UNIVERSIDADE FEDERAL DO AMAPÁ

DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Gabriel Melo da Silveira

UMA PROPOSTA DE IDENTIFICAÇÃO DE COMPONENTES REUTILIZÁVEIS BASEADA EM INTERAÇÕES DE CLASSES

Trabalho de Conclusão de Curso

Macapá

2021

UNIVERSIDADE FEDERAL DO AMAPÁ

DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Gabriel Melo da Silveira

UMA PROPOSTA DE IDENTIFICAÇÃO DE COMPONENTES REUTILIZÁVEIS BASEADO EM INTERAÇÕES DE CLASSES

Trabalho de Conclusão de Curso submetido à Banca Examinadora do Curso de Ciência da Computação da UNIFAP para a obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Julio Cezar Costa Furtado

Macapá

2021

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Central da Universidade Federal do Amapá
Jamile da Conceição da Silva – CRB-2/1010

Silveira, Gabriel Melo.

S587p Uma proposta de identificação de componentes reutilizáveis baseado em interações de classes / Gabriel Melo Silveira. – 2021.

1 recurso eletrônico. 77 f.

Trabalho de conclusão de curso (Graduação em Ciência da Computação) – Campus Marco Zero, Universidade Federal do Amapá, Coordenação do Curso de Ciência da Computação, Macapá, 2021.

Orientador: Professor Doutor Júlio Cezar Costa Furtado

Modo de acesso: World Wide Web.

Formato de arquivo: Portable Document Format (PDF

Inclui referências.

1. Engenharia de Software. 2. Software - Computação. 3. Componente de software. 4. Software - Desenvolvimento 5. Software – Reutilização. I. Furtado, Júlio Cezar Costa, orientador. II. Título.

Classificação Decimal de Dewey, 22 edição, 005.1

SILVEIRA, Gabriel Melo. **Uma proposta de identificação de componentes reutilizáveis baseado em interações de classes**. Orientador: Júlio Cezar Costa Furtado. 2021. 77 f. Trabalho de conclusão de curso (Graduação em Ciência da Computação) – Campus Marco Zero, Universidade Federal do Amapá, Coordenação do Curso de Ciência da Computação, Macapá, 2021.

UNIVERSIDADE FEDERAL DO AMAPÁ
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

ATA DE DEFESA DE TCC

Realizou-se no dia 20 de dezembro de 2021, às 15h00, via videoconferência pelo Google Meet, adefesa do TCC intitulado: **"UMA PROPOSTA DE IDENTIFICAÇÃO DE COMPONENTES REUTILIZÁVEIS BASEADA EM INTERAÇÕES DE CLASSES"**, da discente GABRIEL MELO DA SILVEIRA, matricula 201412200037. A Banca Examinadora foi composta pelo Prof. Dr. JULIO CEZAR COSTA FURTADO, presidente da banca e orientador; Prof. Me. ADOLFO FRANCESCO DE OLIVEIRA COLARES e Prof. Me. MARCO ANTÔNIO LEAL DA SILVA, examinadores. Concluída a defesa, foram realizadas as arguições e comentários. Em seguida, procedeu-se o julgamento pelos membros da Banca Examinadora, tendo o trabalho sido APROVADO com nota 9,0.

E, para constar, eu, Prof. Dr. JULIO CEZAR COSTA FURTADO, orientador e presidente da Banca Examinadora, lavrei a presente ata que, após lida e achada conforme, foi assinada por mim e demais membros da Banca Examinadora.

Macapá, 20 de dezembro de 2021.

Prof. Dr. JULIO CEZAR COSTA FURTADO
Orientador do TCC

ADOLFO FRANCESCO DE OLIVEIRA COLARES:74382080282
Assinado de forma digital por ADOLFO FRANCESCO DE OLIVEIRA COLARES:74382080282

Prof. Me. ADOLFO FRANCESCO DE OLIVEIRA COLARES
Examinador (UNIFAP)

Prof. Me. MARCO ANTÔNIO LEAL DA SILVA
Examinador (UNIFAP)

AGRADECIMENTOS

Em primeiro lugar quero agradecer imensamente a minha mãe, Joana Melo, que me ajudou em todos os momentos e me incentivou nessa jornada na universidade desde o primeiro momento. Quero agradecer a minha esposa, Sara Flausino, por toda a ajuda e apoio para superar os obstáculos durante a graduação.

Agradeço também ao meu orientador, Prof. Dr. Julio Furtado, ao suporte prestado, as dúvidas esclarecidas e a paciência durante toda a elaboração e escrita deste trabalho.

Memorar as amizades feitas durante o curso, que me acompanharam e ajudaram em diversas situações, e me proporcionaram momentos impares durantes todos os encontros e conversas.

Em conclusão, agradeço ao colegiado e o corpo técnico do curso de Ciência da Computação, e a todos que colaboram e estiveram presentes para conclusão desse trajeto.

RESUMO

A engenharia de software baseado em componentes reutilizáveis tem por objetivo principal o reuso de software, para isso ela aplica o processo reutilização utilizando partes de softwares. Essa prática possibilita a economia de recursos, simplifica modificações e implantações. Visando nos benefícios deste processo, este trabalho apresenta uma proposta de metodologia para seleção de classe com potencial de reutilização. A metodologia tem como recursos para identificação a engenharia reserva, que gera diagramas de classe possibilitando o mapeamento estrutural do software, a contabilização de interações entre as classes e a aplicação do conceito de estatístico do desvio padrão, para identificar as classes singulares dentro do conjunto. Esta pesquisa tem como objetivo abordar detalhadamente sua elaboração, contribuir para a evolução do processo de reuso no desenvolvimento de softwares e destacar a importância deste ramo da engenharia de software.

PALAVRAS-CHAVE: reutilização; metodologia; desvio padrão; engenharia reversa; componentes.

ABSTRACT

Software engineering based on reusable components has as its main objective the reuse of software, for that it applies the reuse process using parts of software. This practice saves resources, simplifies modifications and deployments. Aiming at the benefits of this process, this work presents a proposed methodology for class selection with potential for reuse. The methodology has as resources for identification the reserve engineering, which generates class diagrams enabling the structural mapping of the software, the accounting of interactions between classes and the application of the standard deviation statistical concept, to identify the singular classes within the set. This research aims to approach its elaboration in detail, contribute to the evolution of the reuse process in software development and highlight the importance of this branch of software engineering.

KEYWORDS: reuse; methodology; standard deviation; reverse engineering; components.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AE	Associação por Envio
AS	Associação por Solicitação
CBSE	*Component-Based Software Engineering*
CRUD	*Create, Read, Update, Delete*
DA	Dependência sendo Alimentando
DD	Dependência sendo Dependente
DP	Desvio Padrão
ES	Engenharia de Software
ESBC	Engenharia de Software Baseada em Componentes
IDE	*Integrated Development Environment*
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

A reutilização de software tem sido uma das áreas mais importantes para desenvolvimento da produtividade e qualidade do software (JHA E O'BRIEN, 2011). Tendo como objetivo o aumento da reutilização de software e devido o desenvolvimento orientado a objetos não ter gerado um aumento de reuso de software, os projetistas criaram a engenharia baseada em componentes, visando justamente a reutilização de software em maior escala (SOMMERVILLE, 2011).

A ideia de reutilização de software iniciou-se juntamente com o limiar da programação, surgindo por meio do reuso de ideais, abstrações e processos (PRESSMAN, 2011). A noção rudimentar de reutilizar fragmentos de software existe há várias décadas, originando-se com a noção de sub-rotina. O reuso compreende em usar software existente para a produção de novos softwares. A reutilização não se limita a utilização de fragmentos de código fonte, não obstante ele contempla todo o trabalho produzido durante o processo de desenvolvimento de software (Guerra e Santos, 1999).

A prática do reuso de software oferece, sem determinar um processo rígido, uma pluralidade de condições que possibilitam a adaptação às dificuldades de um projeto (GUERRA E SANTOS, 1999).

Szperski (1997) afirma que a Engenharia de Software Baseada em Componentes é um avanço no reuso de software, por possibilitar que o componente seja reutilizado sem alteração na sua integração, sem custos de desenvolvimento, somente na montagem da estrutura com os componentes. Sua proposta proporcionar um módulo autônomo, de fácil implantação, que não altere a estrutura do software e padronizado, com todas suas informações documentadas. Para que o desenvolvedor tenha uma facilidade maior e economia de tempo, construindo a estrutura do software utilizando esses blocos modulares.

Em vista disso, essa pesquisa está baseada na Engenharia de Software Baseada em Componentes, propondo o desenvolvimento de uma metodologia identificação de componentes reutilizáveis para sistemas de volume médio e grande, aplicar a metodologia em sistemas para obtenção de componentes com a maior precisão possível, realizar um comparativo com outras metodologias de identificação de componentes e analisar os resultados.

1.1. Motivação, Justificativa e Contribuição à área

Com o desenvolvimento de software para linguagem de programação de alto nível, o reuso era realizado com funções e objetos, obtidos de bibliotecas, porém havia limitações no

custo e cronograma. Devido a essas adversidades a produção de software buscando ferramentas e práticas para uma produção de maior velocidade, menor custo e melhor qualidade, consequentemente surgiu a abordagem de reuso de código. Com a expansão do desenvolvimento de softwares a prática tomou proporções maiores, com os níveis de abstração, objeto, componente e sistema (SOMMERVILLE, 2011).

A reutilização de software ajuda a aumentar sua produtividade e a qualidade de projetos. Além disso, ela pode ser aplicada a qualquer ciclo de vida de softwares, não sendo exclusiva apenas do código fonte (JONES, 1993). Fruto do avanço do reuso de software, a Engenharia de Software Baseada em Componentes apresenta um grande avanço no reuso de software, devido possibilitar o reuso de maneira mais simplificada, utilizando um esquema de "montagem" da estrutura do software, maiores sem custos de desenvolvimento (SOFTEX, 2007).

A Engenharia de Software Baseada em Componentes tem seu foco no desenvolvimento de componentes reutilizáveis com o objetivo que sejam utilizados em diversos softwares, em vez de serem utilizados apenas em seus softwares de origem (AMPATZOGLOU *et al.*, 2012). Componentes são independentes, logo não conflitam entre si e sua implantação em um sistema não afeta o resto da estrutura. Sua comunicação funciona por meio de interfaces definidas, e a substituição de um componente por outro pode ser realizada sem causar problemas. Além que sua infraestrutura oferta um conjunto de serviços-padrão que podem ser utilizados, dispensando a produção desses serviços pela equipe de desenvolvimento (SOMMERVILLE, 2011). O componente pode ser apenas um fragmento do código-fonte até mesmo um sistema inteiro, além disso, ele pode estar na forma de projeto ou código para ser incorporado a um sistema (ROSSI, 2004).

Neste cenário, a utilização de componentes reutilizáveis é a medida mais viável para a produção de softwares, pois diminui o custo de produção, exigindo apenas a identificação de componentes praticamente prontos no interior de sistemas. O uso de componentes proporciona produtividade ligados ao reuso, há ganhos de qualidade e funcionalidade fornecidos pela tecnologia. Além que o fato de um componente ser reutilizado em diversas situações expõe ele a vários testes e tenha seus erros corrigidos, chegando maturidade mais rapidamente (SOFTEX, 2007).

A engenharia de software baseada em componentes engloba a responsabilidade com a reutilização de software, com foco no aumento da produtividade e a redução dos custos com a qualidade (SPAGNOLI *et al*., 2004). Sendo assim, a identificação de componentes tem grande potencial para melhorar o desenvolvimento de sistemas.

A produção de sistemas exige uma rápida produção, para sua entrega ao cliente, porém a codificação manual de um sistema começando do zero demanda bastante tempo e um grande esforço. Necessitando assim de instrumentos para o auxiliar na produção, reduzindo o tempo de produção e os gastos com desenvolvimento, mas principalmente melhorando a qualidade dos softwares. E a reutilização de software é uma das técnicas essenciais para o aumento da produtividade de desenvolvimento de softwares (OLIVEIRA 2007).

A reutilização de software tem como objetivo procurar aumentar a qualidade e a produtividade do desenvolvimento de software, buscando impedir a duplicação do trabalho e reaproveitando fragmentos de projetos passados (LUCRÉDIO, 2009). Com o crescimento do mercado de softwares, as empresas procuram cada vez mais a qualidade e a produtividade na produção de softwares. Então o reuso se torna alternativa, pois apresenta benefícios por meio da ESBC, que logo presume-se a existência de componentes para o reuso (SPAGNOLI *et al.*, 2003). A reutilização de software abrange o projeto por inteiro, porém não é comum em todos os projetos realizados, devido não ser apenas uma técnica, e sim um processo contínuo de uso de componentes já desenvolvidos e a criação de novos (FILHO, 2013).

Há poucos estudos sobre métodos de identificação ou extração de componentes reutilizáveis, entre eles foram encontrados apenas os seguintes: “A Methodology On Extracting Reusable Software Candidate Components From Open Source Games”, descrita no artigo (AMPATZOGLOU *et al.*, 2012); “A Method Based on Naming Similarity to Identify Reuse Opportunities”, representada pelo artigo (OLIVEIRA *et al*., 2016); “Automatic Identification of Reusable Software Development Assets: Methodology and Tool”, exposta no artigo (OLIVEIRA *et al*., 2007). A pouca quantidade de pesquisas agrava ainda mais o quadro de reuso de software e os poucos estudos da ESBC.

Nesse contexto, adotar uma nova metodologia para a identificação de componentes reutilizáveis, baseada na quantidade de interações das classes, pode resultar em um aumento da utilização de componentes durantes o processo de desenvolvimento de softwares, necessitando apenas a análise de sistemas de um domínio em comum, para identificar os potenciais componentes para reuso.

Esta pesquisa visa colaborar com aumento do reuso de software em projetos de desenvolvimento de software, mas principalmente, gerando uma nova solução no setor de identificação de componentes reutilizáveis abordado pela CBSE, ele sendo um dos ramos mais promissores da Engenharia de software, porém pouco abordado.

1.2. Objetivos

O projeto tem como principal foco o desenvolvimento de uma metodologia para identificar possíveis componentes reutilizáveis. Para o atendimento de tal resultado, os objetivos a seguir devem ser contemplados.

1.2.1. Objetivos Específicos

1. Desenvolver a metodologia;
2. Aplicar a metodologia em sistemas para obtenção de componentes com a maior precisão possível;
3. Realizar testes comparativos com outras metodologias;
4. Analisar os resultados do método em relação a outra metodologia.

1.3. Metodologia de Pesquisa

A metodologia empregada nesta pesquisa será composta por quatro etapas, onde a primeira etapa é composta por um estudo da literatura sobre Engenharia de Software Baseada em Reuso, abordando sua origem, os seus princípios e seus benefícios. Assim como foi estudado também Engenharia de Software Baseado em Componentes, explorando suas características e análises sobre sua relevância no desenvolvimento de softwares. Esta fase será a base de toda pesquisa, sendo o referencial teórico necessário para a compreensão e produção da metodologia.

A segunda etapa é composta pela avaliação de abordagens existentes, que irá analisar pesquisas e metodologias de identificação de componentes reutilizáveis, para que haja uma referência no desenvolvimento da metodologia que será proposta. Nesta etapa, é pesquisado trabalhos com o tema de identificação de componentes reutilizáveis, e são avaliadas as metodologias utilizadas nessas pesquisas, seu conjunto de métricas para a definição de reutilização, quais parâmetros de sistemas serem avaliados, para que haja uma noção do que adotar ou corrigir.

A terceira etapa consiste no desenvolvimento da metodologia, a criação do conjunto de regras que determinarão a construção e execução de parâmetros para obtenção de componentes reutilizáveis. Esta etapa será definida os critérios de linguagem de programação e domínio analisados, métricas para definir um componente reutilizável, ferramentas utilizadas e o passo a passo da execução da metodologia.

E por último, a quarta etapa é composta de teste e comparação dos resultados, após o desenvolvimento da metodologia, ela será executada e comparada a outras metodologias

analisadas na segunda etapa. Nesta etapa será escolhido um domínio para teste, procedimento proposto será executado, assim como outra sistemática, os resultados obtidos serão comparados, após a comparação será analisado se o processo teve um desempenho superior, igual ou inferior a das outras metodologias e em seguida haverá uma tomada de decisão sobre a sistemática baseada no seu desempenho para correção ou mantê-la.

1.4. Estrutura do Trabalho

Esta seção tem por objetivo a descrição dos capítulos posteriores deste trabalho, iniciando pelo capítulo 2. O próximo capitulo tratará da contextualização teórica do trabalho, apresentando os conceitos utilizados como base do estudo, e cada seção descreverá cada componente que compõe a pesquisa.

A seção 2.1. inicia o capítulo com a contextualização da Engenharia de Software, área da ciência da computação que o trabalho se fundamenta, apresentando as definições que descrevam os objetivos, funcionamento e paradigmas deste campo. Em seguida, a seção 2.2. apresentará um conceito estudado pela engenharia de software, o reuso de software, utilizado como base deste trabalho, e mostrará qual sua importância de empregar esse conceito em um projeto, assim como quais suas características e classificação.

Posteriormente, o capítulo apresenta na seção 2.3., um dos ramos da engenharia de software, a Engenharia de Software Baseada em Componentes, descrevendo qual objetivo, quais suas características e sua aplicação em um projeto. E finalizando o capítulo 2, a seção 2.4. encerra com os trabalhos científicos estudados, que também usam como base esses conceitos.

O capítulo 3 abordará a metodologia de identificação de componentes proposta neste trabalho e quais suas etapas. Primeiramente, na seção 3.1. será abordada as bases conceituais da engenharia de software, com foco na estruturação de um software através do diagrama de classes em UML para realizar a engenharia reversa do código-fonte. Além de abordar o conceito do desvio padrão para analisar dados que destoam em conjunto de valores.

Em seguida, na seção 3.2. será descrita cada etapa da metodologia proposta neste trabalho, desde a realização da engenharia reversa, como ela acontece, a contabilização e classificação de interações, e finalmente a seleção de potenciais classes reutilizáveis.

Por fim, os comentários finais sobre o capítulo nas considerações finais na seção 3.3. encerrando a apresentação da metodologia de identificação de componentes.

Por conseguinte, o capítulo 4 irá expor a execução da metodologia proposta, e para avaliar seu desempenho, será apresentado um comparativo com outra pesquisa. Inicialmente a seção 4.1. abordará a metodologia apresentada neste trabalho, que será executada em três

projetos. Discorrendo os parâmetros para seleção de projetos, e a aplicação até os resultados em cada um dos cenários.

