2 =
    @idRFC);
51
52
53
54
55 SET @idInterfaceMetodos = (SELECT 'Meta_Scenario'.'idMeta_Scenario'
56 FROM 'Meta_Metric_Ranges_Meta_Scenario' INNER JOIN Meta_Scenario ON
    Meta_Scenario.idMeta_Scenario = Meta_Metric_Ranges_Meta_Scenario.
    Meta_Scenario_idMeta_Scenario where
57 Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange1 =
    @idANPM);
58
59
60
61
62 SET @idClasseFilhos = (SELECT 'Meta_Scenario'.'idMeta_Scenario'
63 FROM 'Meta_Metric_Ranges_Meta_Scenario' INNER JOIN Meta_Scenario ON
    Meta_Scenario.idMeta_Scenario = Meta_Metric_Ranges_Meta_Scenario.
    Meta_Scenario_idMeta_Scenario where
64 Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange1 =
    @idNOC);
65
66
67
68 SET @idClasseGrande = (SELECT 'Meta_Scenario'.'idMeta_Scenario'
69 FROM 'Meta_Metric_Ranges_Meta_Scenario' INNER JOIN Meta_Scenario ON
    Meta_Scenario.idMeta_Scenario = Meta_Metric_Ranges_Meta_Scenario.
    Meta_Scenario_idMeta_Scenario where
70 Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange1 =
    @idACCM
71 and Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange2 =
    @idAMLOC);
72
73
74
75 SET @idClasseExposta = (SELECT 'Meta_Scenario'.'idMeta_Scenario'
76 FROM 'Meta_Metric_Ranges_Meta_Scenario' INNER JOIN Meta_Scenario ON
    Meta_Scenario.idMeta_Scenario = Meta_Metric_Ranges_Meta_Scenario.
    Meta_Scenario_idMeta_Scenario where
77 Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange1 =
    @idNPA);
78
79
```

```
80
81 SET @idComplexidadeEstrutural = (SELECT 'Meta_Scenario'.'idMeta_Scenario
82 '
83 FROM 'Meta_Metric_Ranges_Meta_Scenario' INNER JOIN Meta_Scenario ON
84     Meta_Scenario.idMeta_Scenario = Meta_Metric_Ranges_Meta_Scenario.
85     Meta_Scenario_idMeta_Scenario where
86     Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange1 =
87     @idCBO
88     and Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange2
89     = @idLCOM4);
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
```

80

```
81 SET @idComplexidadeEstrutural = (SELECT 'Meta_Scenario'.'idMeta_Scenario
82 '
83 FROM 'Meta_Metric_Ranges_Meta_Scenario' INNER JOIN Meta_Scenario ON
84     Meta_Scenario.idMeta_Scenario = Meta_Metric_Ranges_Meta_Scenario.
85     Meta_Scenario_idMeta_Scenario where
86     Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange1 =
87     @idCBO
88     and Meta_Metric_Ranges_Meta_Scenario.Meta_Metric_Ranges_idMetricRange2
89     = @idLCOM4);
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
```

```
122 ('quantity_Scenario','D_Scenario_Clean_Code_idScenario',
123 'D_Project_idProject',
124 'D_Release_idRelease',
125 'D_Class_idClass')
126 VALUES
127 (1, @idClasseFilhos, @idProject, @idRelease, ?);
128
129
130
131 INSERT INTO 'source_info'.'Temporary_F_Scenario_Class'
132 ('quantity_Scenario','D_Scenario_Clean_Code_idScenario',
133 'D_Project_idProject',
134 'D_Release_idRelease',
135 'D_Class_idClass')
136 VALUES
137 (1, @idClasseGrande, @idProject, @idRelease, ?);
138
139
140
141 INSERT INTO 'source_info'.'Temporary_F_Scenario_Class'
142 ('quantity_Scenario','D_Scenario_Clean_Code_idScenario',
143 'D_Project_idProject',
144 'D_Release_idRelease',
145 'D_Class_idClass')
146 VALUES
147 (1, @idClasseExposta, @idProject, @idRelease, ?);
148
149
150 INSERT INTO 'source_info'.'Temporary_F_Scenario_Class'
151 ('quantity_Scenario','D_Scenario_Clean_Code_idScenario',
152 'D_Project_idProject',
153 'D_Release_idRelease',
154 'D_Class_idClass')
155 VALUES
156 (1, @idComplexidadeEstrutural, @idProject, @idRelease, ?);
157
158
159
160 INSERT INTO F_Scenario_Class ('F_Scenario_Class'.'quantity_Scenario',
161 'F_Scenario_Class'.'D_Scenario_Clean_Code_idScenario',
162 'F_Scenario_Class'.'D_Project_idProject',
163 'F_Scenario_Class'.'D_Release_idRelease',
164 'F_Scenario_Class'.'D_Class_idClass') SELECT 'Temporary_F_Scenario_Class'
165   .'quantity_Scenario','Temporary_F_Scenario_Class'.'D_Scenario_Clean_Code_idScenario',
166   'Temporary_F_Scenario_Class'.'D_Project_idProject',
167   'Temporary_F_Scenario_Class'.'D_Release_idRelease',
```

```

166 'Temporary_F_Scenario_Class'.'D_Class_idClass' FROM
    Temporary_F_Scenario_Class WHERE Temporary_F_Scenario_Class.
    D_Scenario_Clean_Code_idScenario IS NOT NULL;
167
168 SET @total_scenarios = (SELECT COUNT(*) FROM source_info.F_Scenario_Class
    WHERE D_Release_idRelease= @idRelease);
169
170 SET @rate = (@total_scenarios/?);
171
172 DROP TABLE IF EXISTS 'Temporary_F_Rate_Scenario';
173 CREATE TEMPORARY TABLE 'Temporary_F_Rate_Scenario' (
174     'idRateScenario' int(11) NOT NULL AUTO_INCREMENT,
175     'RateScenario' double DEFAULT NULL,
176     'Quantiy_Scenarios' double DEFAULT NULL,
177     'numberOfClasses' int(11) DEFAULT NULL,
178     'D_Project_idProject' int(11) NOT NULL,
179     'D_Release_idRelease' int(11) NOT NULL,
180     PRIMARY KEY ('idRateScenario'))
181 ENGINE=InnoDB DEFAULT CHARSET=utf8;
182
183 INSERT INTO 'source_info'.'Temporary_F_Rate_Scenario'
184 ('RateScenario', 'Quantiy_Scenarios', 'numberOfClasses',
185 'D_Project_idProject', 'D_Release_idRelease')
186 VALUES
187 (@rate, @total_scenarios, ?, @idProject, @idRelease);
188
189 SET @quantiy_Violation = (SELECT COUNT(*) FROM source_info.
    F_Project_Violation WHERE D_Release_idRelease= @idRelease);
190 SET @quantiy_Bug = (SELECT COUNT(*) FROM source_info.F_Project_Bug WHERE
    D_Release_idRelease= @idRelease);
191
192
193 REPLACE INTO 'source_info'.'F_Rate_Scenario'
194 SET 'RateScenario' = @rate, 'numberOfClasses' = ?, 'Quantiy_Scenarios' =
    @total_scenarios,
195 'D_Release_idRelease' = @idRelease, 'D_Project_idProject' = @idProject, 'Quantiy_Violations' =
    @quantiy_Violation, 'Quantiy_Bugs' =
    @quantiy_Bug ;

```

Código-Fonte 2 – Script SQL de Identificação de Cenários de limpeza de Código-Fonte e Cálculo da Taxa de Aproveitamento de Oportunidades de Melhoria de Código-Fonte

Na quarta transformação, como se mostra na Figura 34, foram cobertos os processos de negócio para avaliar as violações do código-fonte em cada classe do Projeto em uma determinada *release*. Para tal, foram obtidos os valores de cada violação para cada classe utilizando o componente *JSON Input*. Após a coleta das violações, foram inseridas,

utilizando o componente *Combination Lookup* a fim de evitar duplicações ao longo das *releases*, as classes do projeto.



Figura 34 – Quarta Transformação realizada no Kettle

Em seguida, foram identificados as violações do Código-Fonte e suas características utilizando o componente *Execute SQL Script*. O código-fonte, que foi colocado no componente *Execute SQL Script* é descrito no Código-Fonte 3, onde cada ? foi substituído por uma variável dentro da *Transformation*.