Em seguida, na seção 4.2. executada a metodologia citada nos trabalhos relacionados, “A Method Based on Naming Similarity to Identify Reuse Opportunities”. O trabalho será replicado nos mesmos cenários, para que se possa analisar seus resultados.

Com os resultados obtidos das duas metodologias, a seção 4.3. tratará de realizar o comparativo dos dados atingidos. A análise comparativa ocorrerá por meio de tabelas e gráficos, avaliando paralelamente o desempenho das duas pesquisas.

Finalizando o capítulo, a seção 4.4. encerra apresentação e o comparativo das metodologias de identificação de componentes reutilizáveis, tecendo breves comentários.

No último capítulo do trabalho, capítulo 5, discorrerá sobre as considerações finais da pesquisa, apresentando um breve resumo do que foi apresentado. Logo após, na seção 5.1., serão analisados os resultados obtidos na execução da metodologia, e se esta teve um desempenho satisfatório em comparação a outra metodologia de identificação. Concluindo o trabalho, a seção 5.2., discutirá as previsões para a evolução da metodologia proposta.

2 CONTEXTUALIZAÇÃO E TERMINOLOGIAS DO TRABALHO

2.1. A Engenharia de Software

Durante 1965 a 1975, correu uma quantidade alarmante de softwares com algum tipo de erro, em que os desenvolvedores não conseguiam corrigir essa quantidade de erros com o processo de manutenção, devido neste período as empresas só visualizavam o software como um mero programa, produzido em alguma linguagem programação, e não focavam nas possíveis falhas humanas que poderiam causar erros no programa, este período foi denominado Crise de Software (SOUZA, 2014). Tendo em vista solucionar esse conjunto de erros, as primeiras técnicas para prevenir tais problemas, proveu concepção a ideia de Engenharia de Software (SOUZA, 2014).

Segundo Sommerville (2011), a engenharia de software tem como objetivo auxiliar a produção de software profissional, contendo em si um leque de técnicas, que incentivam a especificação, projeto e evolução do software, que geralmente não tem uso na produção individual.

Friedrich Bauer (1969) conceituou Engenharia de Software como uso de fundamentos da engenharia para a produção de um software, que seja confiável e que seja funcional durante sua execução.

Essa área da computação orientada para a documentação, análise e gerência dos processos que compõe a produção de software, para a garantia de qualidade, desempenho e execução correta. Por meio dela são estudados cada etapa da produção do software, começando desde a organização de ideias, passando pela estruturação do sistema, produção, avaliação e entrega, cada processo sendo gerenciado e analisado, visando o menor custo de recursos e a maior qualidade. Sempre avaliando não apenas os processos micros, mas também toda a estrutura de processos, examinando o modelo de produção.

Figura 1 - Camadas da Engenharia de Software

Fonte: Pressman (2011)

Pressman (2011) afirma que a engenharia de software é uma tecnologia de camadas, tendo como base o foco na qualidade, e sucedido por processo, métodos e o seu topo sendo ferramentas. Resumindo, Sommerville (2011) define Engenharia de software como uma disciplina de engenharia que abrange todos os pontos do desenvolvimento de um software, desde sua concepção ao uso.

2.2. Reutilização de Software

Sametinger (1997) explica que a matriz do reuso ocorreu devido muitos sistemas conterem fragmentos parecidos ou idênticos que são desenvolvidos repetidamente do zero, logo isso motivou o reuso desses fragmentos. A reutilização de software é uma estratégia fundamental durante o desenvolvimento de sistemas, seguindo a ideia de armazenar trechos de códigos, funções, classes ou métodos de um domínio, para seu uso posterior em outro sistema.

Frakes (2014) resume reutilização de software como uso de conhecimento ou artefatos de software existentes para desenvolvimento de um novo software.

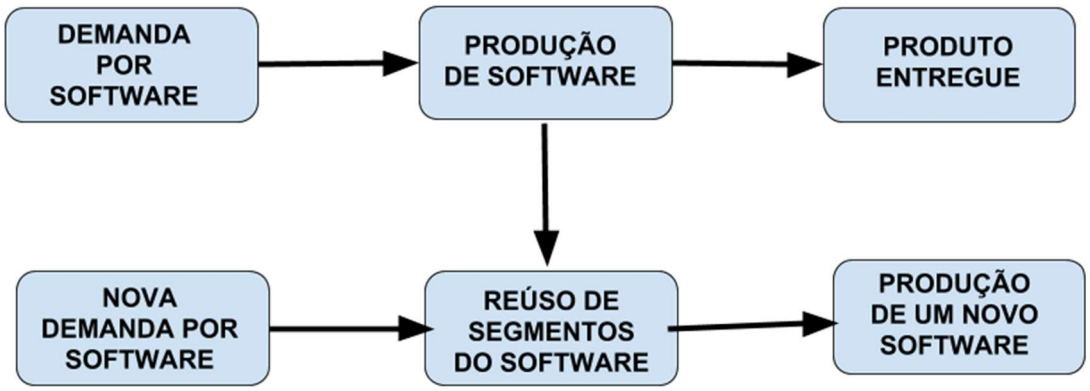
Sommerville (2011) afirma que a engenharia baseada em reuso tem como objetivo maximizar o reuso de softwares; trazendo assim benefícios ao projeto como aumento na produção, diminuição de risco e a redução de custo. A reutilização de um software vem como uma ferramenta útil, assim necessitando apenas de algumas modificações para sua inclusão no sistema.

Sametinger (1997) esclarece que a reutilização de software tem um impacto positivo na qualidade do software e nos custos da produção. Ele cita alguns benefícios da reutilização, por exemplo, para a qualidade do software, o acúmulo da correção de erros da reutilização para reutilização. Assim como para a produtividade, diminuindo a quantidade de código a ser produzido.

A reutilização de software tem vários níveis, dependendo do tamanho unidade a ser reaproveitada. O reuso de software pode ser feito de um sistema de aplicação, componente ou objetos e funções, dependendo apenas da demanda de produção da equipe. Os níveis de reutilização são classificados em: (I) Reuso de código fonte, em que trechos do código são reutilizados em outro software ou nele mesmo; (II) Reuso de partes de software, é o reuso de parcelas de sistemas em diversos projetos; (III) Integração dinâmica de componentes de diversas fontes, que são aplicações prontas utilizadas na forma de *plug-in*; (IV) Componetização, esse nível é caracterizado pela atualização e integração ao software de maneira dinâmica (SOFTEX, 2007).

Sommerville (2002) declara que o reuso de software beneficia-se do fato de softwares de mesmo domínio compartilham de semelhanças no código e por isso tem grande potencial de reuso.

Figura 2 - Ciclo de Vida da Reutilização de Software



Fonte: Elaborada pelo autor

2.3. Engenharia de Software Baseada em Componentes

Segundo Rossi (2004) um componente trata-se de um pacote coeso de artefatos de software podendo ser independentemente produzido e distribuído como uma unidade, e que pode ser constituído por outros componentes para desenvolver uma parcela maior de um software. Guerra e Silva (1999) faz o seguinte comparativo, os componentes são essencialmente análogos a objetos, sua natureza específica ocorre somente na produção de software.

Surgindo a partir da reutilização de software, essa subdivisão da Engenharia de Software utiliza do conceito da reuso, porém, normalizando este através de componentes. A engenharia de software baseado em componentes é um processo que produz sistemas utilizando componentes reutilizáveis (PRESSMAN, 2011). Realizando uma analogia, o desenvolvimento em utilizando componentes pode ser comparado ao uso das peças de montar Lego®, que permitem a montagem de diversos objetos, de várias maneiras devido aos seus vários formatos e encaixes, dependendo apenas da criatividade do construtor, da mesma maneira funcionam os componentes, que podem ser conectados, estruturando o software (SOFTEX, 2007).

Szyperki (1997) afirma que um componente é um pacote desenvolvido e testado separadamente, depois ele é difundido como um elemento que tem o potencial para ser integrado a outros componentes para assim estruturar algo com mais funcionalidades.

Para Pressman (2011) componente é um bloco construtivo modular para software de computador. Já Sommerville (2011) conceitua que os componentes são abstrações de nível mais alto do que objetos e são definidos por suas interfaces.

Os componentes têm a obrigação de serem produzidos de maneira tão independentes de determinado contexto o máximo possível, de maneira a permitir a seu aperfeiçoamento e integração (GUERRA E SILVA, 1999).

Baseando-se nos conceitos citados, um componente é uma estrutura padronizada, independente, para reaproveitamento de software e integração em sistemas.

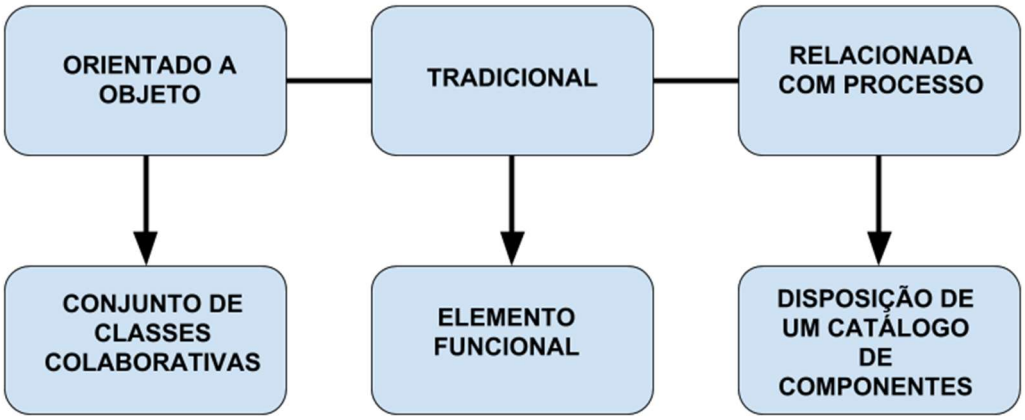
Tabela 1 - Diferenças entre ESBC e Engenharia de Software Tradicional

ENGENHARIA DE SOFTWARE TRADICIONAL	**ENGENHARIA DE SOFTWARE BASEADA EM COMPONENTES**
Análise dos requisitos do cliente	Análise dos requisitos do cliente
Especificação do sistema a ser desenvolvido	Especificação do sistema a ser desenvolvido
Aprovação da especificação pelo cliente	Aprovação da especificação pelo cliente
Projeto da arquitetura do software, onde alguns desenhos de projeto podem ser reutilizados	Busca e seleção dos componentes que serão utilizados
Implementação do programa, buscando reutilizar funções já desenvolvidas para outros projetos;	Desenvolvimento das partes que não foram atendidas por componentes já existentes
Testes	Integração
Implantação do sistema no cliente	Testes de integração
	Implantação do sistema no cliente

Fonte: SOFTEX (2007)

Pressman (2011) classifica o componente de três maneiras, dependendo do ponto de vista adotado. O primeiro ponto de vista é do orientado a objeto, na qual o componente engloba um conjunto de classes colaborativas, e cada classe foi criada para conter todos os atributos relevantes para sua implementação; para a visão da engenharia de software tradicional, o componente é um elemento funcional, que contém a lógica de processamento, estrutura de dados e um interface, montando assim uma estrutura que pode implementada e reaproveitada; e por último a visão relacionada com processos, em que um catálogo de componentes é colocado à disposição da equipe de desenvolvimento enquanto surge a demanda por componentes.

Figura 3 - Classificação de Componentes



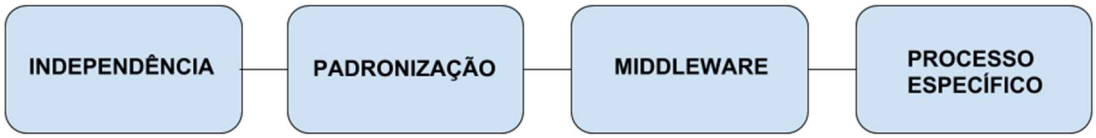
Fonte: Sommerville (2011)

2.3.1. Características

Sommerville (2011) classifica a engenharia baseada em componentes em quatro fundamentos principais, o primeiro fundamento é a independência, como são módulos independentes sua implementação ou sua remoção não pode afetar o resto do sistema. Seu segundo fundamento trata-se da padronização, todo componente precisa seguir normas do modelo de componente, essas que definem a interface e comunicação do componente.

O terceiro fundamento, *Middleware*, é a ponte entre o componente e o sistema para sua implantação, ele também fornece suporte para alocação de recursos, gerenciamento de transações, proteção e concorrência. O componente precisa ser implantável, essa característica determina que o componente deve agir como uma entidade autônoma. E o quarto fundamento, documentação, uma característica principal, na qual todo componente deve ser documentado, para que ao ser utilizado, o usuário tenha acesso a todas suas informações.

Figura 4 - Características da CBSE



Fonte: Elaborada pelo autor (2018)

2.4. Trabalhos relacionados

A tese intitulada "*A Method Based on Naming Similarity to Identify Reuse Opportunities*", representada aqui pelo artigo (OLIVEIRA *et al*., 2016), aborda um método para a identificação de possíveis componentes, baseado na semelhança entre os nomes das classes de sistemas de mesmo domínio, aplicado em uma ferramenta, chamada JReuse, que faz análise de sistemas e em que ranqueia a possibilidade de sua reusabilidade. Seu estudo vem do advento de analisadores de similaridades, e cita exemplos de comparação de dialetos, verificação ortográfica e detector de plágio, para tal comparativo, o autor utiliza uma adaptação do algoritmo de Levenshtein Distance, em que analisa e calcula a semelhança lexical das classes e métodos de softwares.

Oliveira *et al.* (2016) em sua metodologia utilizada para extração de possíveis componentes aplica o algoritmo de Levenshtein Distance para realizar o comparativo, focado nos principais elementos do paradigma de desenvolvimento orientado a objetos que são as classes e métodos. Outra exigência desse método é limitar essa abordagem apenas a sistema desenvolvidos na linguagem de programação Java, que contenham a quantidade mínima de 500 linhas de código.

O método proposto é aplicado sobre os nomes das classes e métodos do sistema avaliado, adotando um percentual mínimo de similaridade de 80% entre os nomes das classes e métodos, em seguida é montada uma lista com os nomes que atingem a porcentagem exigida, as classes e métodos com as maiores taxas de similaridade são melhores classificadas nesta lista.

A pesquisa intitulada "*Automatic Identification of Reusable Software Development Assets: Methodology and Tool*", representada aqui pelo artigo (OLIVEIRA *et al*., 2007) aborda um método de identificação de componentes por meio de engenharia reversa em sistemas legados. Essa metodologia é aplicada por meio de uma ferramenta chamada *DigitalAssets Discoverer*. A metodologia proposta é composta por 4 etapas: A primeira é a digitalização dos

softwares para a análise; na segunda, é criada uma base de conhecimento fundamentado na análise do código-fonte do software analisado, e todas suas características. A terceira etapa trata-se da execução de 3 metodologia de identificação de ativos; por último, o *DigitalAssets Discoverer* apresenta os resultados como possíveis ativos reutilizáveis, e o analista decide se utiliza algum dos resultados como componente e fazer sua exportação.

Outra pesquisa intitulada "*A Methodology On Extracting Reusable Software Candidate Components From Open Source Games*", representada aqui pelo artigo (AMPATZOGLOU *et al.*, 2012) propõe uma metodologia de extração de componentes de jogos de código aberto, da linguagem Java, essa metodologia utiliza algoritmos fortes baseados em caminho. Para executar a metodologia há necessidade de criar um gráfico das dependências entre as classes do sistema. Inicialmente acontece a criação de uma estrutura de dados para o armazenamento dos possíveis componentes, em seguida as classes são identificadas, depois são listadas a quantidade de dependências externas em ordem decrescente, cada dependência da lista se torna um possível componentes que são armazenados na estrutura de dados, posteriormente a etapa 2 é repetida. A metodologia ainda contém uma análise estrutural de artefatos e interdependências, uma análise de dominância e uma matriz de estrutura de dependência.

A pesquisa intitulada "*A Survey of Business Component Identification Methods and Related Techniques*", representada pelo artigo (WANG *et al.*, 2005) explana sobre as metodologias de identificação de componentes, classificando-as em quatros tipos: (I) Metodologias baseadas em engenharia de domínio, (II) Métodos de análise de agrupamento baseado em coesão-acoplamento, (III) Métodos baseados em CRUD, (IV) Outros métodos.

As metodologias baseadas engenharia de domínio, têm seu foco na reutilização da arquitetura do domínio e a adaptação do componente, não visando tanto o desempenho. Enquanto que os métodos de análise de agrupamento baseados em coesão-acoplamento, realiza um cálculo da força das dependências semânticas entre duas unidades empresariais e converte os modelos comerciais em gráfico direcional ponderado, posteriormente, reúnem o gráfico utilizando técnicas de agrupamento de gráficos ou matrizes. O método de análise de agrupamento baseado coesão-acoplamento tem seu uso na estatística matemática para classificação precisa. Ele integra os elementos com alta coesão para produzir padrões específicos e seu uso é recorrente no campo da mineração de dados e reconhecimento de padrões (WANG *et al.*, 2005).

As metodologias baseadas em matrizes CRUD, utilizam elementos de negócios comportamentais, como caso de uso, eventos, operações e elementos de negócios estáticos, como entidades comerciais. Esses métodos empregam quatro semânticas de relacionamentos

que dão origem ao seu nome (*Create*, *Read*, *Update*, *Delete*), tendo como prioridades a sequência *Create* > *Delete* > *Update* > *Read*. E por último, outros métodos, esses são métodos em que ainda não há uma técnica madura, como por exemplo, método de identificação de componentes orientado para decomposição de negócios, baseado em similaridade, de decomposição orientada à variação, baseado na minimização de perda de informação e baseado em estabilidade do modelo de negócios.

3 METODOLOGIA DE IDETIFICAÇÃO DE COMPONENTES

Neste capítulo será abordado a estrutura da metodologia de identificação de componentes, quais suas etapas, funcionamento e técnicas necessárias para sua utilização. A sistemática de identificação está fundamentada no uso da medida de desvio padrão e da engenharia reversa para visualização, análise e lógica de extração de potenciais componentes reutilizáveis.

3.1. Base para a criação da metodologia

Nesta etapa serão descritos os conceitos que servirão de base para a estruturação e desenvolvimento da metodologia de identificação de potenciais classes para reuso, proposta neste trabalho.

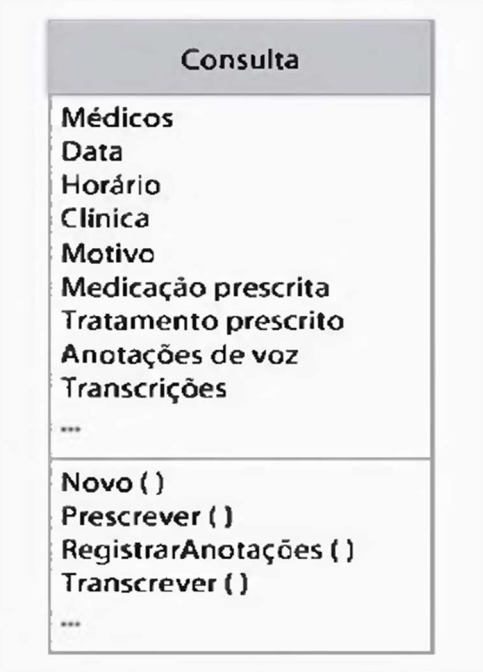
3.1.1. Diagrama de Classe em UML

Guedes, G.T.A (2014), descreve UML como uma linguagem visual, que tem como característica fundamental a modelagem sistemas empregando o paradigma de Orientação a Objetos. Seu principal objetivo é auxiliar na definição de características cruciais de cada etapa da composição e comportamento de um software, produzindo a estrutura antes mesmo do software ser programador.

Dentro desta linguagem existem vários tipos de modelagem, um dos mais utilizados é o Diagrama de Classe, segundo Sommerville (2011) o diagrama de classe tem como objetivo auxiliar na produção de um software orientado a objetos exibindo as suas classes e as associações entre elas.

As classes são representadas por um retângulo divido em três setores, esses que compõem as três características de uma classe: 1. Nome da classe; 2. Os atributos da classe; e 3. Suas operações, como pode ser visualizado na Figura 5. O nome da classe fica localizado no setor superior, para identificar classe, no setor do meio pertence aos atributos da classe, esta que corresponde à característica mais crítica da classe, segundo Pressman (2011), os atributos definem a classe, definindo o que esta simboliza, descrevendo suas propriedades. Por último, as operações situam-se no terceiro setor, as operações correspondem aos métodos em Java e outras linguagens, segundo Sommerville (2011).

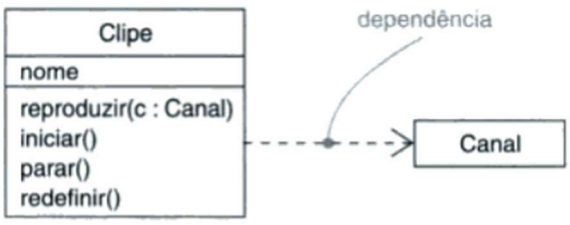
Figura 5 - Representação de uma Classe



Fonte: Sommerville (2011)

Segundo Pressman (2011), na UML os relacionamentos entre as classes denominados como associações, essas que são classificadas como: dependências, associações e generalizações. Booch *et al.* (1998) conceitua as dependências como uma relação em que uma classe utiliza uma característica de outra classe, gerando uma correlação entre as classes, a Figura 6 mostra a representação gráfica deste relacionamento, esse que é composto por uma linha pontilhada com uma seta entre as classes.

Figura 6 - Representação da Dependência entre Classes

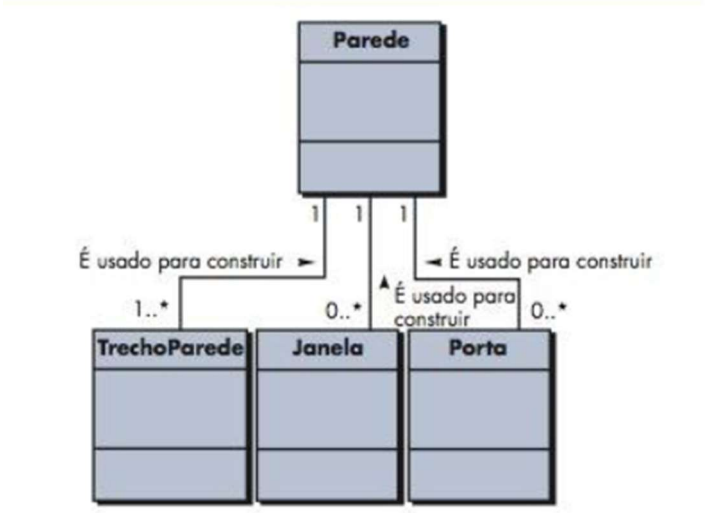


Fonte: Booch *et al.* (1998)

Booch *et al.* (1998) descreve as associações como um tipo de relacionamentos estrutural entre classes, uma ligação direta entre classes, constituindo um conjunto dentro do sistema. Sua

representação gráfica é composta por uma linha ou por uma linha com uma seta, esse tipo de relacionamento pode ser visualizado na Figura 7.

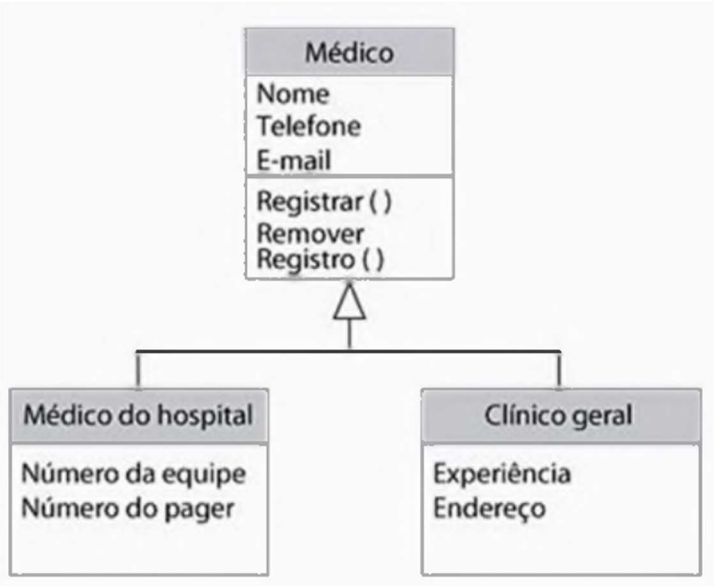
Figura 7 - Representação da Associação entre Classes



Fonte: Pressman (2011)

Booch *et al.* (1998) define o terceiro tipo de relacionamento, generalização, como uma relação classes classes-mãe e suas classes-filha, no caso as classes-filha são derivações da classe-mãe, herdando suas propriedades. Esse relacionamento é representado por uma linha com uma grande seta triangular.

Figura 8 - Representação da Generalização entre Classes



Fonte: Sommerville (2011)

O diagrama de classes em UML torna-se necessário para uma visualização do software, proporcionando uma melhor perspectiva para analisar a estrutura do software, segundo Guedes (2018), o diagrama de classes expõe uma visão estática da disposição das classes, focando com a estrutura lógica delas.

3.1.2. Desvio Padrão

Lunet *et al.* (2006), conceituam desvio padrão como uma medida de dispersão, em que seu valor final representa a variabilidade das verificações em relação à média. Essa grandeza pode ser utilizada quando o objetivo da análise visa descrever a variabilidade de uma amostra.

Segundo Feijoo (2010), essa medida de dispersão tem como finalidade representar a heterogeneidade de um conjunto. Logo, se faz necessário para a análise de conjuntos de valores e identificar quais conjuntos contém uma maior variação em seus valores. Esse conceito se faz necessário para avaliar a quantidade de interações entre as classes de um sistema e em seguida apontar quais classes tem uma quantidade heterogênea de interações dentro do sistema.

A fórmula matemática para a obtenção dessa medida é composta pelos seguintes elementos: Desvio Padrão (DP), somatória (∑), conjunto de valores (xi), média aritmética dos valores (MA) e quantidade de valores (n), seu resultado é representado pela letra grega σ. A representação fórmula pode ser visualizada abaixo.

Figura 9 - Fórmula do Desvio Padrão

$$DP = \sqrt{\frac{\sum_{i=1}^{n}(x_i - M_A)^2}{n}}$$

Fonte: Elaborada pelo autor (2019)

1. Primeiramente, é calculada a média aritmética do conjunto de valores (MA), somando todos os valores do conjunto (xi) e dividindo pela quantidade de valores (n);
2. Após obter a média aritmética, será tirada a diferença de cada valor do conjunto (xi) pela MA;

3. Cada valor da diferença será elevado ao quadrado;
4. Ao obter os valores das potencializações, todos os valores serão somados (∑);
5. O resultado da somatória então será dividido pela quantidade de valores (n);
6. E por último, será calculada a raiz quadrada do valor resultante da divisão;
7. Obtendo o valor final do desvio padrão (DP).

Para exemplificar a utilização do desvio padrão, será proposto o uso dessa medida para analisar dois grupos de valores, esses que estão expostos na figura 10.

Figura 10 - Conjuntos de Valores

4	4,2	4,5	4,7	4,8	4,9	5	5,1	5,5	5,6	6,1
1	2	2,5	4	4,5	5,5	6	6,4	7	7,5	8

Fonte: Martins, M. (2013)

Os dois conjuntos contêm valores diferentes, porém a média dos dois conjuntos é 4,9, porém o valor de seu desvio padrão é diferente. O cálculo para obter o valor do desvio padrão dos dois conjuntos é representado nas Figuras 11 e 12.

Figura 11 - Cálculo do Desvio Padrão do Primeiro Conjunto

$$DP_1 = \sqrt{\frac{\sum_{i=1}^{n}(x_i - M_A)^2}{n}}$$

$$DP_1 = \frac{\sqrt{\sum_{i=1}^{11}(x_1 - 4{,}9)^2 + (x_2 - 4{,}9)^2 + \cdots + (x_{11} - 4{,}9)^2}}{11}$$

$$DP_1 = \frac{\sqrt{\sum_{i=1}^{11}(-0{,}9)^2 + (-0{,}7)^2 + (-0{,}4)^2 + (-0{,}2)^2 + (-0{,}1)^2 + (0)^2 + (0{,}1)^2 + (0{,}2)^2 + (0{,}6)^2 + (0{,}7)^2 + (1{,}2)^2}}{11}$$

$$DP_1 = \frac{\sqrt{0{,}9 + 0{,}7 + 0{,}4 + 0{,}2 + 0{,}1 + 0 + 0{,}1 + 0{,}2 + 0{,}6 + 0{,}7 + 1{,}2}}{11}$$

$$DP_1 = \frac{\sqrt{5{,}1}}{11} = \sqrt{0{,}46}$$

$$DP_1 = 0{,}6\ \sigma$$

Fonte: Elaborada pelo autor (2019)

Após realizar o cálculo baseado nos valores do primeiro conjunto de valores, foi obtido desvio padrão de valor 6 σ. Em seguida obtêm-se o valor do desvio padrão do segundo conjunto.

Figura 12 - Cálculo do Desvio Padrão do Segundo Conjunto

$$DP_2 = \sqrt{\frac{\sum_{i=1}^{n}(x_i - M_A)^2}{n}}$$

$$DP_2 = \frac{\sqrt{\sum_{i=1}^{11}(x_1 - 4{,}9)^2 + (x_2 - 4{,}9)^2 + \cdots + (x_{11} - 4{,}9)^2}}{11}$$

$$DP_2 = \frac{\sqrt{\sum_{i=1}^{11}(-3{,}9)^2 + (-2{,}9)^2 + (-2{,}4)^2 + (-0{,}9)^2 + (-0{,}4)^2 + (0{,}6)^2 + (1{,}1)^2 + (1{,}5)^2 + (2{,}1)^2 + (2{,}6)^2 + (3{,}1)^2}}{11}$$

$$DP_2 = \frac{\sqrt{15{,}21 + 8{,}41 + 5{,}76 + 0{,}81 + 0{,}16 + 0{,}36 + 1{,}21 + 2{,}25 + 4{,}41 + 6{,}76 + 9{,}61}}{11}$$

$$DP_2 = \frac{\sqrt{54{,}95}}{11} = \sqrt{4{,}99}$$

$$DP_2 = 2{,}23\,\sigma$$

Fonte: Elaborada pelo autor (2019)

O valor do desvio padrão do segundo conjunto de valores foi 2,23 σ, o valor do desvio padrão do segundo conjunto é maior que o do primeiro conjunto, logo contata-se que os valores do segundo conjunto têm disparidade maior entre seus valores, tal que ela pode ser percebida ao analisar a Figura 13, que exibe de maneira gráfica os dois conjuntos.

Figura 13 - Gráfico dos Dois Conjuntos de Valores

Fonte: Martins, M. (2013)

Analisando a Figura 13 constata-se a diferença de dispersão dos valores dentro de cada conjunto. Assim como Feijoo (2010) relata, existe a correlação entre a aumento do desvio padrão um conjunto e a dispersão dos valores dentro do conjunto.

3.2. A Metodologia Proposta

3.2.1. Seleção dos Projetos

Primeiramente devem ser localizados códigos-fonte da linguagem Java, esse critério se tem devido à grande popularidade da linguagem, sendo umas das três principais linguagens de programação (TIOBE, 2021), a pesquisa preferiu abordar apenas essa linguagem para execução da metodologia, porém não rejeita a possibilidade de abordar a metodologia em outras linguagens de programação orientadas a objeto. E por último, estes devem pertencer ao mesmo domínio, tendo em vista sistemas de um mesmo domínio regularmente partilham similaridades estruturais (SOMMERVILLE, 2011), logo há a possibilidade de obter segmentos análogos.

3.2.2. Engenharia Reversa

Após obter os códigos-fonte, deve ser utilizado um Ambiente de Desenvolvimento Integrado ou *Integrated Development Environment* (IDE) para utilizar a ferramenta de visualização dos diagramas. O IDE empregado nesta pesquisa foi o Eclipse IDE for Java, dado que contém uma ferramenta disponível para integração chamada *UML Lab*. Esta ferramenta possibilita realizar a engenharia reversa da composição do sistema gerando a representação gráfica do código-fonte por meio de um diagrama de classes em UML, exibindo todas as classes do sistema e suas relações.

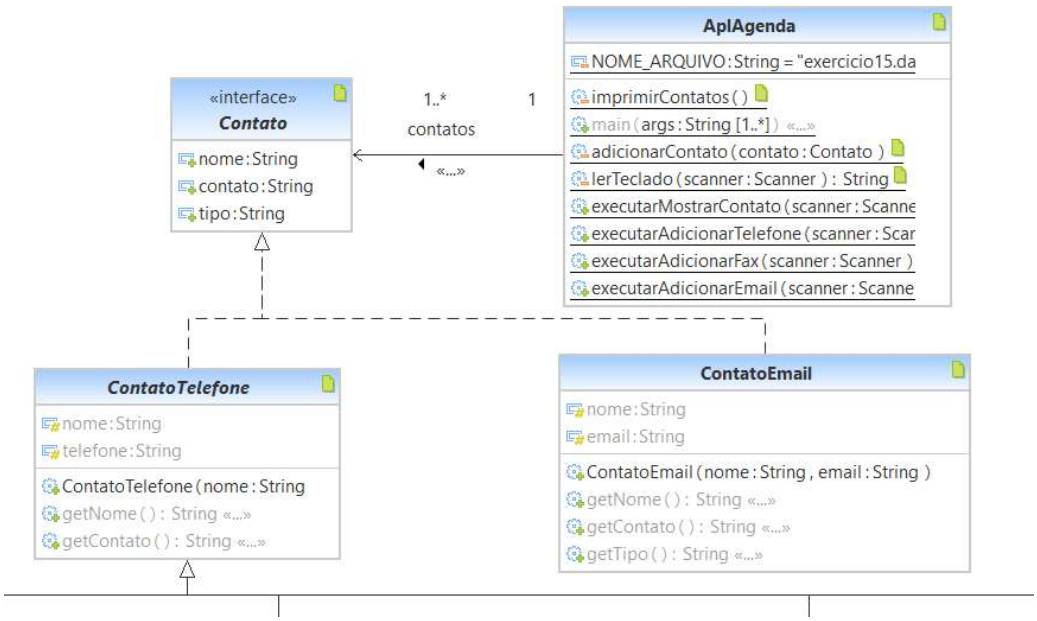
O diagrama de classes gerado pela ferramenta caracteriza a estrutura do código-fonte de maneira gráfica, apresentando os elementos de um sistema. As classes são retratadas por retângulos, constituídos pelo nome da classe, seus métodos e suas variáveis. As relações entre as classes são retratadas por setas que interligam as classes, compondo um mapa do sistema retratado.

Posteriormente a geração do diagrama, é possível visualizar o sistema por completo, logo, deve-se quantificar, listar e classificar as relações das classes do sistema. Cada classe pode ter ligações com outras classes, essas ligações que podem ser classificadas como associação ou dependência.

Com a geração dos diagramas de classe baseados no código-fonte do software, é possível obter um panorama geral da estruturação deste sistema. Segundo Booch *et al.* (2005), compara os diagramas UML a modelos de projeto arquitetônico de uma casa, sendo necessário para ter uma visão geral do projeto que será executado. Trazendo em si os planos detalhados, e gerais, para assim fornecer uma perspectiva ampla do sistema, exibindo assim uma amostra do como o sistema pretender ser ou é.

Dotado destas informações, logo a geração de diagramas UML de um sistema fornece assim uma visão completa do projeto, possibilitando uma engenharia reversa da estrutura do software. Para isso há a necessidade de primeiramente explorar a composição do diagrama de classe UML.

Figura 14 - Estrutura de Classes



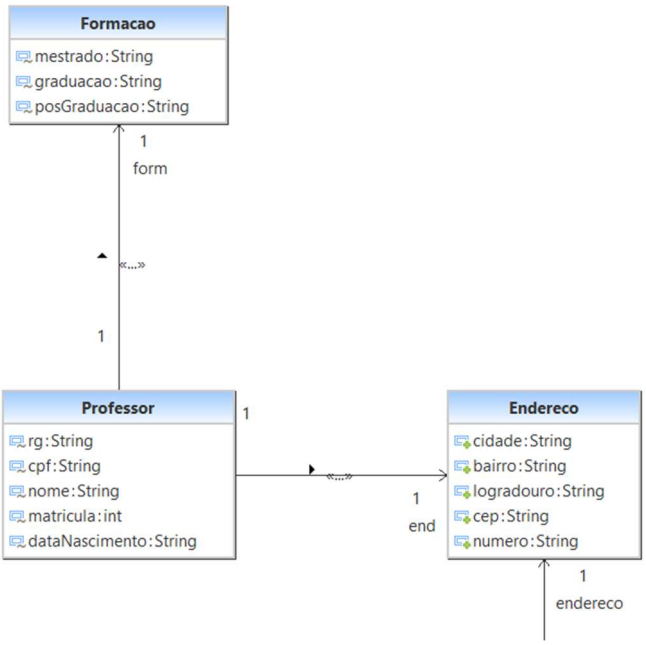
Fonte: Elaborada pelo autor (2021)

Como explica Costa (2001), as classes de diagrama são como a representação estrutural de um conjunto de atributos, métodos e relações com outras classes. E estes relacionamentos definem a maneira que os objetos do sistema serão implementados, o autor os classifica em associação, generalização e agregação. Essas relações podem representa apenas uma associação entre classes ou uma herança de atributos.