```

CyclomaticComplexity', 'DataflowAnomalyAnalysis', 'DefaultPackage',
, 'EmptyCatchBlock', 'EmptyIfStmt', 'ExcessiveClassLength',
ExcessiveMethodLength', 'ExcessiveParameterList',
ExcessivePublicCount', 'FieldDeclarationsShouldBeAtStartOfClass',
'FinalFieldCouldBeStatic', 'GodClass', 'IfStmtsMustUseBraces',
ImmutableField', 'InefficientStringBuffering',
InsufficientStringBufferDeclaration', 'IntegerInstantiation',
JUnit4TestShouldUseTestAnnotation', 'LocalVariableCouldBeFinal',
LongInstantiation', 'LongVariable', 'MethodArgumentCouldBeFinal',
MethodNamingConventions', 'ModifiedCyclomaticComplexity',
NcssMethodCount', 'NPathComplexity', 'NullAssignment',
OneDeclarationPerLine', 'OnlyOneReturn', 'AvoidRethrowingException
', 'CallSuperInConstructor', 'ForLoopsMustUseBraces',
SimplifyBooleanExpressions', 'UseConcurrentHashMap',
SimplifyBooleanExpressions') THEN '3'
18 WHEN '?' IN ('PositionLiteralsFirstInComparisons',
PrematureDeclaration', 'PreserveStackTrace',
RedundantFieldInitializer', 'ShortClassName', 'ShortInstantiation',
, 'ShortVariable') THEN '4'
19 WHEN '?' IN ('SignatureDeclareThrowsException',
SimpleDateFormatNeedsLocale', 'SimplifyBooleanReturns',
SingularField', 'StdCyclomaticComplexity', 'TooManyFields',
TooManyMethods', 'UncommentedEmptyConstructor',
UncommentedEmptyMethod', 'UnnecessaryConstructor',
UnnecessaryFullyQualifiedNames', 'UnnecessaryLocalBeforeReturn',
UnnecessaryParentheses', 'UnusedFormalParameter', 'UnusedImports',
UnusedLocalVariable', 'UnusedModifier', 'UnusedPrivateField',
UnusedPrivateMethod', 'UseCollectionIsEmpty', 'UselessParentheses',
, 'UselessStringValueOf', 'UseLocaleWithCaseConversions',
UseObjectForClearerAPI', 'UseStringBufferForStringAppends',
UseVarargs', 'VariableNamingConventions', 'EmptyStatementNotInLoop
, 'LogicInversion') THEN '5'
20 ELSE '0'
21 END);
22
23
24 INSERT INTO 'F_Project_Violation'
25 ('quantiy_Violation', 'D_Class_Violation_idClass_violation',
'D_Class_Violation_idRules', 'D_Class_Violation_idSeverite',
26 'D_Project_idProject', 'D_Release_idRelease', 'D_Time_idTime')
27 VALUES
28 (1, ?, @idRules, @idSeverite, @idProject, @idRelease, @idTime);

```

Código-Fonte 3 – Script SQL de Identificação das violações de Código-Fonte

Na quinta transformação, como se mostra na Figura 35, foram cobertos os processos de negócio para avaliar *bugs* do código-fonte em cada classe do Projeto em uma

determinada *release*. Para tal, foram obtidos os valores de cada *bug* para cada classe utilizando o componente *Text file Input*. Após a coleta dos *bugs* foi utilizado o componente *Filter rows* para desprezar as linhas em brancos do arquivo CSV, o componente *If field value is null* para eliminar os os valores nulos dos campos *start* e *end* e o componete *Concat Fields* para juntar os campos *start* e *end* em um único campo chamado *line*. Em seguida, foram inseridas, utilizando o componente *Combination Lookup* a fim de evitar duplicações ao longo das *releases*, as classes do projeto.

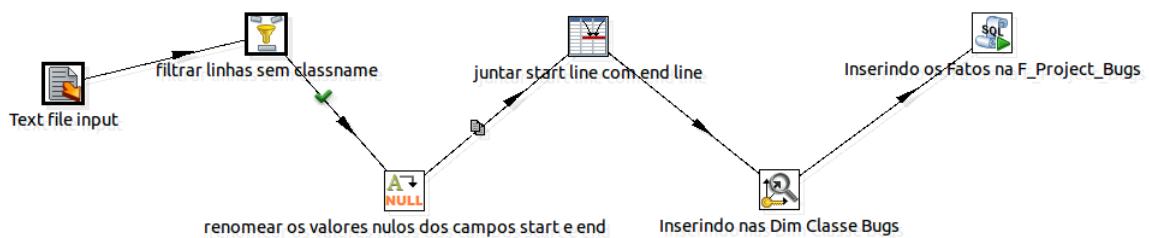


Figura 35 – Quinta Transformação realizada no Kettle

Por fim, foram identificados *bugs* do Código-Fonte e suas características utilizando o componente *Execute SQL Script*.O código-fonte, que foi colocado no componente *Execute SQL Script* é descrito no Código-Fonte 4, onde cada ? foi substituído por uma variável dentro da *Transformation*.