3.2.3. Classificação das Interações da UML

Durante sua associação, uma classe pode solicitar ou enviar informações para outra(s) classe(s), estas interações serão classificadas da seguinte forma: Associação por Solicitação (AS) ou Associação por Envio (AE), esse tipo de relação é representado por setas que vão de uma classe à outra.

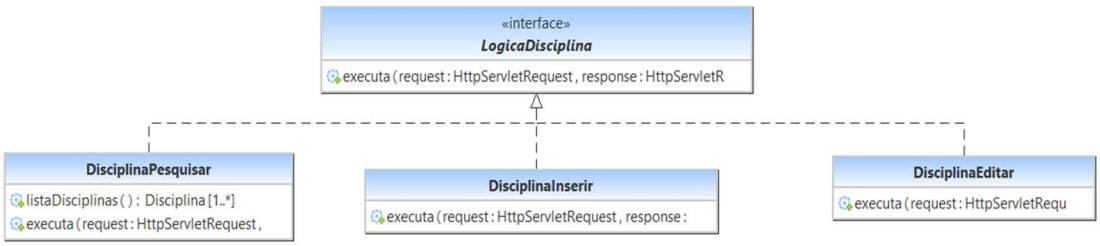
Figura 15 - Associação entre Classes



Fonte: Elaborada pelo autor (2019)

De mesma forma, uma classe pode ter uma Dependência de duas maneiras: Quando uma classe depende de outra(s), Dependência sendo Dependente (DD) ou quando outra(s) classe(s) depende(m) dela, Dependência sendo Alimentando (DA), estas relações são representadas por setas tracejadas ligando classes.

Figura 16 - Dependência entre Classes



Fonte: Elaborada pelo autor (2019)

Levando em consideração essa classificação, é possível quantificar cada tipo de relação, os valores devem ser armazenados para que possam ser utilizados quando o cálculo do desvio for aplicado. O armazenamento das informações de um sistema deve classificar as informações

categorizando em: Nome da classe, Associação por Solicitação, Associação por Envio, Dependência sendo Dependente e Dependência sendo Alimentando.

3.2.4. Contabilização de Interações

Após realizar a identificação a classificação das interações entre as classes do sistema, o quarto passo a ser realizado é a contabilização das interações identificadas. Sendo necessários assim criar um quadro para a catalogação, e neste segue a quantidade de classes, a identificação de cada classe, a quantidade de associações por solicitação (AS), quantidade de solicitações por envio (AE), quantidade de dependências sendo dependentes (DD) e de dependências sendo alimentada (DA). Como pode ser visto na Tabela 2 abaixo:

Tabela 2 - Exemplo de tabela de contabilização de interações

Nº	CLASSES	AS	AE	DD	DA
1	Principal	0	0	0	17
2	Curso	4	0	1	0
3	Matriz	0	0	4	1
4	opcaoinvalidaException	0	0	3	0
5	Pesquisador	1	0	3	0
6	Funcionario	3	1	2	0
7	Pessoa	0	0	2	0
8	Teste	0	0	0	2
9	AlunoPosGraduacao	0	2	1	0
10	Menu	0	0	2	1
11	Tecnico	0	1	2	0
12	Disciplina	1	2	1	0
13	Professor	2	1	1	0
14	ProjetoPesquisa	1	2	1	0
15	Aluno	2	2	1	1
16	BaseDadosAlunos	0	1	1	1
17	BaseDadosCursos	0	1	1	1
18	BaseDadosDisciplina	0	1	1	1
19	BaseDadosProjetos	0	1	1	1
20	BaseDadosTecnicos	0	1	1	1

Tabela 2 - Exemplo de tabela de contabilização de interações

(Continuação)

Nº	CLASSES	AS	AE	DD	DA
21	FolhaPagamento	0	0	0	1
22	JanelaSimples	0	0	1	0
23	TabelaMenu	0	0	0	0

Fonte: Elaborada pelo autor (2021)

3.2.5. Aplicação do Desvio Padrão

O quinto passo a ser realizado é a aplicação do desvio padrão, a aplicação se dá pelo cálculo utilizando quantidade de interações de associações e dependências de cada classe. Para isso a tabela exemplo precisa ser consultada, buscando a primeira classe listada: “Principal”. Em seguida, os valores de cada tipo de interação devem ser abstraídos, neste caso temos as seguintes interações nesta classe: 0 associações por solicitação, 0 associações por envio, 0 dependências sendo dependente, e 17 dependências alimentando.

Neste momento os valores abstraídos devem ser aplicados na fórmula do desvio padrão. Primeiramente será tirada a média dos quatro valores (M_A), esta que totalizará o valor 4,25. Em seguida cada valor das quatro interações será subtraído pela média aritmética do conjunto. Após a subtração, o valor restante de cada operação será elevado ao quadrado, obtendo então quatro potências, estas que serão somadas e divididas.

Por último, será tirada a raiz quadrada do resultado da divisão, esta que resultará no valor do desvio padrão do conjunto de interações da classe “Principal”, o desvio padrão do conjunto é 7,36, como pode ser observado na Figura 17, que exemplifica cada passo da obtenção do desvio padrão.

Figura 17 - Exemplo de Aplicação do Desvio Padrão

$$DP = \sqrt{\frac{\sum_{i=1}^{n}(x_i - M_A)^2}{n}}$$

$$DP = \sqrt{\frac{\sum_{i=1}^{4}(0 - 4{,}25)^2 + (0 - 4{,}25)^2 + (0 - 4{,}25)^2 + (17 - 4{,}25)^2}{4}}$$

$$DP = \sqrt{\frac{\sum_{i=1}^{4}(-4{,}25)^2 + (-4{,}25)^2 + (-4{,}25)^2 + (12{,}75)^2}{4}}$$

$$DP = \sqrt{\frac{18{,}0625 + 18{,}0625 + 18{,}0625 + 162{,}5625}{4}}$$

$$DP = \sqrt{\frac{216{,}75}{4}} = \sqrt{54{,}1875}$$

$$DP = 7{,}36\sigma$$

Fonte: Elaborada pelo autor (2021)

O cálculo realizado para auferir o desvio padrão da classe “Principal” deve ser repetido nas demais classes da tabela. Ao final do processo será obtido os valores de todas as 23 classes do sistema exemplo, resultando em uma nova tabela com os valores do desvio padrão (DP) abaixo.

Tabela 3 - Exemplo de tabela com as interações e o desvio padrão

Nº	CLASSES	AS	AE	DD	DA	DP
1	Principal	0	0	0	17	7,36
2	Curso	4	0	1	0	1,64
3	Matriz	0	0	4	1	1,64
4	opcaoinvalidaException	0	0	3	0	1,30
5	Pesquisador	1	0	3	0	1,22
6	Funcionario	3	1	2	0	1,12
7	Pessoa	0	0	2	0	0,87
8	Teste	0	0	0	2	0,87
9	AlunoPosGraduacao	0	2	1	0	0,83
10	Menu	0	0	2	1	0,83
11	Tecnico	0	1	2	0	0,83

Tabela 3 - Exemplo de tabela com as interações e o desvio padrão

(Continuação)

Nº	CLASSES	AS	AE	DD	DA	DP
12	Disciplina	1	2	1	0	0,71
13	Professor	2	1	1	0	0,71
14	ProjetoPesquisa	1	2	1	0	0,71
15	Aluno	2	2	1	1	0,50
16	BaseDadosAlunos	0	1	1	1	0,43
17	BaseDadosCursos	0	1	1	1	0,43
18	BaseDadosDisciplina	0	1	1	1	0,43
19	BaseDadosProjetos	0	1	1	1	0,43
20	BaseDadosTecnicos	0	1	1	1	0,43
21	FolhaPagamento	0	0	0	1	0,43
22	JanelaSimples	0	0	1	0	0,43
23	TabelaMenu	0	0	0	0	0,00

Fonte: Elaborada pelo autor (2021)

Como observado na Tabela 3, essa contém a coluna Desvio Padrão (DP) em que são exibidos os valores obtidos do desvio padrão das interações de cada classe do sistema analisado. Contendo estes valores podemos passar para a próxima etapa, a seleção das classes, esta será responsável pela triagem das classes com mais interações.

3.2.6. Seleção das Classes Acima da Média Geral

A última etapa a ser executada nesta metodologia trata-se da seleção das classes, que acontece após o cálculo do desvio padrão das interações das classes. A partir dos valores obtidos no desvio, torna-se possível quantificar a variação de interações de uma classe, agora a próxima etapa será taxar uma média dessas variações para então parametrizar quais foram as classes que mais variaram.

Para contemplar estes dados será efetuado o cálculo da média aritmética, sendo assim todos os valores da coluna DP serão somados e em seguidas divididos pelo total de classes. A execução do processo descrito neste parágrafo pode ser visualizada na Figura 18, representação da abaixo.

Figura 18 - Cálculo da Média Aritmética dos Desvios Padrão

$$M_A = \frac{x_1 + x_2 + x_3 \ldots + x_n}{n}$$

$$M_A = \frac{7{,}36 + 1{,}64 + 1{,}64 + 1{,}30 + 1{,}22 + 1{,}12 + \cdots + 0{,}00}{23}$$

$$M_A = \frac{24{,}15}{23}$$

$$M_A = 1{,}05$$

Fonte: Elaborada pelo autor (2021)

O valor atingido com a média aritmética dos desvios padrões foi de 1,05, portanto esse valor será empregado como referência, para realização de uma taxação entre os valores do desvio padrão, selecionando as classes com o valor de desvio superior que a média geral obtida entre estas.

Tabela 4 - Tabela de Classes com Desvio Padrão Acima da Média

Nº	CLASSES	DP
1	Principal	7,36
2	Curso	1,64
3	Matriz	1,64
4	opcaoinvalidaException	1,30
5	Pesquisador	1,22
6	Funcionario	1,12

Fonte: Elaborada pelo autor (2021)

Como mostrado acima na Tabela 4, selecionando apenas as classes com o desvio padrão igual ou acima da média foram obtidas seis classes. Finalizado o processo, pode-se julgar que as seis classes selecionadas são potenciais componentes reutilizáveis considerando que todas contêm uma quantidade acima da comum de interações dentro do conjunto de classes do sistema selecionado. Logo, a quantidade superior de interações indica que estas classes têm relevância para a realização dos processos executados no software.

3.3. Considerações Finais

Este capítulo descreveu as duas bases fundamentais para o desenvolvimento da metodologia de identificação, o diagrama de classes em UML e o desvio padrão, abordando seus conceitos, etapas e funcionamento.

Foi abordada a introdução à linguagem UML e sua funcionalidade na arquitetura de um software, utilizando especificamente o diagrama de classes para analisar as interações entre as classes de um software. Em seguida, foi apresentado a medida do desvio padrão, necessária para o exame da dispersão de valores dentro de um conjunto, suas representações e etapas, assim como uma demonstração prática de seu uso.

Por último foi apresentado a aplicação prática dos conceitos de análise de diagrama UML, cálculo do desvio padrão utilizando os valores das interações das classes de um sistema exemplo. Assim como o tratamento dos dados obtidos para avaliação de potenciais componentes reutilizáveis.

A metodologia de identificação de componentes tem como objetivo auxiliar a Engenharia de Softwares Baseada em Componentes propondo um processo de identificação de potenciais componentes dentro de um sistema analisando as interações entre as suas classes.

4 AVALIAÇÃO DA METODOLOGIA

Neste capítulo será abordada a execução da sistemática proposta nesta pesquisa, experimentando os procedimentos apresentados em três sistemas selecionados de domínio em comum, por conseguinte, outra metodologia será aplicada nos mesmos sistemas. Ao final, serão analisados os resultados, para comparação avaliativa dos seus desempenhos na obtenção de potenciais componentes reutilizáveis.

4.1. Avaliação da Metodologia Baseada em Interações Entre Classes

Para analisar o desempenho da metodologia proposta neste trabalho, esta será aplicada durante este capítulo em três sistemas. Permanecendo os mesmos critérios da seleção utilizados para o exemplo da seção 3.3.1, foram selecionados três projetos armazenados em uma plataforma de hospedagem de código-fonte, seguindo dois critérios: serem projetos da mesma linguagem de programação e domínio.

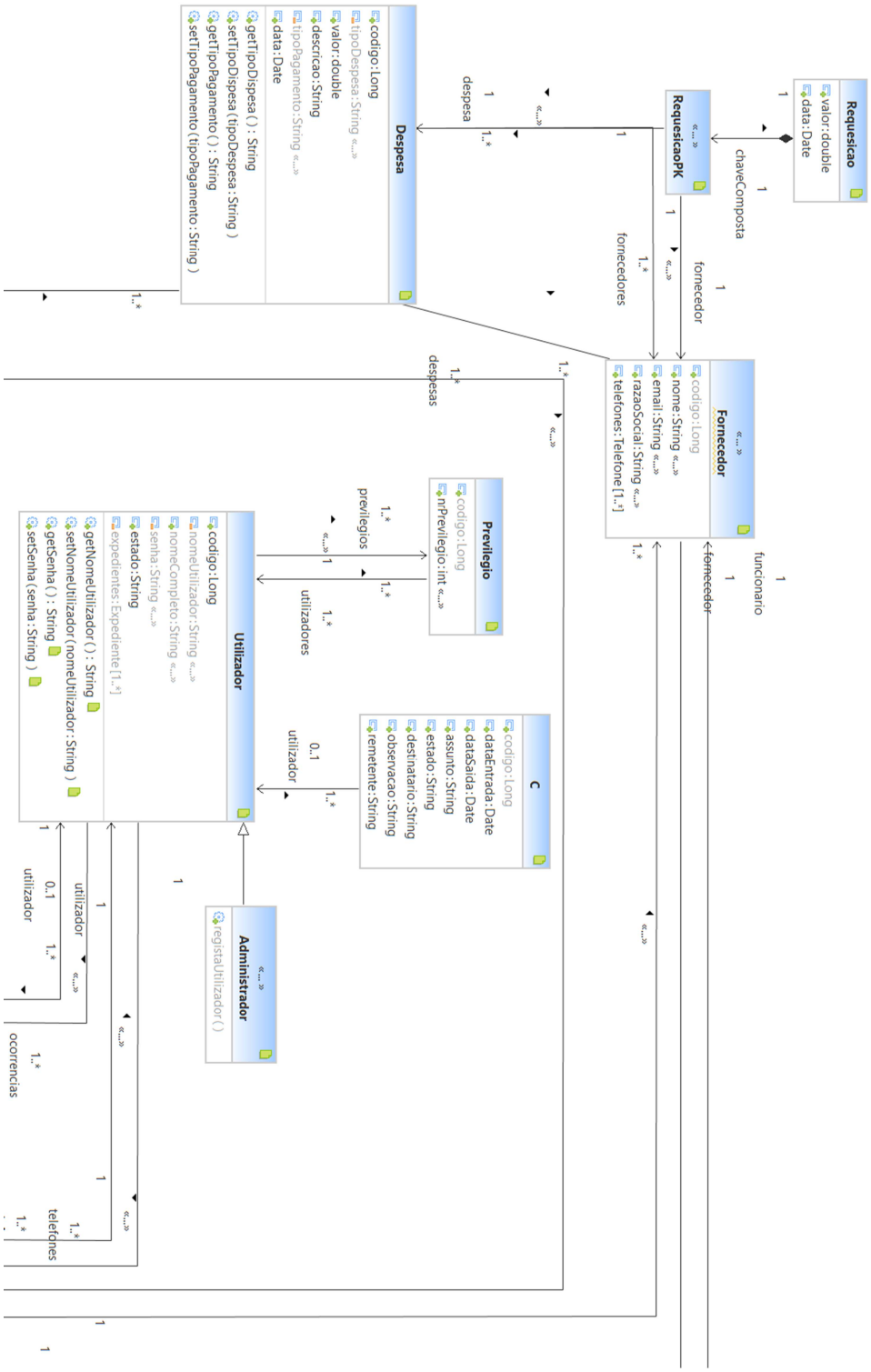
O domínio escolhido para esta pesquisa foi sistemas de gestão escolar, o critério utilizado para a escolha foi que geralmente tratam-se de sistemas de médio porte, que contém uma quantidade de classes variante entre 20 e 100, possibilitando uma análise menos limitada. Empregando o segundo critério pré-estabelecido, os sistemas precisam ser desenvolvidos na linguagem Java.

Seguindo os parâmetros citados anteriormente, os três projetos selecionados foram: (I) sistemagestaoescolar (https://github.com/benyAlves/sistemagestaoescolar), (II) GestaoEscolar (https://github.com/gumiranda/GestaoEscolar) e (III) kaderneta (https://github.com/kenethy/kaderneta). Todos estes estão disponíveis na plataforma de hospedagem: https://github.com/.

4.1.1. Aplicação da Metodologia no Sistema sistemagestaoescolar

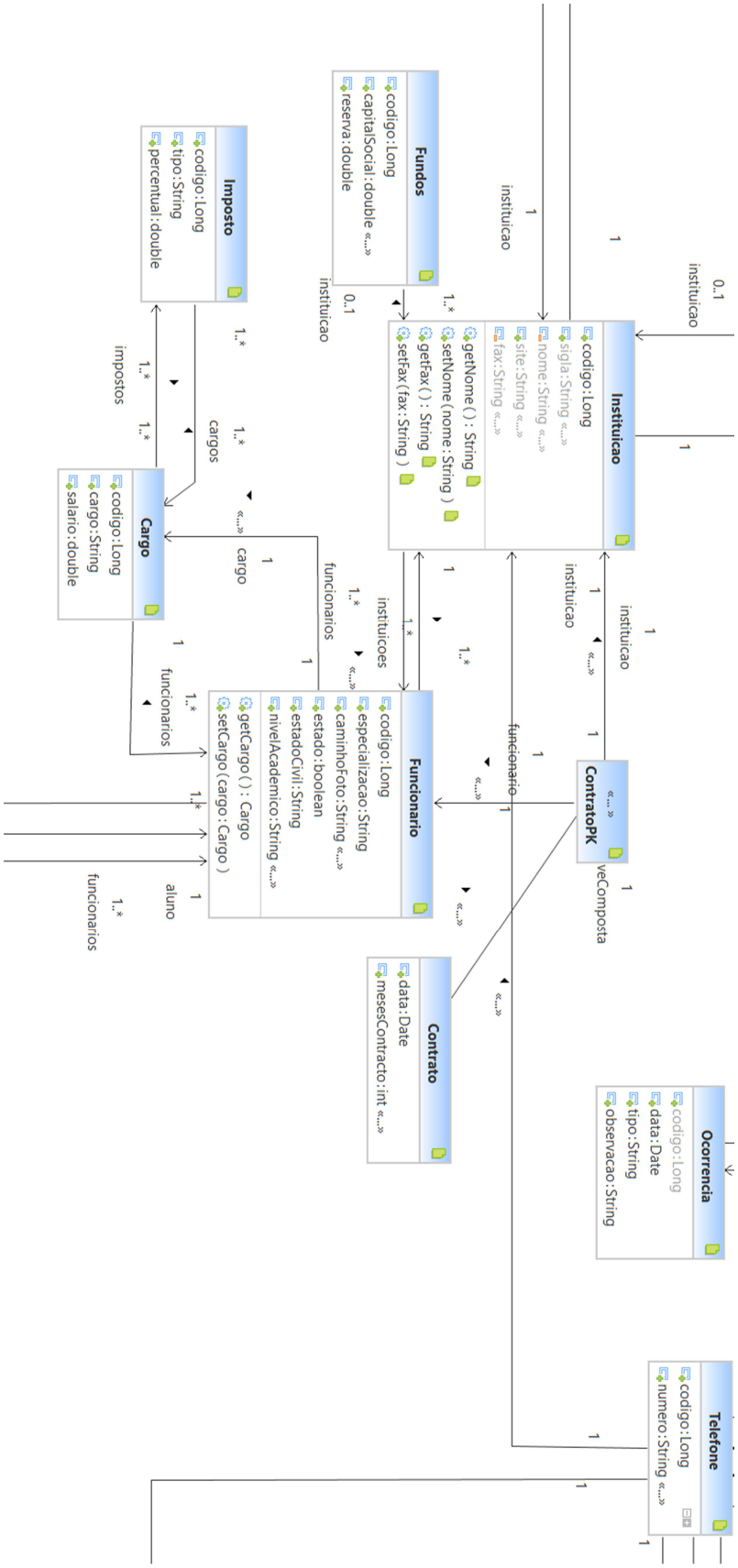
O primeiro projeto a ser aplicada a metodologia foi o sistemagestaoescolar, inicialmente foi feita a engenharia reversa do código convertendo para diagrama de classe para analisar as interações do projeto. O diagrama do sistemagestaoescolar foi gerado pela ferramenta UML *Lab* e pode ser visualizada abaixo nas Figuras 19 a 24.

Figura 19 - Diagrama de Classes do Sistema - sistemagestaoescolar (Parte 1)



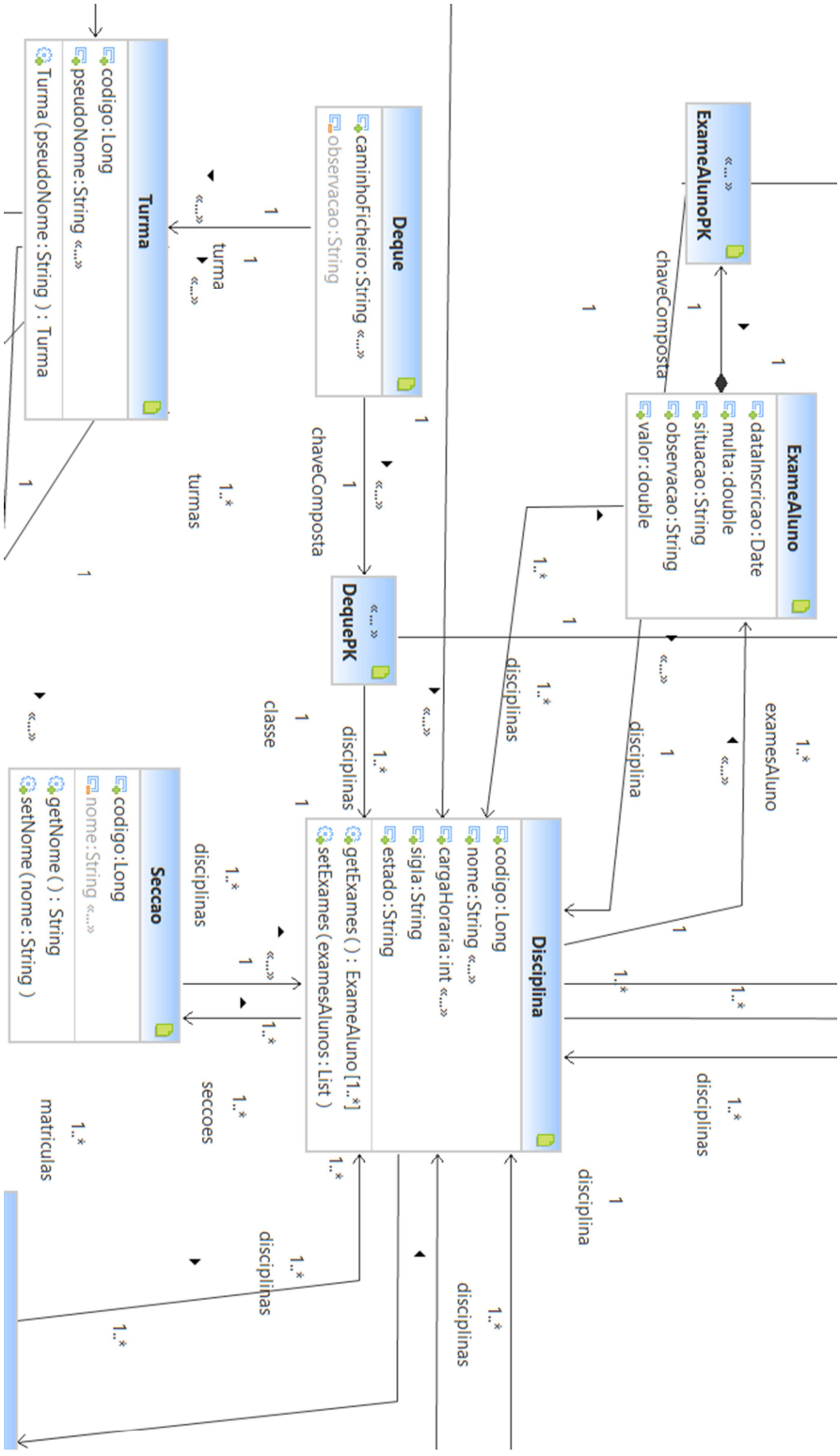
Fonte: Elaborada pelo autor (2021)

Figura 20 - Diagrama de Classes do Sistema - sistemagestaoescolar (Parte 2)



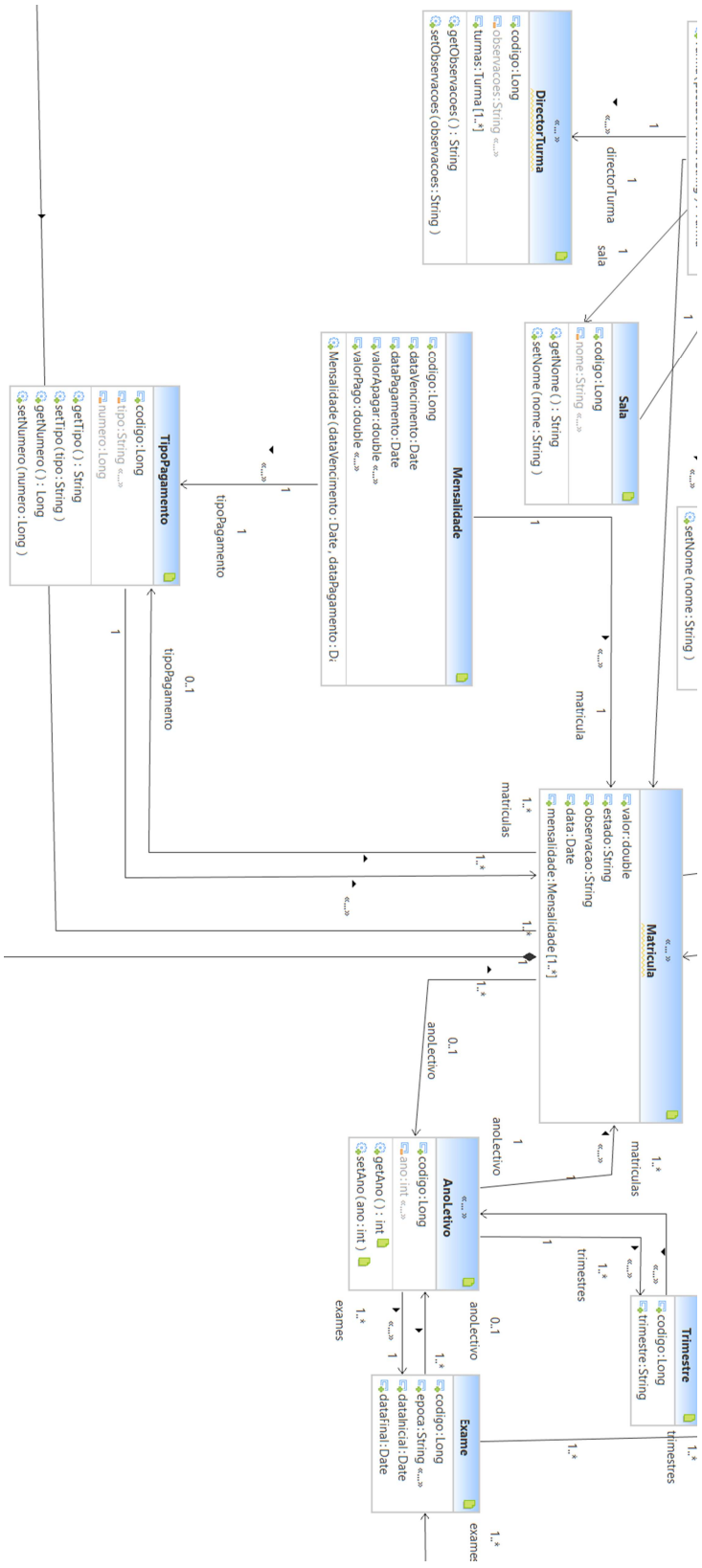
Fonte: Elaborada pelo autor (2021)

Figura 21 - Diagrama de Classes do Sistema - sistemagestaoescolar (Parte 3)



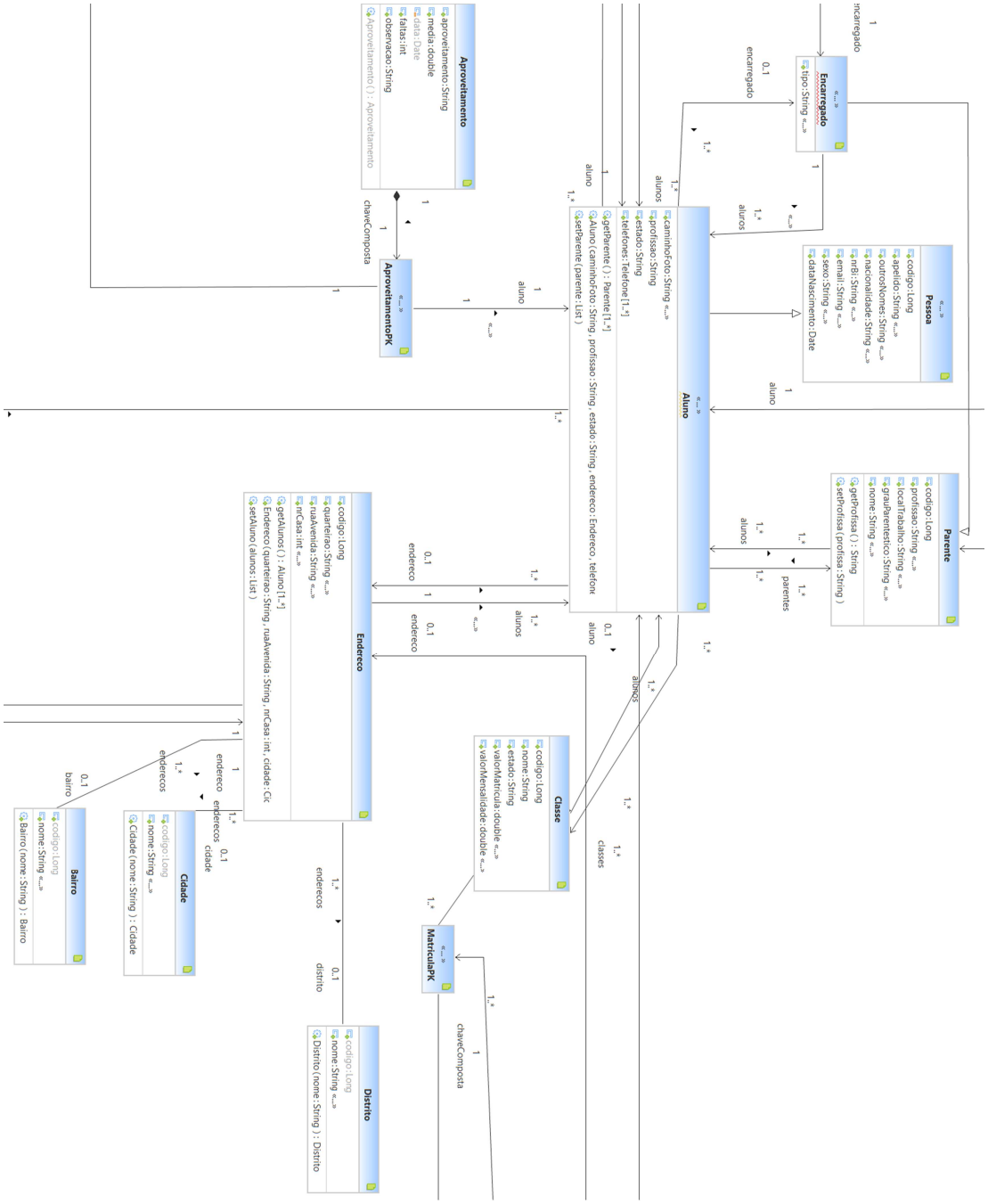
Fonte: Elaborada pelo autor (2021)

Figura 22 - Diagrama de Classes do Sistema - sistemagestaoescolar (Parte 4)



Fonte: Elaborada pelo autor (2021)

Figura 23 - Diagrama de Classes do Sistema - sistemagestaoescolar (Parte 5)



Fonte: Elaborada pelo autor (2021)

Figura 24 - Diagrama de Classes do Sistema - sistemagestaoescolar (Parte 6)

Fonte: Elaborada pelo autor (2021)

Após a análise do diagrama de classes foi criada uma tabela para contabilizar todos os relacionamentos das classes que compõem Sistema sistemagestaoescolar, então essas interações foram devidamente classificadas (AS, AE, DD e DA) e o valor do desvio padrão (DP) obtido pelo total de interações de cada classe.

Tabela 5 - Tabela de interações e desvio padrão do Sistema - sistemagestaoescolar

SISTEMA - sistemagestaoescolar						
Nº	**CLASSES**	**AS**	**AE**	**DD**	**DA**	**DESVIO**
1	Administrador	1	0	0	0	0,433
2	Aluno	7	9	0	0	4,062
3	AnoLetivo	3	3	0	0	1,500
4	Aproveitamento	1	0	0	0	0,433
5	AproveitamentoPK	2	1	0	0	0,829
6	Bairro	0	1	0	0	0,433
7	Cargo	2	2	0	0	1,000
8	Cidade	0	1	0	0	0,433
9	Classe	1	2	0	0	0,829
10	Contrato	1	0	0	0	0,433
11	ContratoPK	2	1	0	0	0,829
12	Deque	2	0	0	0	0,866
13	DequePK	2	1	0	0	0,829
14	Despesa	2	2	0	0	1,000
15	DirectorTurma	0	1	0	0	0,433
16	Disciplina	4	9	0	0	3,700
17	Distrito	0	1	0	0	0,433
18	Encarregado	2	2	0	0	1,000
19	Endereco	5	3	0	0	2,121
20	Exame	2	2	0	0	1,000
21	ExameAluno	2	1	0	0	0,829
22	ExameAlunoPK	2	1	0	0	0,829
23	Expediente	1	1	0	0	0,500
24	Fornecedor	2	4	0	0	1,658
25	Funcionario	2	5	0	0	2,046
26	Fundos	1	0	0	0	0,433
27	Imposto	1	1	0	0	0,500
28	Instituicao	2	5	0	0	2,046
29	Matricula	5	5	0	0	2,500
30	MatriculaPK	2	1	0	0	0,829
31	Mensalidade	2	0	0	0	0,866
32	Ocorrencia	1	1	0	0	0,500

Tabela 5 - Tabela de interações e desvio padrão do Sistema - sistemagestaoescolar

(Continuação)

SISTEMA - sistemagestaoescolar						
Nº	**CLASSES**	**AS**	**AE**	**DD**	**DA**	**DESVIO**
33	Parente	1	3	0	0	1,225
34	Pessoa	0	1	0	0	0,433
35	Privilegio	1	1	0	0	0,500
36	Requisicao	1	0	0	0	0,433
37	RequisicaoPK	2	1	0	0	0,829
38	Sala	1	1	0	0	0,500
39	Seccao	2	0	0	0	0,866
40	Telefone	7	2	0	0	2,861
41	TipoPagamento	1	2	0	0	0,829
42	Trimestre	2	1	0	0	0,829
43	Turma	3	3	0	0	1,500
44	Utilizador	3	3	0	0	1,500
45	AdministradorDao	1	0	0	0	0,433
46	AlunoDao	1	0	0	0	0,433
47	AnoLetivoDao	1	0	0	0	0,433
48	AproveitamentoDao	1	0	0	0	0,433
49	BairroDao	1	0	0	0	0,433
50	CargoDao	1	0	0	0	0,433
51	CidadeDao	1	0	0	0	0,433
52	ClasseDao	1	0	0	0	0,433
53	ContratoDao	1	0	0	0	0,433
54	DequeDao	1	0	0	0	0,433
55	DespesaDao	1	0	0	0	0,433
56	DirectorTurmaDao	1	0	0	0	0,433
57	DisciplinaDao	1	0	0	0	0,433
58	DistritoDao	1	0	0	0	0,433
59	EncarregadoDao	1	0	0	0	0,433
60	EnderecoDao	1	0	0	0	0,433
61	ExameDao	1	0	0	0	0,433
62	ExameDisciplinaDao	1	0	0	0	0,433
63	FornecedorDao	1	0	0	0	0,433
64	FuncionarioDao	1	0	0	0	0,433
65	FundosDao	1	0	0	0	0,433
66	GenericDao	1	34	0	0	14,584
67	ImpostoDao	1	0	0	0	0,433
68	InstituicaoDao	1	0	0	0	0,433
69	MatriculaDao	1	0	0	0	0,433
70	MensalidadeDao	1	0	0	0	0,433