```

1
2 SET @release_number = (SELECT substring_index('?', '.', 1));
3
4 SET @idProject = (SELECT max(idProject) from D_Project);
5
6 SET @idTime = (SELECT max(idTime) from D_Time);
7
8 SET @idRelease = (SELECT max(idRelease) from D_Release);
9
10 SET @Project_Language = (SELECT project_language from D_Project where
     idProject = @idProject);
11
12 SET @idType = (SELECT idType from D_Type where type_name = '?');
13
14
15
16 SET @idPriority = ( CASE
17     WHEN '?' IN ('RpC_REPEAT_CONDITIONAL_TEST', ' / '
18         ES_COMPARING_PARAMETER_STRING_WITH_EQ', ' /
19         ES_COMPARING_PARAMETER_STRING_WITH_EQ', ' / '
20         RC_REF_COMPARISON', 'RC_REF_COMPARISON', 'DM_NUMBERCTOR', ' /
21         NP_NULL_ON_SOME_PATH', 'RCN_REDUNDANT_NULLCHECK_OF_NONNULL_VALUE
22         ', 'RCN_REDUNDANT_NULLCHECK_OF_NULL_VALUE') THEN '1'

```

```

18 WHEN   '?' IN ('BX_UNBOXING_IMMEDIATELY_REBOXED', ,
                  WMI_WRONG_MAP_ITERATOR', ,
                  RCN_REDUNDANT_NULLCHECK_WOULD_HAVE_BEEN_A_NPE', ,           /
                  EC_UNRELATED_TYPES', ,           /SE_BAD_FIELD', ,
                  DB_DUPLICATE_BRANCHES', , 'DLS_DEAD_LOCAL_STORE', , 'DM_CONVERT_CASE
                  ', , 'EC_UNRELATED_TYPES', , 'EI_EXPOSE_REP', , 'EI_EXPOSE_REP2', ,
                  NM_FIELD_NAMING_CONVENTION', , 'NP_LOAD_OF_KNOWN_NULL_VALUE', ,
                  RV_RETURN_VALUE_IGNORED_NO_SIDE_EFFECT', , 'SE_BAD_FIELD', ,
                  SS_SHOULD_BE_STATIC', , 'UCF_USELESS_CONTROL_FLOW') THEN '2'
19 WHEN   '??' IN ('ES_COMPARING_STRINGS_WITH_EQ', ,           /
                  ES_COMPARING_STRINGS_WITH_EQ', , 'UPM_UNCALLED_PRIVATE_METHOD', ,
                  DE_MIGHT_IGNORE', , 'SIC_INNER_SHOULD_BE_STATIC_ANON', ,
                  HE_EQUALS_USE_HASHCODE', , 'ME_ENUM_FIELD_SETTER', ,
                  REC_CATCH_EXCEPTION') THEN '3'
20 ELSE '0'
21 END);
22
23 INSERT INTO 'F_Project_Bug'
24 ('quantiy_Bug', 'D_Class_Bug_idClass_Bug', 'D_Class_Bug_idType', ,
     D_Class_Bug_idPriority',
25 'D_Project_idProject', 'D_Release_idRelease', 'D_Time_idTime')
26 VALUES
27 (1, ?, @idType, @idPriority, @idProject, @idRelease, @idTime);

```

Código-Fonte 4 – Script SQL de Identificação de bugs no Código-Fonte

A.2 Implementação do Job

Como explicado anteriormente, o Kettle utiliza o *Job* para executar tarefas, em nível mais alto, de fluxo de controle, tais como, mandar um email em caso de falha, baixar um arquivo, executar transformações e entre outras atividades. Dessa forma os principais componentes internos do *Job*, que foram utilizados no trabalho, são mostrados na Figura 36.



Figura 36 – Componentes do Kettle que foram utilizadas nos *Jobs*

O componente *Start* é utilizado para marcar o início da execução de um determinado *Job*. Nele, é possível programar a execução repetida de um determinado *job*, como por exemplo, a cada hora, dia ou mês; O componente *Transformation* serve para executar uma determinada Transformação, que fora especificada anteriormente. Já o componente

set variables serve para setar valores para variáveis que as transformações precisam para ser iniciada, neste trabalho foi criada uma variável que recebe o caminho dos arquivos das análises das ferramentas Analizo, FindBugs e PMD.

Após se construir o arquivo *Job* do presente trabalho, obteve-se a Figura 37.



Figura 37 – *Job* deste Trabalho

No *Job*, a transformação "Dados do Projeto" corresponde a execução da Figura 31. Já a transofrmação "Análise dos Valores Percentis" corresponde a Figura 32, a transofrmação "Análise dos Cenários de Limpeza de Código-Fonte"corresponde a Figura 33, a transofrmação "Análise das Violações PMD" corresponde a Figura 34 e por fim, a transofrmacão "Análise dos Bugs do FindBugs" corresponde a Figura 35.

APÊNDICE B – Implementação do Dashboard

B.1 Dashboard Geral

Neste apêndice, será apresentado a implementação dos *Dashboards* no CDE. O primeiro *dashboard* a ser apresentado será o *dashboard* Geral demonstrado na Figura 38.

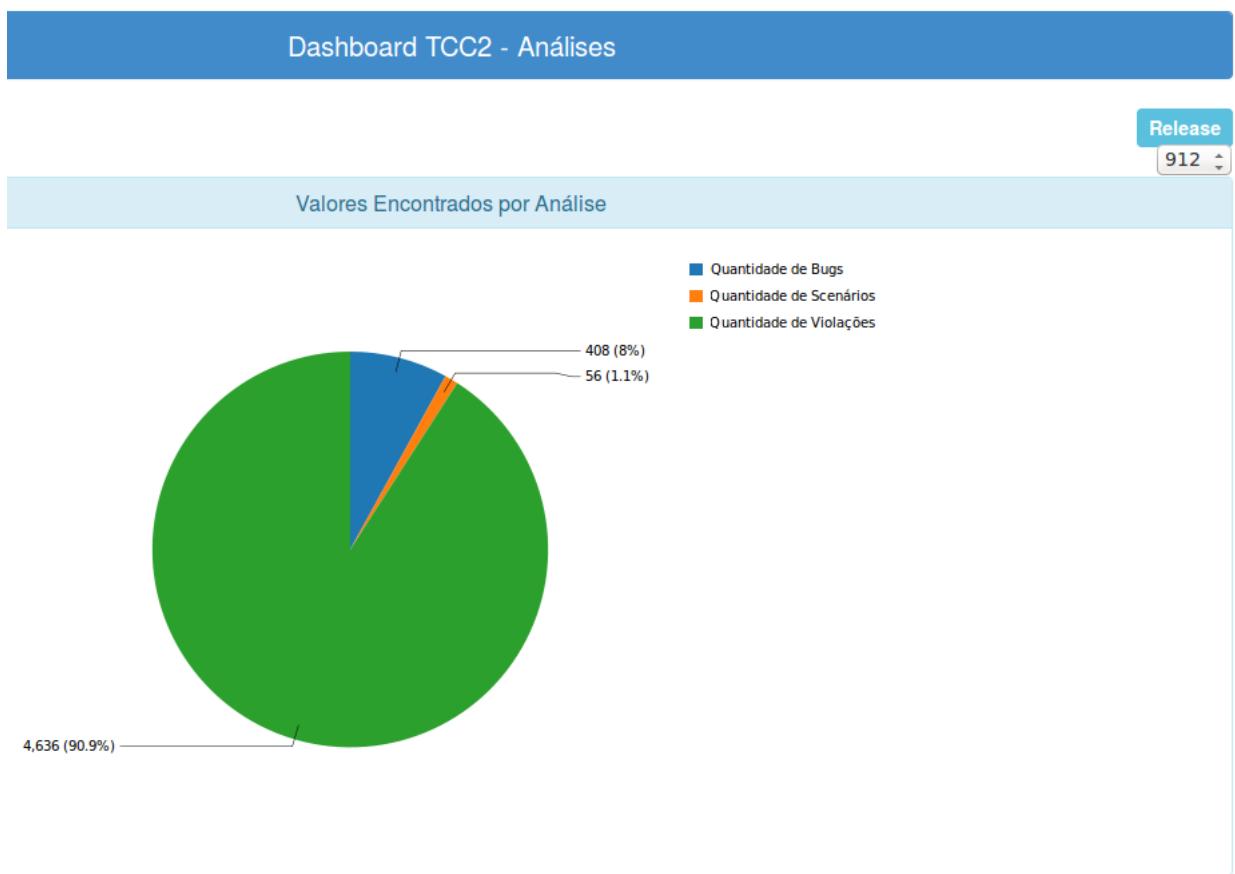


Figura 38 – *Dashboard* Geral

Como explicado anteriormente, o CDE oferece três perspectivas: *Layout*, *Components*, *DataSources*. A estrutura *Layout* do dashboard Geral foi dividido em um *Resource*, três *Row* e quatro *Column*. Dentro do *Resource* foi escrito o código css abaixo:

```
body{
    margin-top: 0.5cm;
}
paneltitle{
```

```

    vertical-align: top;
}
.dashboardHeading{
    height:10px;
}

.releaseTextCss{
    /*top:-2px;*/
    text-align:right;
}
.releaseSelectorCss{
    /*top:-2px;*/
    text-align:right;
}
.panel1Css{
    /*padding-right:30px;*/
    text-align: center;
}
.panel2Css{
    padding-left:0px;
    text-align: center;
}
.panel3Css{
    text-align: center;
    overflow:auto;
}

```

A primeira *Row* que define o nome do *dashboard* na tela, possui um *Column* com o código HTML abaixo:

```
<div class="panel panel-primary">
    <div class="panel-heading">Dashboard TCC2 - Análises</div>
</div>
```

A segunda *Row*, possui duas *Column* que definem o nome da *release* com o código HTML abaixo e o *selector* da *release* que o utiliza a declaração *releaseSelectorCss* do CSS.

```
<span class="label label-info" align="right">Release</span>
```

A última *Row*, possui uma *Column* que define o painel do gráfico *pieChart* e possui o código HTML abaixo:

```
<div class="panel panel-info">
  <div class="panel-heading">
    <h3 class="panel-title">Valores Encontrados por Análise</h3>
  </div>
  <div class="panel-body" id="Panel3">
    Panel content
  </div>
</div>
```

A estrutura *Components* foi dividido em três *Groups*, *Charts*, *Generic* e *Selects*. O *Group Charts* é responsável pelo componente *CCC pie Chart* que possui o código javascript abaixo e que possibilita o *drill down* para os outros *dashboards*.

```
function sendParameter(scene){
  var url = null;
  var urlPmd='http://localhost:8080/pentaho/api/repos/
:public:dashboard:pmd:dashboard%20pmd%20v3.wcdf/generatedContent';
  var urlBug='http://localhost:8080/pentaho/api/repos/
:public:dashboard:findbugs:dashboard%20findbugs%20v3.wcdf
/generatedContent';
  var urlScenarios='http://localhost:8080/pentaho/api/repos/
:public:dashboard:scenarios:
dashboard%20scenarios%20v3.wcdf/generatedContent';

  var vars = scene.vars;
  var c = vars.category.value;
  var v = vars.value.value;

  if (c == "Quantidade de Violações") {
    url = urlPmd;
    c = "PMD";
  }else if (c == "Quantidade de Bugs") {
    url = urlBug;
    c = "FindBugs";
  } else {
    url = urlScenarios;
```

```

    c = "Scenários de Limpeza";
}

alert("análise:" + c + "\nquantidade: "+v);

window.location=url;
}

```

O *Group OLAP parameter* é responsável pela configuração do parâmetro utilizado para a construção do *selector* da *release* e possui o valor de propriedade abaixo:

```
[D Release.Release name].[All D Release.Release names]
```

O *Group Select Component* é responsável pelas configurações do *selector* da *release*.

A perspectiva de *Datasources* possui apenas o *Group MDX Queries* e possui duas consultas, a *mdx over mondrianjndi* que é responsável pela consulta do gráfico *CCC pie Chart* que é descrita abaixo:

```

with member [Measures].[Name] as '${release}.CurrentMember.UniqueName'
select TopCount( filter({Descendants
([D Release.Release name].[All D Release.Release names]
,[D Release.Release name]
.[Release name])}, not isempty(([D Release.Release name]
.CurrentMember)) ) , 50) on ROWS,
{[Measures].[Name]} on Columns
from [project violation]

```

E a segunda consulta também é do tipo *mdx over mondrianjn* e é responsável pela consulta do *selector* da *release* e está descrita abaixo:

```

WITH
MEMBER [Measures].[Quantidade de Scenários] AS [Measures].[Quantiy Scenarios]
MEMBER [Measures].[Quantidade de Violações] AS [Measures].[Quantiy Violations]
MEMBER [Measures].[Quantidade de Bugs] AS [Measures].[Quantiy Bugs]
SELECT
NON EMPTY {Hierarchize({[D Project.Project abbreviation].
[Project abbreviation].Members})} ON COLUMNS,
NON EMPTY {Hierarchize({{[Measures].
[Quantidade de Scenários], [Measures].[Quantidade de Violações],
```

```
[Measures]. [Quantidade de Bugs]}})} ON ROWS  
FROM [rate scenario]  
WHERE ${release}
```

B.2 Dashboards Cenários, FindBugs e PMD

Os *Dashboards* Cenários, FindBugs e PMD que são apresentados nas Figuras 39, 40 e 41 possuem a perspectiva de *layout* e *Components* praticamente iguais mudando apenas as consultas na perspectiva de *Datasources*.

A perspectiva de *layout* dos três *dashboards* foram divididos em um *Resource*, quatro *Row* e seis *Column*. Dentro do *Resource* foi escrito o código css abaixo:

```
body{  
    margin-top: 0.5cm;  
}  
  
.paneltitle{  
    vertical-align: top;  
}  
  
.dashboardHeading{  
    height:10px;  
}  
  
.releaseTextCss{  
    /*top:-2px;*/  
    text-align:right;  
}  
  
.releaseSelectorCss{  
    /*top:-2px;*/  
    text-align:right;  
}  
  
.panel1Css{  
    /*padding-right:30px;*/  
    text-align: center;  
}  
  
.panel2Css{  
    padding-left:0px;  
    text-align: center;  
}
```

```
.panel3Css{
    text-align: center;
    overflow:auto;
}
```

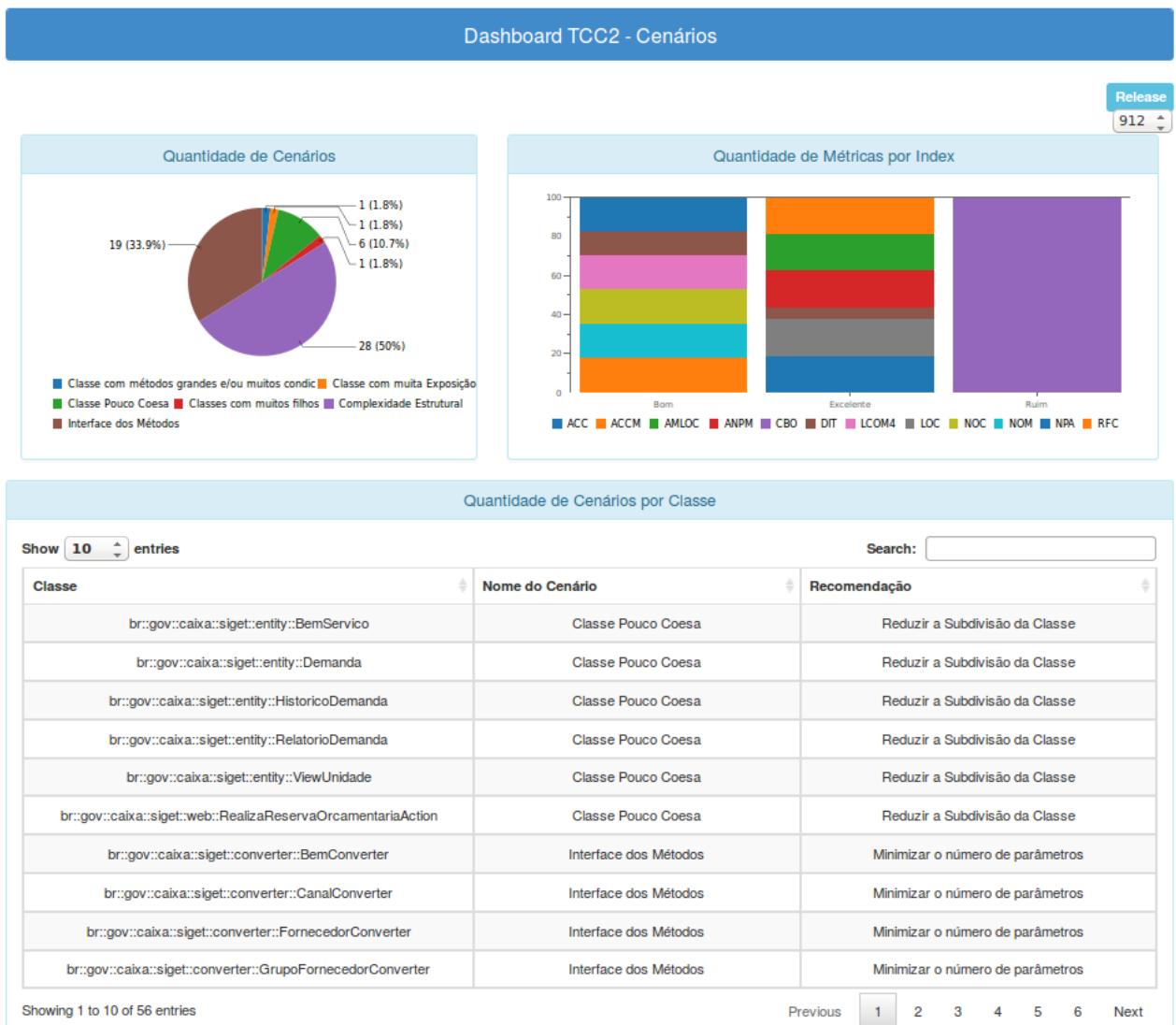


Figura 39 – Dashboard Cenários

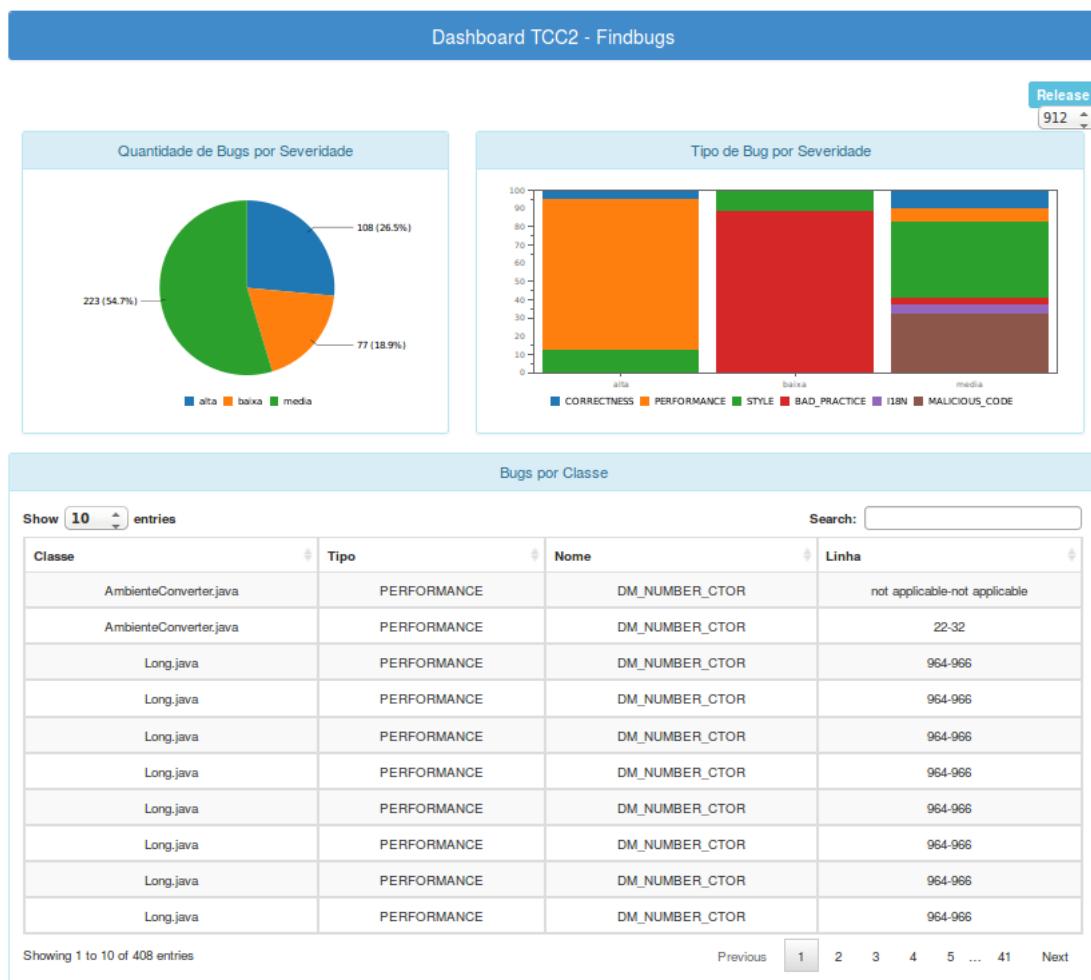


Figura 40 – Dashboard FindBugs

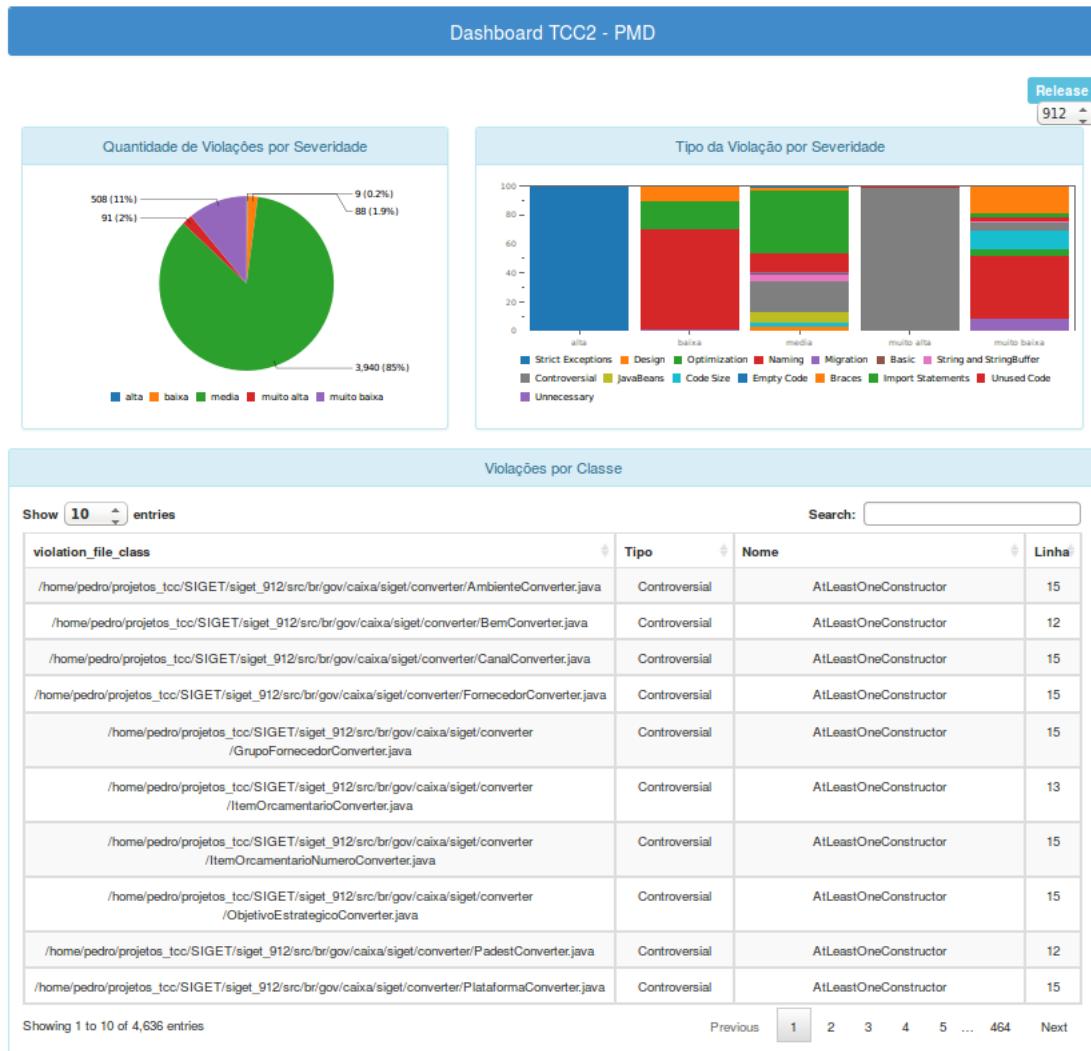


Figura 41 – Dashboard PMD

A primeira *Row* que define o nome do *dashboard* na tela, possui um *Column* com um código HTML.

Dashboard Cenários:

```
<div class="panel panel-primary">
  <div class="panel-heading">Dashboard TCC2 - Cenários</div>
</div>
```

Dashboard Findbugs:

```
<div class="panel panel-primary">
  <div class="panel-heading">Dashboard TCC2 - Findbugs</div>
</div>
```

Dashboard PMD:

```
<div class="panel panel-primary">
  <div class="panel-heading">Dashboard TCC2 - PMD</div>
</div>
```

A segunda *Row*, possui duas *Column* que definem o nome da *release* com o código HTML abaixo e o *selector* da *release* que o utiliza a declaração *releaseSelectorCss* do CSS.

```
<span class="label label-info" align="right">Release</span>
```

A terceira *Row*, possui duas *Column*, a primeira define o painel do gráfico *Pie Chart* e possui o código HTML abaixo:

Dashboard Cenários:

```
<div class="panel panel-info">
  <div class="panel-heading">
    <h3 class="panel-title">Quantidade de Cenários</h3>
  </div>
  <div class="panel-body" id="Panel1">
    Panel content
  </div>
</div>
```

Dashboard Findbugs:

```
<div class="panel panel-info">
  <div class="panel-heading">
    <h3 class="panel-title">Quantidade de Bugs por Severidade</h3>
  </div>
  <div class="panel-body" id="Panel1">
    Panel content
  </div>
</div>
```

Dashboard PMD:

```
<div class="panel panel-info">
  <div class="panel-heading">
    <h3 class="panel-title">Quantidade de Violações por Severidade</h3>
  </div>
  <div class="panel-body" id="Panel1">
```

```
    Panel content
  </div>
</div>
```

e a segunda que define o painel do gráfico *Stacked Bar Chart* e possui o código HTML abaixo:

Dashboard Cenários:

```
<div class="panel panel-info">
  <div class="panel-heading">
    <h3 class="panel-title">Quantidade de Métricas por Index</h3>
  </div>
  <div class="panel-body" id="Panel2">
    Panel content
  </div>
</div>
```

Dashboard Findbugs:

```
<div class="panel panel-info">
  <div class="panel-heading">
    <h3 class="panel-title">Tipo de Bug por Severidade</h3>
  </div>
  <div class="panel-body" id="Panel2">
    Panel content
  </div>
</div>
```

Dashboard PMD:

```
<div class="panel panel-info">
  <div class="panel-heading">
    <h3 class="panel-title">Tipo da Violação por Severidade</h3>
  </div>
  <div class="panel-body" id="Panel2">
    Panel content
  </div>
</div>
```

A última *Row* define o painel do componente *table Component* e possui o código HTML abaixo:

Dashboard Cenários:

```
<div class="panel panel-info">
  <div class="panel-heading">
    <h3 class="panel-title">Quantidade de Cenários por Classe</h3>
  </div>
  <div class="panel-body" id="Panel3">
    Panel content
  </div>
</div>
```

Dashboard Findbugs:

```
<div class="panel panel-info">
  <div class="panel-heading">
    <h3 class="panel-title">Bugs por Classe</h3>
  </div>
  <div class="panel-body" id="Panel3">
    Panel content
  </div>
</div>
```

Dashboard PMD:

```
<div class="panel panel-info">
  <div class="panel-heading">
    <h3 class="panel-title">Violações por Classe</h3>
  </div>
  <div class="panel-body" id="Panel3">
    Panel content
  </div>
</div>
```

A estrutura *Components* foi dividida em quatro *Groups*, *Charts*, *Others*, *Generic* e *Selects*. O *Group Charts* é responsável pelos componentes *CCC pie Chart* e *CCC 100% Stacked Bar Chart*; O *Group Others* é responsável pelo *table Component*; *Group Generic* é responsável pela configuração do parâmetro utilizado para a construção do *selector* da *release* e o *Group Selects* é responsável pelas configurações do *selector* da *release*.

A perspectiva de *Datasources* possui apenas o *Group SQL Queries* e possui quatro consultas, todas do tipo *sql over sqljndi*. A primeira é responsável pela consulta do gráfico *CCC pie Chart* e possuem consultas do tipo SQL que estão descritas abaixo:

Dashboard Cenários:

```
select
    dsc.scenario_name as "Nome do Cenário",
    sum(fsc.quantity_Scenario) as "Quantidade"
from
    F_Scenario_Class fsc,
    D_Release re,
    D_Scenario_Clean_Code dsc
where
    fsc.D_Scenario_Clean_Code_idScenario = dsc.idScenario
    and re.idRelease = fsc.D_Release_idRelease
    and re.release_name = ${release_param}
group by
    dsc.scenario_name
```

Dashboard Findbugs:

```
select
    p.priority_name as "Prioridade", -- Para alterar o nome da
        coluna basta alterar a string entre aspas
    sum(fpb.quantiy_Bug) as "Quantidade"
from
    F_Project_Bug fpb,
    D_Priority p,
    D_Release re
where
    fpb.D_Class_Bug_idPriority = p.idPriority
    and re.idRelease = fpb.D_Release_idRelease
    and re.release_name = ${release_param} -- Colocar o parametro
        com a release desejada aqui
group by
    p.priority_name
```

Dashboard PMD:

```
select
```

```

    s.severite_name as "Severidade",
    sum(fpv.quantiy_Violation) as "Quantidade"
from
F_Project_Violation fpv,
D_Release re,
D_Severite s
where
fpv.D_Release_idRelease = re.idRelease
and fpv.D_Class_Violation_idSeverite = s.idSeverite
and re.release_name = ${release_param}
group by
s.severite_name

```

A segunda consulta é responsável pela consulta do componente *CCC 100% Stacked Bar Chart* e são descritas abaixo:

Dashboard Cenários:

```

select
    q.quality_index as "Index",
    m.metric_abbreviation as "Métrica"

from
F_Project_Metric fpm,
D_Release re,
D_Metric m,
D_Quality q
where
fpm.D_Quality_idQuality = q.idQuality
and fpm.D_Metric_idMetric = m.idMetric
and re.idRelease = fpm.D_Release_idRelease
and re.release_name = ${release_param}
group by
q.quality_index,
m.metric_abbreviation;

```

Dashboard Findbugs:

```

select
    p.priority_name as "Prioridade",
    t.type_category as "Tipo",

```

```

    sum(fpb.quantiy_Bug) as "Quantidade"
from
F_Project_Bug fpb,
D_Priority p,
D_Release re,
D_Type t
where
fpb.D_Class_Bug_idPriority = p.idPriority
and re.idRelease = fpb.D_Release_idRelease
and fpb.D_Class_Bug_idType = t.idType
and re.release_name = ${release_param}
group by
p.priority_name,
t.type_category;

```

Dashboard PMD:

```

select
    s.severite_name as "Severidade",
    r.rule_set as "Tipo",
    sum(fpv.quantiy_Violation) as "Quantidade"
from
F_Project_Violation fpv,
D_Release re,
D_Severite s,
D_Rules r
where
    fpv.D_Class_Violation_idSeverite = s.idSeverite
and re.idRelease = fpv.D_Release_idRelease
and fpv.D_Class_Violation_idRules = r.idRules
and re.release_name = ${release_param}
group by
s.severite_name,
r.rule_name;

```

A terceira consulta é responsável pela consulta do *table Component* e são descritas abaixo:

Dashboard Cenários:

```
select c.class_name as "Classe", dsc.scenario_name as "Nome do Cenário",
```

```
dsc.recomendations as "Recomendação"

from D_Project pj, D_Release re, F_Scenario_Class fsc,
D_Scenario_Clean_Code dsc, D_Class c

where

pj.idProject = fsc.D_Project_idProject
and re.idRelease = fsc.D_Release_idRelease
and fsc.D_Scenario_Clean_Code_idScenario = dsc.idScenario
and c.idClass = fsc.D_Class_idClass
and re.release_name = ${release_param};
```

Dashboard Findbugs:

```
select c.bug_file_class as "Classe", ty.type_category as "Tipo",
ty.type_name as "Nome", c.bug_line_class as "Linha"

from D_Class_Bug c, F_Project_Bug fpb, D_Type ty,
D_Release re

where

c.idClass_bug = fpb.D_Class_Bug_idClass_Bug
and ty.idType = fpb.D_Class_Bug_idType
and re.idRelease = fpb.D_Release_idRelease
and re.release_name = ${release_param};
```

Dashboard PMD:

```
select
    dcv.violation_file_class,
r.rule_set as "Tipo",
r.rule_name as "Nome",
dcv.violation_line_class as "Linha"

from
F_Project_Violation fpv,
D_Release re,
D_Severite s,
```

```
D_Rules r,
D_Class_Violation dcv
where
fpv.D_Release_idRelease = re.idRelease
and fpv.D_Class_Violation_idSeverite = s.idSeverite
and fpv.D_Class_Violation_idRules = r.idRules
and dcv.idClass_violation = fpv.D_Class_Violation_idClass_violation
and re.release_name = ${release_param}
```

A última consulta é responsável pela consulta do *selector* da *release* e são descritas abaixo:

Dashboard Cenários:

```
select DISTINCT re.release_name

from D_Release re, F_Project_Metric fpm

where

re.idRelease = fpm.D_Release_idRelease;
```

Dashboard Findbugs:

```
select DISTINCT re.release_name

from D_Release re, F_Project_Bug fpb

where

re.idRelease = fpb.D_Release_idRelease;
```

Dashboard PMD:

```
select DISTINCT re.release_name

from D_Release re, F_Project_Violation fpv

where

re.idRelease = fpv.D_Release_idRelease;
```

APÊNDICE C – Questionário



Questionário ICC₂ - Uso de um Ambiente de Data Warehousing para Aferição da Qualidade Interna de Software: um Estudo de Caso Preliminar na Caixa Econômica Federal

*Obrigatório

Em qual equipe da Caixa você trabalha? *

- CETEC/PEDeS
- GITECBR
- Outra

Qual a sua área de atuação? *

A quanto tempo você trabalha nessa área?

- menos de 2 anos
- de 3 a 5 anos
- mais de 5 anos

Você entendeu o conceito sobre cenários de limpeza? *

- Sim
- Não

Qual sua opinião quanto a detecção de cenários de limpeza de código através da solução DW?

*

- Ótimo
- Muito Bom
- Bom
- Ruim
- Muito Ruim
- Sem opinião

Você já utilizou a ferramenta PMD? *

- Sim
- Não

Qual sua opinião em relação a ferramenta PMD? *

- Ótimo
- Muito Bom
- Bom
- Ruim
- Muito Ruim
- Não se aplica

Qual a sua opinião quanto a visualização das análises de violação de código da solução de DW em comparação à visualização do PMD? *

- Muito Melhor
- Melhor
- Imparcial
- Pior
- Muito Pior

Você já utilizou a ferramenta FindBugs? *

- Sim
- Não

Qual sua opinião em relação a ferramenta FindBugs? *

- Ótimo
- Muito Bom
- Bom
- Ruim
- Muito Ruim
- Não se aplica

Qual a sua opinião quanto a visualização das análises de bugs de código da solução de DW em comparação à visualização do FindBugs? *

- Muito Melhor
- Melhor
- Imparcial
- Pior
- Muito Pior

Qual a sua opinião quanto a visualização das análises através do dashboard? *

- Ótimo
- Muito Bom
- Bom
- Ruim
- Muito Ruim
- Sem opinião

Você já utilizou a ferramenta sonarQube? *

- Sim
- Não

Qual sua opinião em relação a ferramenta sonarQube? *

- Ótimo
- Muito Bom
- Bom
- Ruim
- Muito Ruim
- Não se aplica

Em relação a visualizações de erros e violações, qual a sua opinião quanto ao uso da solução de DW em comparação à solução do sonarQube? *

- Muito Melhor
- Melhor
- Imparcial
- Pior
- Muito Pior

Na sua opinião qual o ponto forte da solução de dw?**Na sua opinião qual o ponto fraco da solução de dw?****Na sua opinião a solução apresentada traria benefícios para a CAIXA? ***

- Sim
- Não

Quais benefícios?

APÊNDICE D – Relatório KanbanFlow

Main Backlog (0 tasks)

Sprint Backlog (0 tasks)

In progress (2 tasks)

alterar conclusão

Responsible: Nilton Cesar C Time estimate: 0h Time spent: 2h

revisar parte escrita do tcc

revisar parte escrita do tcc

Responsible: Nilton Cesar C Time estimate: 15h Time spent: 12h

Done (26 tasks)

escrever sobre a execução e analises dos resultados do caso de uso.

Responsible: Nilton Cesar C Time estimate: 7h Time spent: 10h
Grouping date: 23 June 201

escrever sessão sobre metodologia de desenvolvimento

Responsible: Nilton Cesar C Time estimate: 3h Time spent: 3h
Grouping date: 23 June 201

escrever sobre correlação e erro quadratiko medio no TCC

Responsible: Nilton Cesar C Time estimate: 3h Time spent: 3h
Grouping date: 23 June 201

	<p>Documentar alterações no modelo existente</p> <hr/> <p>Responsible: Nilton Cesar C Time estimate: 3h Time spent: 2h Grouping date: 23 June 201</p>	
	<p>realizar entrevista com os questionarios</p> <hr/> <p>elaborar questionario</p> <hr/> <p>Responsible: Nilton Cesar C Time estimate: 2h Time spent: 2h Grouping date: 23 June 201</p> <p><input checked="" type="checkbox"/> validar guia com o Hilmer</p>	
	<p>redefinir protocolo do estudo de caso do tcc1</p> <hr/> <p>redefinir protocolo do estudo de caso do tcc1 e o gqm</p> <hr/> <p>Responsible: Nilton Cesar C Time estimate: 10h Time spent: 11h Grouping date: 23 June 201</p> <p><input type="checkbox"/> validar novo protocolo</p>	
	<p>analisar respostas dos questionarios</p> <hr/> <p>Responsible: Nilton Cesar C Time estimate: 2h Time spent: 2h Grouping date: 23 June 201</p>	
	<p>alterar resumo e abstract</p> <hr/> <p>Responsible: Nilton Cesar C Time estimate: 2h Time spent: 1h Grouping date: 16 June 201</p>	
	<p>escrever sobre o pmd no TCC</p> <hr/> <p>Responsible: Nilton Cesar C Time estimate: 2h Time spent: 2h Grouping date: 15 June 201</p>	
	<p>estudar ferramenta r</p> <hr/> <p>Responsible: Nilton Cesar C Time estimate: 3h Time spent: 3h Grouping date: 15 June 201</p>	
	<p>escrever sobre o findbugs no TCC</p> <hr/> <p>Responsible: Nilton Cesar C Time estimate: 2h Time spent: 2h Grouping date: 15 June 201</p>	

gerar os erro quadrático médio entre as analises do pmd e da solução de dw, por tipo e por release.
Responsible: Nilton Cesar C Time estimate: 0h Time spent: 10h Grouping date: 15 June 201
gerar os erro quadrático médio entre as analises do findbugs e da solução de dw, por tipo e por release.
Responsible: Nilton Cesar C Time estimate: 10h Time spent: 10h Grouping date: 15 June 201
gerar o coeficiente de correlação linear e os graficos de dispersao
Responsible: (none) Time estimate: 3h Time spent: 3h Grouping date: 15 June 201
correlacionar as métricas do pmd, findbugs e cenários de limpeza. com uma consultoria com o professor Nilton Correa foi decidido que n[ao caberia para o tcc a correlçao destas m[etricas.
Responsible: (none) Time estimate: 0h Time spent: 0h Grouping date: 30 May 201
estudar sobre correlações de métricas
Responsible: Nilton Cesar C Time estimate: 5h Time spent: 5h Grouping date: 30 May 201
procurar referencias sobre visualização de dados procurar referencias sobre visualização de dados para serem utilizados na criação de dashboard
Responsible: Nilton Cesar C Time estimate: 5h Time spent: 3h Grouping date: 15 May 201
ler capitulo 1, 2, 3 e 5 do livro Experimentation Software Engineering
Responsible: Nilton Cesar C Time estimate: 3h Time spent: 5h Grouping date: 15 May 201

alterar os dashboards para consultas sql, remover filtros de mês e data e deixar todos com a visualização do dashboard do findbugs

Responsible: Nilton Cesar C Time estimate: 7h Time spent: 7h
Grouping date: 4 May 2015

criar dashboard com os dados gerais com drill down para os demais.

Responsible: Nilton Cesar C Time estimate: 7h Time spent: 7h
Grouping date: 4 May 2015

Validar dashboard do cde com o Hilmer

Responsible: Nilton Cesar C Time estimate: 3h Time spent: 3h
Grouping date: 30 April 20

Criar um dashboard para o pmd, um para o findbug e outro para os scenarios de limpeza.

Responsible: Nilton Cesar C Time estimate: 15h Time spent: 15h
Grouping date: 30 April 20

criar dashboard no cde

Responsible: Nilton Cesar C Time estimate: 10h Time spent: 10h
Grouping date: 30 April 20

criar o processo de etl para o cubo do findbugs

Responsible: Nilton Cesar C Time estimate: 15h Time spent: 15h
Grouping date: 8 April 201!

adaptar o modelo estrela

adaptar o modelo estrela

Responsible: Nilton Cesar C Time estimate: 6h Time spent: 6h
Grouping date: 8 April 201

- validar modelo estrela inserindo cubo checkstyle
- validar modelo estrela inserindo cubo do findbugs
- validar modelo estrela inserindo cubo do pmd

criar o processo de etl para o cubo do pmd

Responsible: Nilton Cesar C Time estimate: 7h Time spent: 7h
Grouping date: 31 March 20