Tabela 5 - Tabela de interações e desvio padrão do Sistema - sistemagestaoescolar

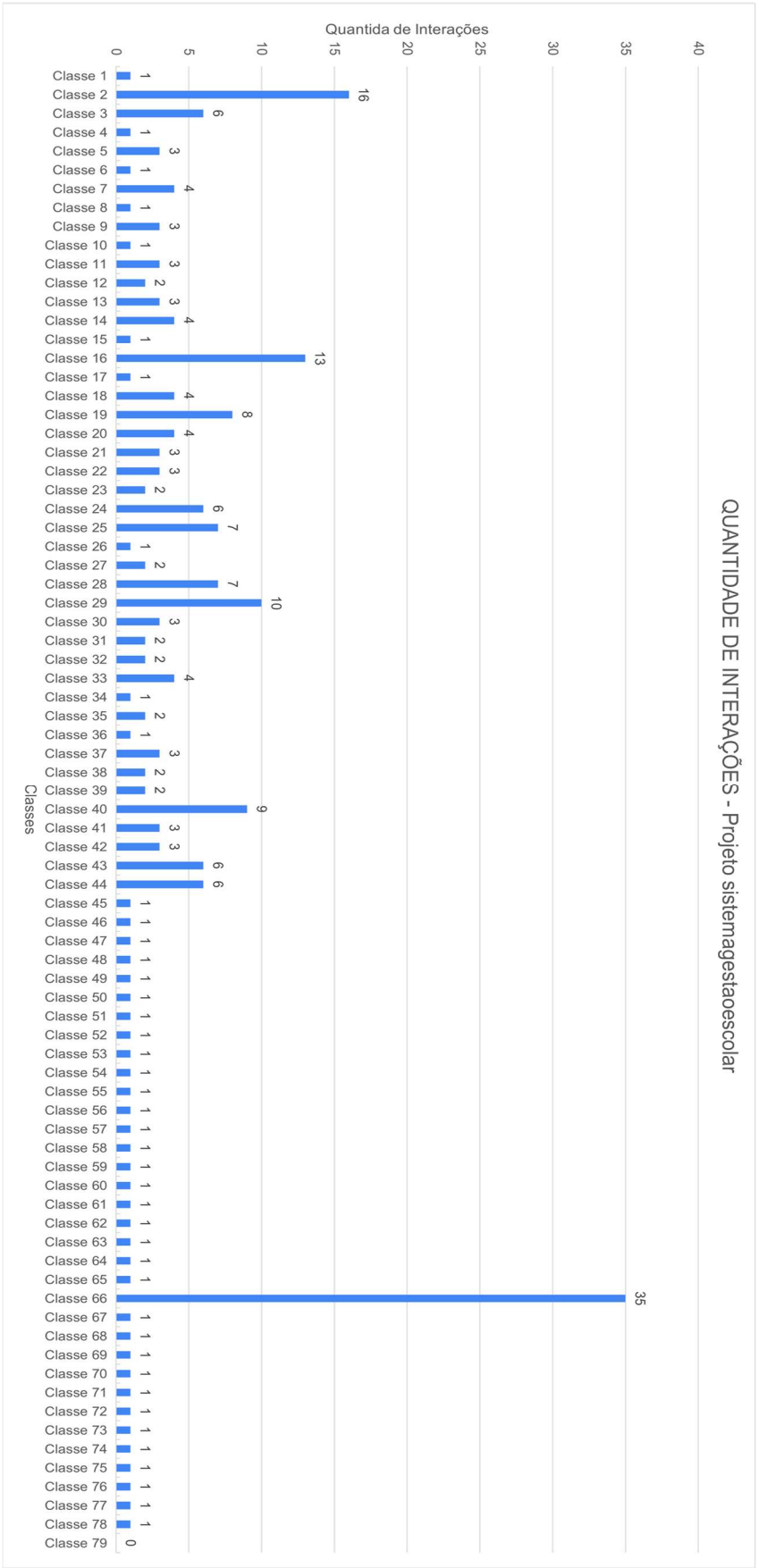
(Conclusão)

SISTEMA - sistemagestaoescolar						
Nº	**CLASSES**	**AS**	**AE**	**DD**	**DA**	**DESVIO**
71	ParenteDao	1	0	0	0	0,433
72	RequisicaoDao	1	0	0	0	0,433
73	SalaDao	1	0	0	0	0,433
74	SeccaoDao	1	0	0	0	0,433
75	TelefoneDao	1	0	0	0	0,433
76	TipoPagamentoDao	1	0	0	0	0,433
77	TurmaDao	1	0	0	0	0,433
78	UtilizadorDao	1	0	0	0	0,433
79	SistemaEscolar	0	0	0	0	0,000

Fonte: Elaborada pelo autor (2021)

Para retratar a quantidade de interações das classes, a seguir foram adicionados exibindo os valores totais de interações. Na tabela já descrita cada classe contém um número de identificação na primeira coluna, este número que será utilizado como representante da classe no gráfico abaixo.

Figura 25 - Quantidade de Interações do Sistema - sistemagestaoescolar

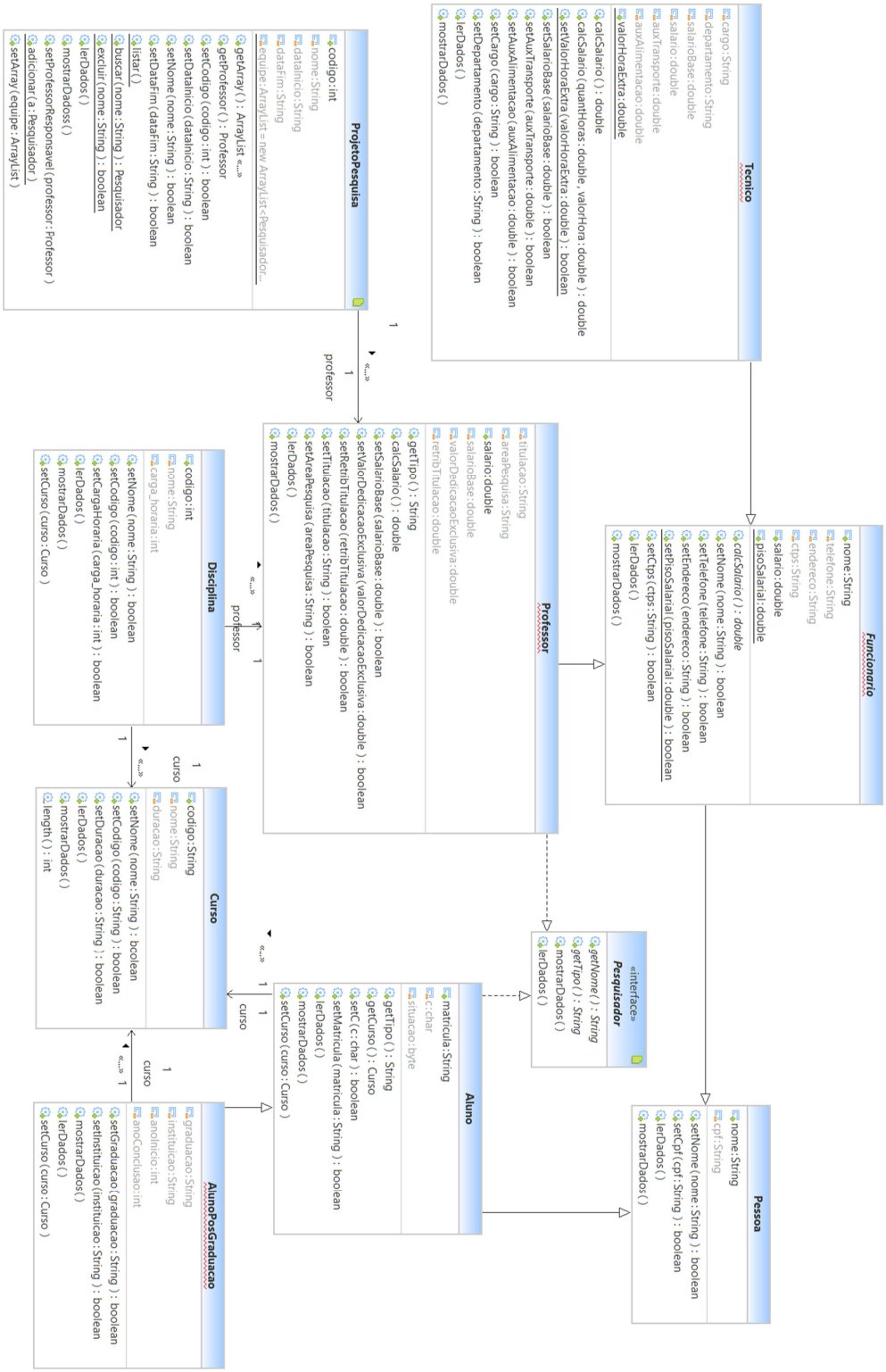


Fonte: Elaborada pelo autor (2021)

4.1.2. Aplicação da Metodologia no Sistema GestaoEscolar

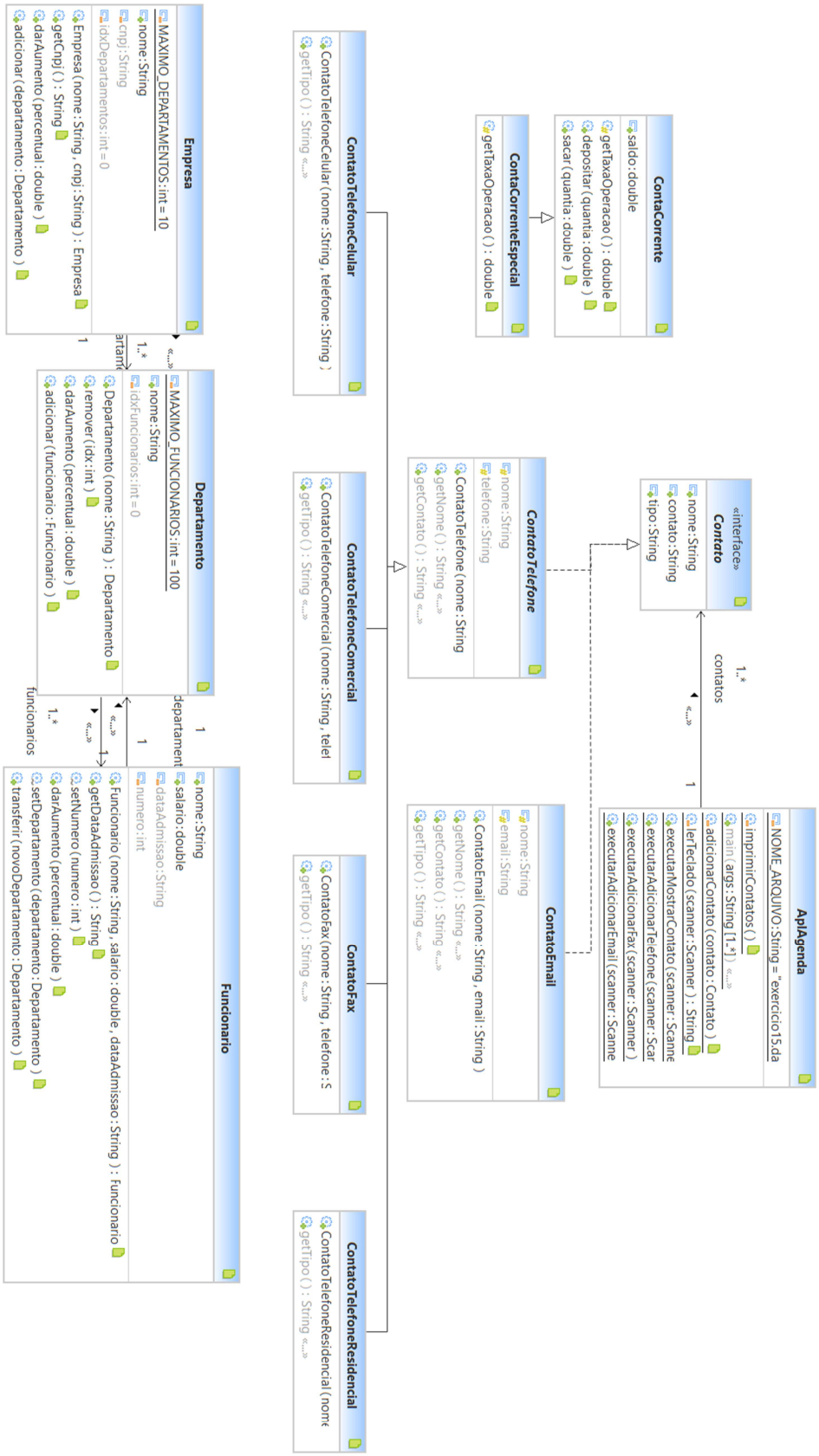
Em seguida, o segundo projeto aplicado GestaoEscolar, após a engenharia reversa do código, gera o diagrama para análise, representado em duas partes, as Figuras 26, 27 e 28.

Figura 26 - Diagrama de Classes do Sistema - GestaoEscolar (Parte 1)



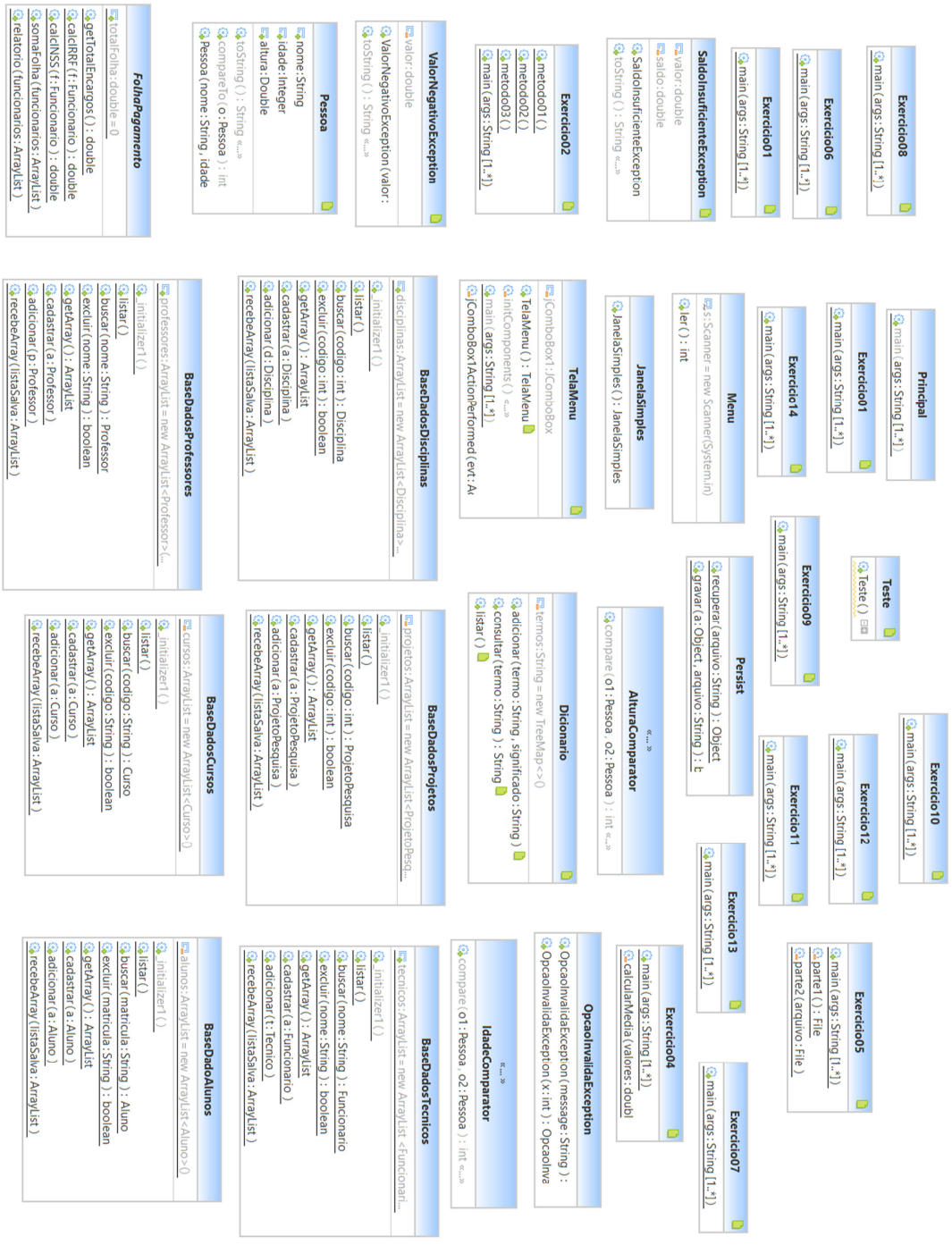
Fonte: Elaborada pelo autor (2021)

Figura 27 - Diagrama de Classes do Sistema - GestaoEscolar (Parte 2)



Fonte: Elaborada pelo autor (2021)

Figura 28 - Diagrama de Classes do Sistema - GestaoEscolar (Parte 3)



Fonte: Elaborada pelo autor (2021)

A Tabela 6 representa a contabilização das interações e o desvio padrão das classes do Sistema GestaoEscolar. Este que tem uma variação do desvio padrão menor que o projeto anterior, devido ter classes com menor quantidade de interações. Analisando a tabela abaixo, também se nota a presença de classes com dependências sendo dependente (DD) e também sendo alimentando (DA).

Tabela 6 - Tabela de interações e desvio padrão do Sistema - GestaoEscolar

SISTEMA - GestaoEscolar						
Nº	**CLASSES**	**AS**	**AE**	**DD**	**DA**	**DESVIO**
1	Aluno	2	1	1	0	0,71
2	AlunoPosGraduacao	2	0	0	0	0,87
3	BaseDadosAlunos	0	0	0	0	0,00
4	BaseDadosCursos	0	0	0	0	0,00
5	BaseDadosDisciplinas	0	0	0	0	0,00
6	BaseDadosProfessores	0	0	0	0	0,00
7	BaseDadosProjetos	0	0	0	0	0,00
8	BaseDadosTecnicos	0	0	0	0	0,00
9	Curso	0	3	0	0	1,30
10	Disciplina	2	0	0	0	0,87
11	FolhaPagamento	0	0	0	0	0,00
12	Funcionario	1	1	0	0	0,50
13	JanelaSimples	0	0	0	0	0,00
14	Menu	0	0	0	0	0,00
15	OpcaoInvalidaException	0	0	0	0	0,00
16	Persist	0	0	0	0	0,00
17	Pesquisador	0	0	0	2	0,87
18	Pessoa	0	2	0	0	0,87
19	Principal	0	0	0	0	0,00
20	Professor	1	2	1	0	0,71
21	ProjetoPesquisa	1	0	0	0	0,43
22	Tecnico	1	0	0	0	0,43
23	TelaMenu	0	0	0	0	0,00
24	Teste	0	0	0	0	0,00
25	ContaCorrente	0	1	0	0	0,43
26	ContaCorrenteEspecial	1	0	0	0	0,43
27	Exercicio01	0	0	0	0	0,00
28	SaldoInsuficienteException	0	0	0	0	0,00
29	ValorNegativoException	0	0	0	0	0,00
30	Departamento	1	2	0	0	0,83
31	Empresa	1	0	0	0	0,43
32	Exercicio05	0	0	0	0	0,00
33	Funcionario	1	2	0	0	0,83
34	Dicionario	0	0	0	0	0,00
35	Exercicio13	0	0	0	0	0,00
36	Exercicio14	0	0	0	0	0,00
37	Pessoa	0	0	0	0	0,00
38	ApIAgenda	1	0	0	0	0,43
39	Contato	0	1	0	1	0,50

Tabela 6 - Tabela de interações e desvio padrão do Sistema - GestaoEscolar

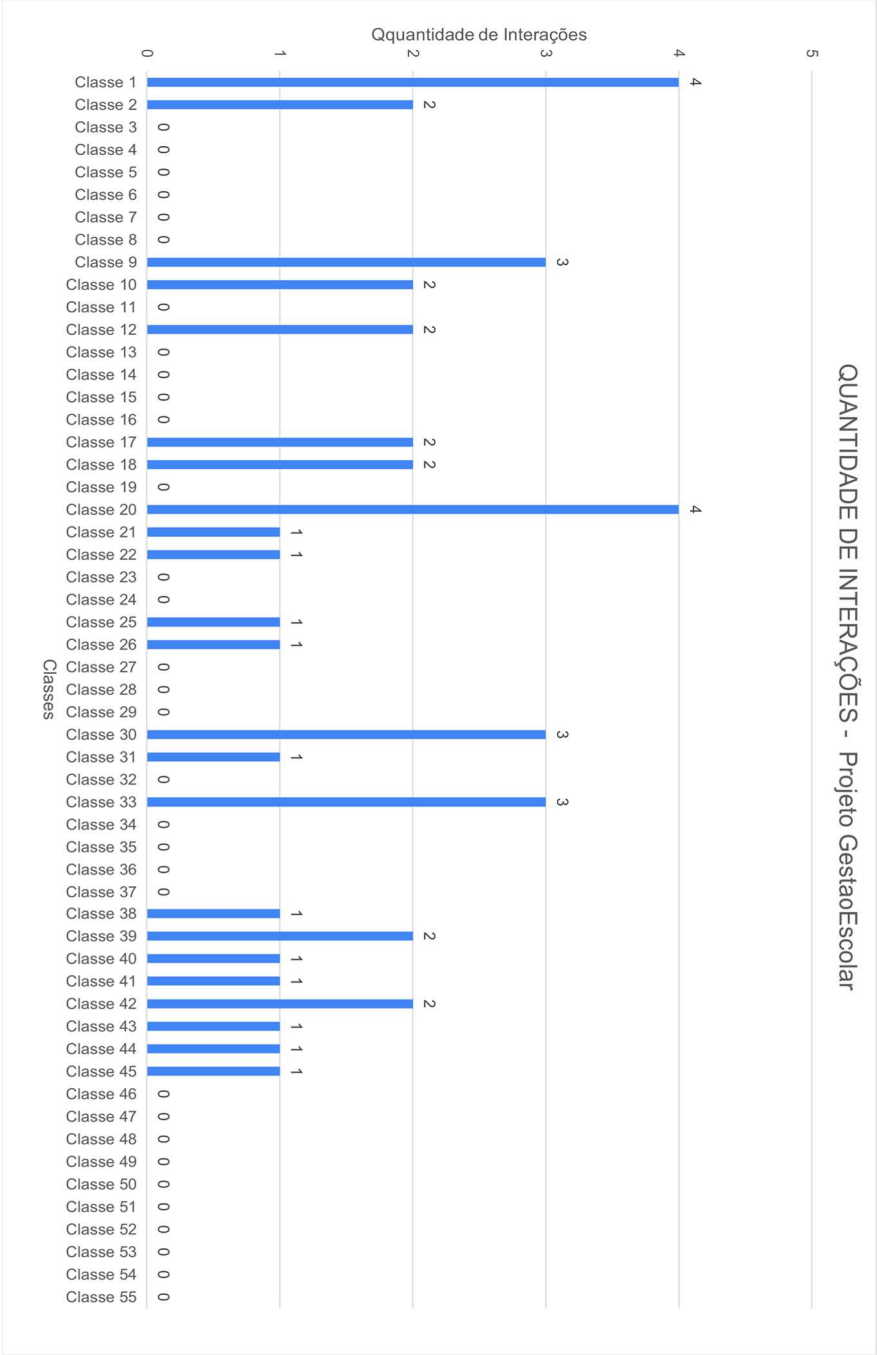
(Continuação)

SISTEMA - GestaoEscolar						
Nº	**CLASSES**	**AS**	**AE**	**DD**	**DA**	**DESVIO**
40	ContatoEmail	0	0	1	0	0,43
41	ContatoFax	1	0	0	0	0,43
42	ContatoTelefone	0	1	1	0	0,50
43	ContatoTelefoneCelular	1	0	0	0	0,43
44	ContatoTelefoneComercial	1	0	0	0	0,43
45	ContatoTelefoneResidencial	1	0	0	0	0,43
46	Exercicio01	0	0	0	0	0,00
47	Exercicio02	0	0	0	0	0,00
48	Exercicio04	0	0	0	0	0,00
49	Exercicio06	0	0	0	0	0,00
50	Exercicio07	0	0	0	0	0,00
51	Exercicio08	0	0	0	0	0,00
52	Execicio09	0	0	0	0	0,00
53	Exercicio10	0	0	0	0	0,00
54	Exercicio11	0	0	0	0	0,00
55	Exercicio12	0	0	0	0	0,00

Fonte: Elaborada pelo autor (2021)

Através da Tabela 6 e o gráfico na Figura 29 é possível visualizar a variação das interações no projeto, que oscilou apenas entre 0 e no máximo 4 relacionamentos. Com uma grande quantidade de classes isoladas, sem conexões como representado na Figura 28.

Figura 29 - Quantidade de Interações do Sistema - GestaoEscolar



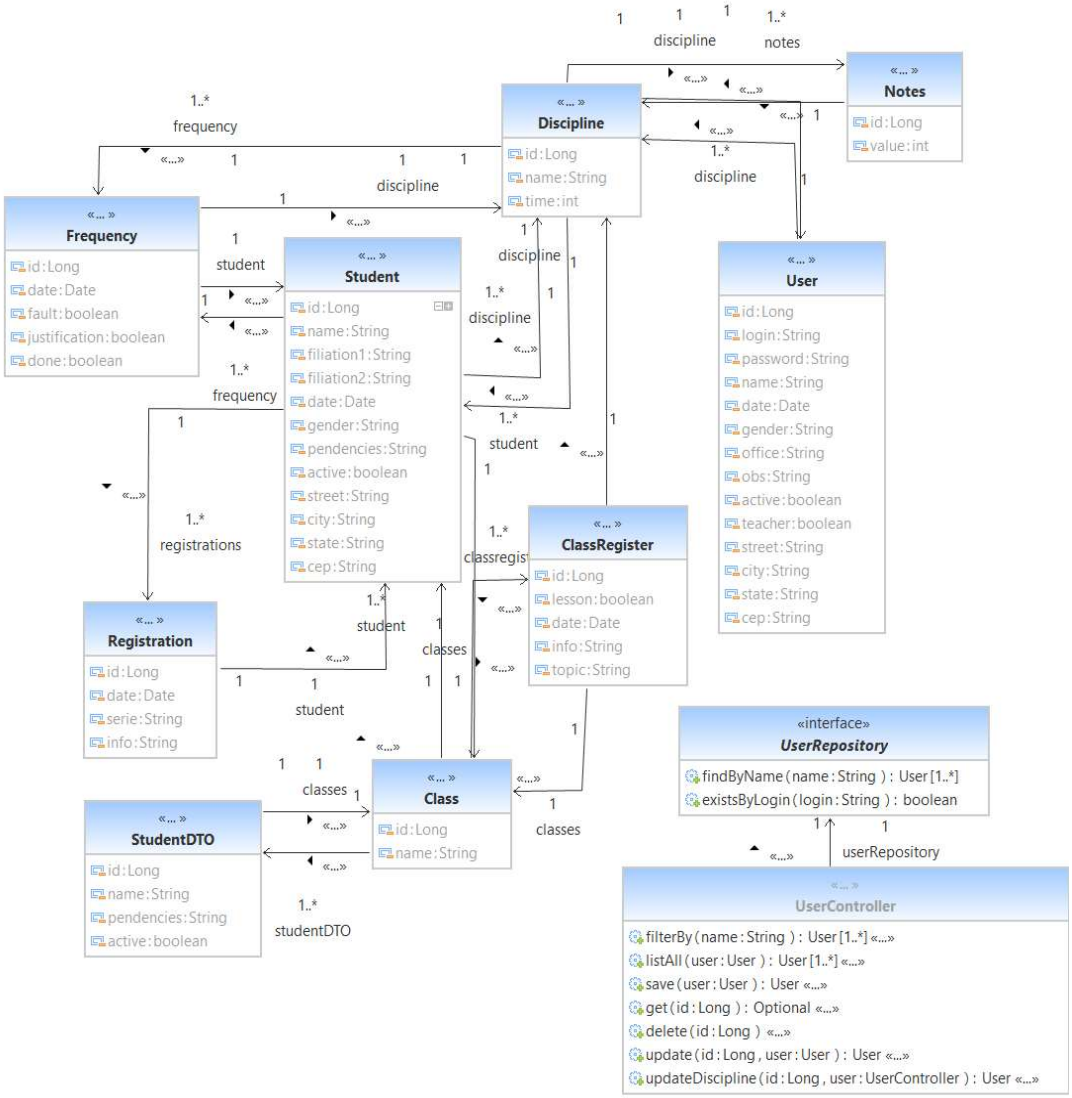
Fonte: Elaborada pelo autor (2021)

4.1.3. Aplicação da Metodologia no Sistema Kaderneta

Por último, o terceiro projeto Kaderneta, após gerado o diagrama de classes, fica visível na Figura 30 que praticamente todas as classes contêm pelo menos um relacionamento, sendo distinto do projeto anterior com várias classes isoladas.

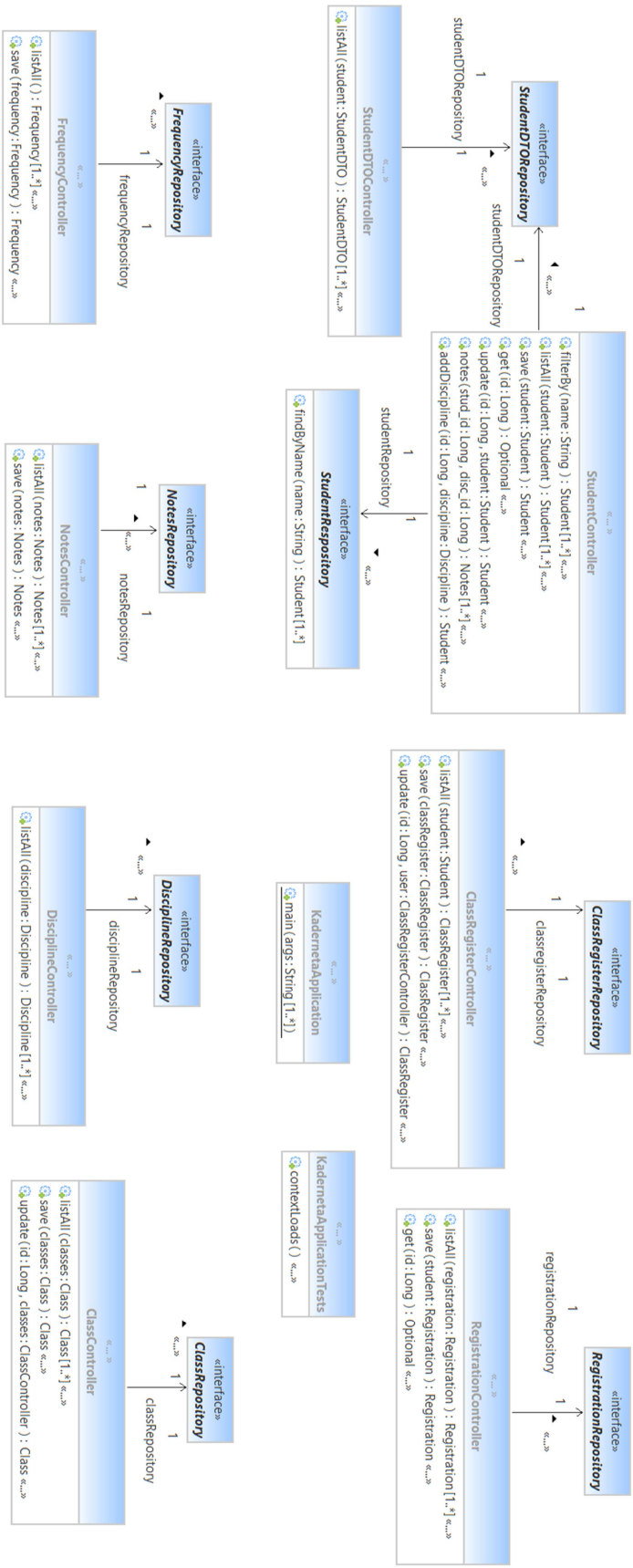
Outro ponto a ser ressaltado no Sistema Kaderneta, é o idioma utilizado nos nomes das classes, neste caso o inglês, este projeto destoa dos outros dois projetos que tem classes com nomes em português. Porém mesmo com a diferença de idioma a metodologia foi aplicada e obteve resultados.

Figura 30 - Diagrama de Classes do Sistema - Kaderneta (Parte 1)



Fonte: Elaborada pelo autor (2021)

Figura 31 - Diagrama de Classes do Sistema - Kaderneta (Parte 2)



Fonte: Elaborada pelo autor (2021)

Nota-se na Tabela 7 que o Sistema Kaderneta contabilizou apenas 29 classes, ao total, sendo este o projeto com a menor quantidade de classes em sua composição. Porém como descrito na tabela abaixo, o projeto contém uma quantidade maior de interações, em relação ao seu antecessor.

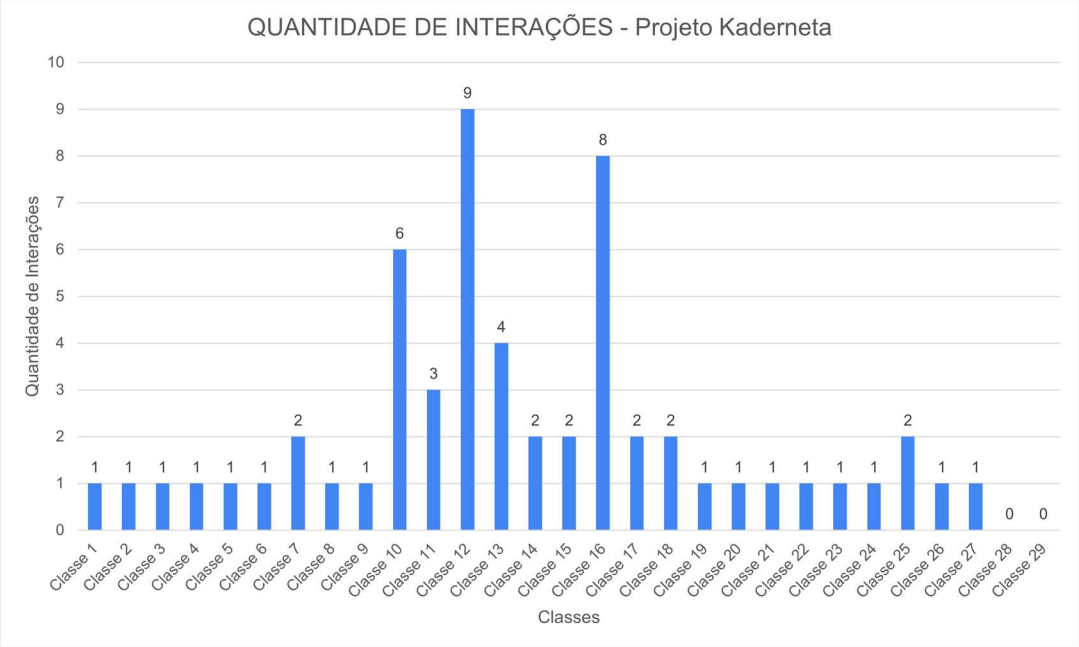
Tabela 7 - Tabela de interações e desvio padrão do Sistema - kaderneta

SISTEMA - kaderneta						
Nº	**CLASSES**	**AS**	**AE**	**DD**	**DA**	**DESVIO**
1	ClassController	1	0	0	0	0,43
2	ClassRegisterController	1	0	0	0	0,43
3	DisciplineController	1	0	0	0	0,43
4	FrequencyController	1	0	0	0	0,43
5	NotesController	1	0	0	0	0,43
6	RegistrationController	1	0	0	0	0,43
7	StudentController	0	2	0	0	0,87
8	StudentDTOController	1	0	0	0	0,43
9	UserController	1	0	0	0	0,43
10	Class	3	3	0	0	1,50
11	ClassRegister	2	1	0	0	0,83
12	Discipline	4	5	0	0	2,28
13	Frequency	2	2	0	0	1,00
14	Notes	1	1	0	0	0,50
15	Registration	1	1	0	0	0,50
16	Student	4	4	0	0	2,00
17	StudentDTO	1	1	0	0	0,50
18	User	1	1	0	0	0,50
19	ClassRegisterRepository	0	1	0	0	0,43
20	ClassRepository	0	1	0	0	0,43
21	DisciplineRepository	0	1	0	0	0,43
22	FrequencyRepository	0	1	0	0	0,43
23	NotesRepository	0	1	0	0	0,43
24	RegistrationRepository	0	1	0	0	0,43
25	StudentDTORespository	0	2	0	0	0,87
26	StudentRepository	0	1	0	0	0,43
27	UserRepository	0	1	0	0	0,43
28	KadernetaApplication	0	0	0	0	0,00
29	KadernetaApplication-Tests	0	0	0	0	0,00

Fonte: Elaborada pelo autor (2021)

Para finalizar a análise, abaixo na Figura 32 contém o gráfico com quantidade de relacionamento da Sistema Kaderneta.

Figura 32 - Quantidade de Interações do Sistema Kaderneta



Fonte: Elaborada pelo autor (2021)

Próxima etapa do processo será a realização da média dos valores de desvio padrão das classes e a seleção dos potenciais componentes reutilizáveis.

4.1.4. Resultados

A metodologia foi aplicada nos três projetos selecionados, o processo de execução se deu primeiramente pela engenharia reversa do código fonte de cada um dos projetos em seus diagramas de classe. Em seguida, são contabilizadas a quantidade de interações de cada uma das classes, essas são classificadas e listadas em uma tabela.

Com base na quantidade de relacionamentos, são calculados os valores de desvio padrão de todas as classes, também listados na tabela. Por conseguinte, calculada a média dos desvios, para então obter o valor de referência para seleção das potenciais classes para reutilização. Então as classes que tiverem o valor de desvio padrão acima da média obtida, são consideradas potenciais componentes reutilizáveis.

Posteriormente ao cálculo da média aritmética dos valores de desvio padrão das classes, foram obtidas as seguintes médias: 0,98 (Sistema sistemagestaoescolar), 0,26 (Sistema GestaoEscolar) e 0,63 (Sistema Kaderneta).

Esses valores serão utilizados como valor de corte para determinar as classes com potencial de reutilização. Abaixo estão representadas as tabelas com as classes com valor desvio acima da média calculada.

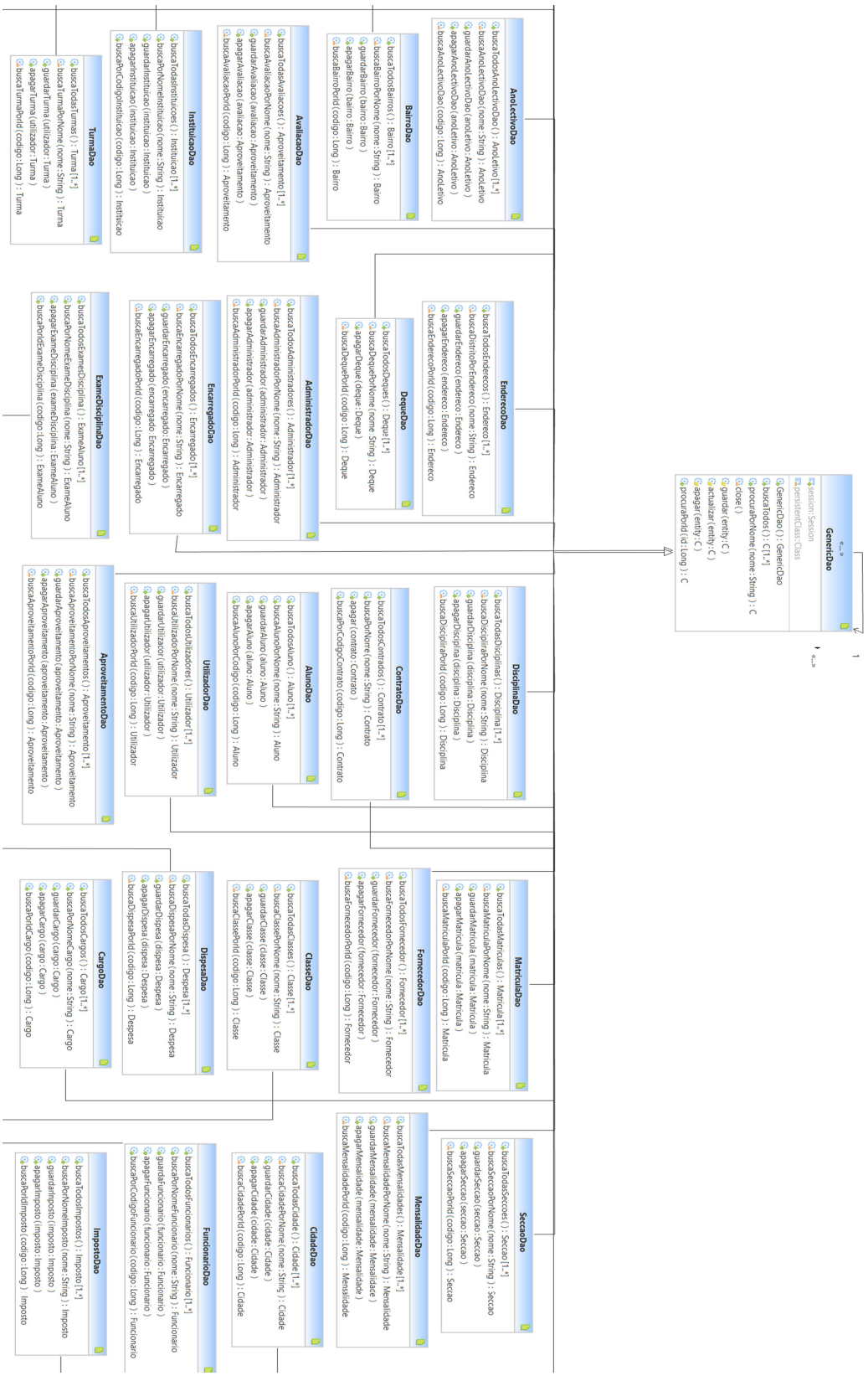
Tabela 8 - Tabela de potenciais componentes reutilizáveis do Sistema - sistemagestaoescolar

Nº	CLASSES	DP
1	GenericDao	14,58
2	Aluno	4,06
3	Disciplina	3,70
4	Telefone	2,86
5	Matricula	2,50
6	Endereco	2,12
7	Funcionario	2,05
8	Instituicao	1,66
9	Fornecedor	1,50
10	AnoLetivo	1,50
11	Turma	1,50
12	Utilizador	1,50
13	Parente	1,22
14	Cargo	1,00
15	Despesa	1,00
16	Encarregado	1,00
17	Exame	1,00

Fonte: Elaborada pelo autor (2021)

A classe “GenericDao” com maior desvio padrão dentre as outras do projeto sistemagestaoescolar, essa e suas interações podem ser contempladas na Figura 31. A principal peça do sistema contém relacionamentos com diversas outras classes, concentrando 34 interações entre as 79 classes.

Figura 33 – Interações da Classe “GenericDAO”



Fonte: Elaborada pelo autor (2021)

Tabela 9 - Tabela de potenciais componentes reutilizáveis do Sistema - GestaoEscolar

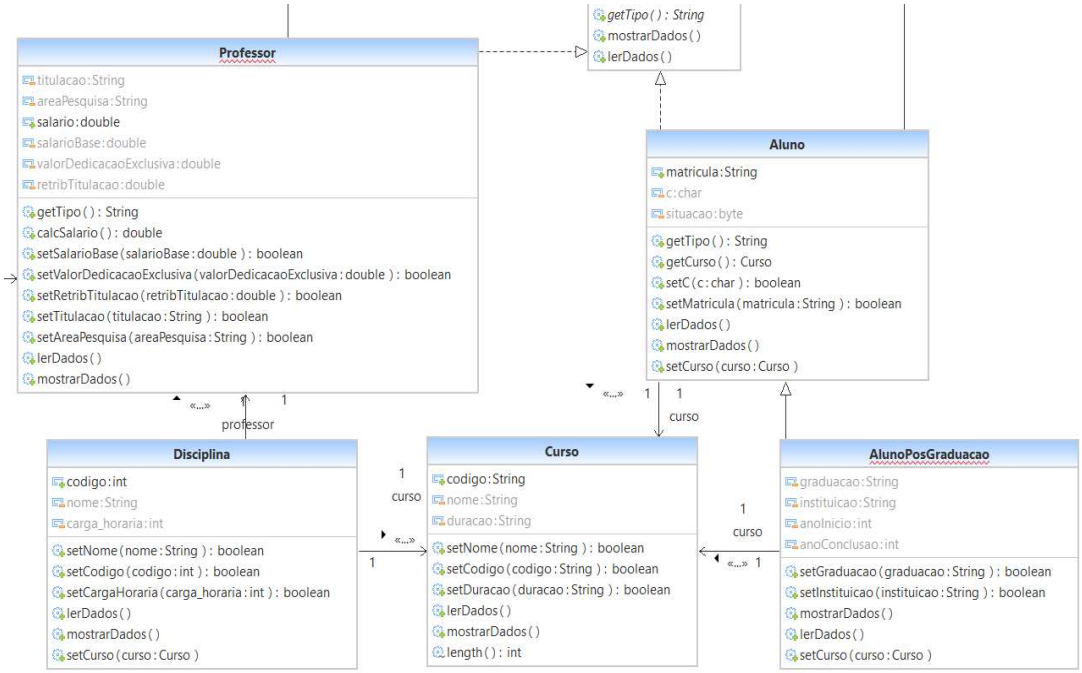
Nº	CLASSES	DP
1	Curso	1,30
2	AlunoPosGraduacao	0,87
3	Disciplina	0,87
4	Pesquisador	0,87
5	Pessoa	0,87
6	Departamento	0,83
7	Funcionario	0,83
8	Aluno	0,71
9	Professor	0,71
10	Funcionario	0,50
11	Contato	0,50
12	ContatoTelefone	0,43
13	ProjetoPesquisa	0,43
14	Tecnico	0,43
15	ContaCorrente	0,43
16	ContaCorrenteEspecial	0,43
17	Empresa	0,43
18	ApIAgenda	0,43
19	ContatoEmail	0,43
20	ContatoFax	0,43
21	ContatoTelefoneCelular	0,43
22	ContatoTelefoneComercial	0,43

Fonte: Elaborada pelo autor (2021)

A principal classe do projeto GestaoEscolar foi “Curso”, contendo o desvio padrão de valor 1,30. Esse projeto não contém valores de desvio padrão elevados em relação aos outros, sendo o sistema com os valores de desvio padrão mais baixo dentre os três projetos.

Abaixo, a Figura 34 apresenta os relacionamentos que a classe “Curso” possui, como mencionado sobre o baixo desvio padrão, percebe-se que a classe dispõe de apenas três relacionamentos.

Figura 34 – Interações da Classe "Curso"



Fonte: Elaborada pelo autor (2021)

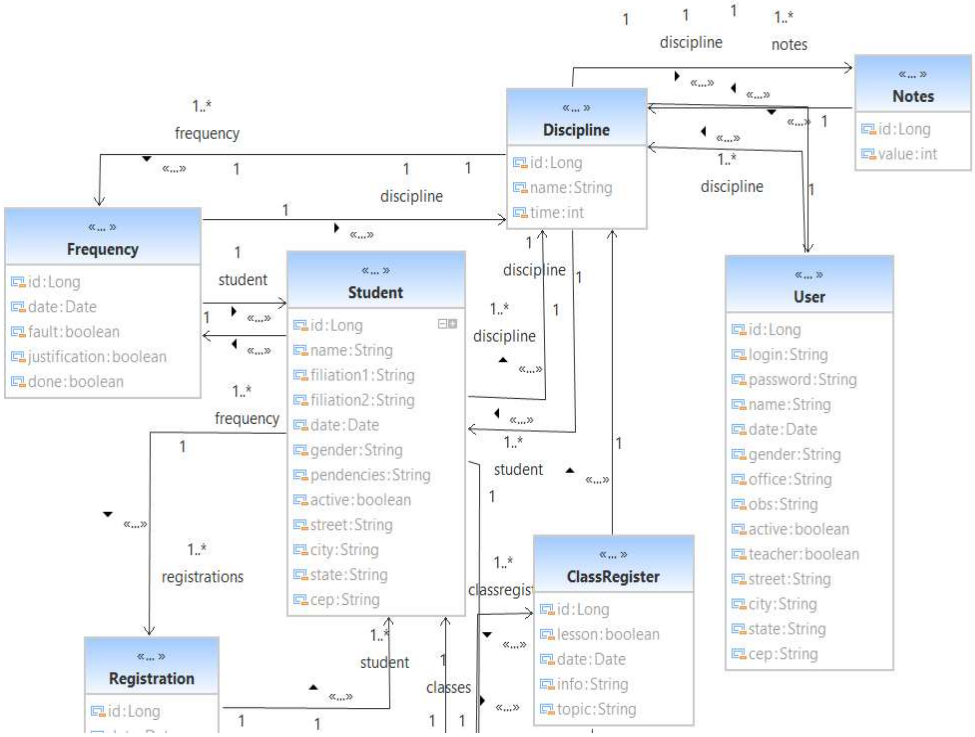
Finalizando os resultados obtidos na aplicação da metodologia de identificação de componentes reutilizáveis baseada nas interações entre classes, a apuração do projeto Kaderneta na Tabela 10.

Tabela 10 - Tabela de potenciais componentes reutilizáveis do Sistema – Kaderneta

Nº	CLASSES	DP
1	Discipline	2,28
2	Student	2,00
3	Class	1,50
4	Frequency	1,00
5	StudentController	0,87
6	StudentDTORespository	0,87
7	ClassRegister	0,83

Fonte: Elaborada pelo autor (2021)

Figura 35 - Interações da Classe "*Discipline*"



Fonte: Elaborada pelo autor (2021)

Analisando as tabelas, foi constatado que o projeto sistemagestaoescolar foram contêm dezessete classes com desvio acima da média 0,98, assim como o projeto GestaoEscolar conta com vinte e três classes com desvio acima de 0,26. E por último, o projeto Kaderneta possui sete classes com desvio acima da média 0,63. Totalizando quarenta e sete classes com potencial de reutilização dentro domínio de sistemas de gestão escolar.

4.2. Avaliação da Metodologia Baseada em Similaridade de Nomes

Para realizar um comparativo entre metodologia, será utilizada a metodologia "A Method Based on Naming Similarity to Identify Reuse Opportunities", citada nos trabalhos relacionados da seção 2.4. Esta metodologia tem como objetivo avaliar conjuntos de sistemas de mesmo domínio e realizar uma análise comparativa entre os nomes de suas classes e métodos para determinar possíveis componentes reutilizáveis (OLIVEIRA, 2016).

4.2.1. Conceito

Nesta metodologia Oliveira (2016) adota como parâmetro, uma taxa de similaridade de pelo menos 75% entre os nomes de classes analisadas, os resultados que alcançarem esse porcentual de semelhança são selecionados como potenciais componentes. A Figura 36 exemplifica o funcionamento do processo.

Figura 36 - Tabela de Avaliação de Similaridade

System A	System B	Similarity Rate
ShoppingCart	ShoppCart	75%
OrderProductId	OrderProduc	78%
Orderservice	Orderservi	83%
Reviwes	Reviwe	85%
Clients	Client	85%
CartController	CartControll	85%
Products	Product	87%
ProductsController	ProductController	94%

Fonte: Oliveira (2016)

A metodologia foi sintetizada em uma ferramenta chamada JReuse, produzida por esta pesquisa, o software importa os códigos-fontes, extrai as classes e métodos, computa as similaridades com base nos parâmetros de comparação lexicográfica, e então seleciona as classes que atendem o critério de similaridade.

A ferramenta JReuse não está disponível para uso, então para poder realizar o comparativo entre as duas metodologias apresentadas, será adotado o critério de 75% de semelhança entre os nomes de classes, e então efetuada a aplicação manual deste nos três sistemas usados pela metodologia anterior.

Na Tabela 11, estão listados os nomes das classes que compões cada projeto utilizado na aplicação da metodologia desta pesquisa. Cada classe está elencada na coluna de seu respectivo projeto. Em seguida será aplicada a metodologia proposta em "*A Method Based on Naming Similarity to Identify Reuse Opportunities*".

Tabela 11 - Tabela de listagem de nomes das classes nos três projetos

Nº	sistemagestaoescolar	GestaoEscolar	Kaderneta
1	Administrador	Aluno	Class
2	AdministradorDao	AlunoPosGraduacao	ClassController
3	Aluno	ApIAgenda	ClassRegister
4	AlunoDao	BaseDadosAlunos	ClassRegisterController
5	AnoLetivo	BaseDadosCursos	ClassRegisterRepository
6	AnoLetivoDao	BaseDadosDisciplinas	ClassRepository
7	Aproveitamento	BaseDadosProfessores	Discipline
8	AproveitamentoDao	BaseDadosProjetos	DisciplineController
9	AproveitamentoPK	BaseDadosTecnicos	DisciplineRepository
10	Bairro	ContaCorrente	Frequency
11	BairroDao	ContaCorrenteEspecial	FrequencyController
12	Cargo	Contato	FrequencyRepository
13	CargoDao	ContatoEmail	KadernetaApplication
14	Cidade	ContatoFax	KadernetaApplicationTests
15	CidadeDao	ContatoTelefone	Notes
16	Classe	ContatoTelefoneCelular	NotesController
17	ClasseDao	ContatoTelefoneComercial	NotesRepository
18	Contrato	ContatoTelefoneResidencial	Registration
19	ContratoDao	Curso	RegistrationController
20	ContratoPK	Departamento	RegistrationRepository
21	Deque	Dicionario	Student
22	DequeDao	Disciplina	StudentController
23	DequePK	Empresa	StudentDTO
24	Despesa	Execicio09	StudentDTOController
25	DespesaDao	Exercicio01	StudentDTORespository
26	DirectorTurma	Exercicio01	StudentRepository
27	DirectorTurmaDao	Exercicio02	User
28	Disciplina	Exercicio04	UserController
29	DisciplinaDao	Exercicio05	UserRepository
30	Distrito	Exercicio06	
31	DistritoDao	Exercicio07	
32	Encarregado	Exercicio08	
33	EncarregadoDao	Exercicio10	
34	Endereco	Exercicio11	
35	EnderecoDao	Exercicio12	
36	Exame	Exercicio13	
37	ExameAluno	Exercicio14	
38	ExameAlunoPK	FolhaPagamento	
39	ExameDao	Funcionario	
40	ExameDisciplinaDao	Funcionario	

Tabela 11 - Tabela de listagem de nomes das classes nos três projetos

(Continuação)

Nº	sistemagestaoescolar	GestaoEscolar
41	Expediente	JanelaSimples
42	Fornecedor	Menu
43	FornecedorDao	OpcaoInvalidaException
44	Funcionario	Persist
45	FuncionarioDao	Pesquisador
46	Fundos	Pessoa
47	FundosDao	Pessoa
48	GenericDao	Principal
49	Imposto	Professor
50	ImpostoDao	ProjetoPesquisa
51	Instituicao	SaldoInsuficienteException
52	InstituicaoDao	Tecnico
53	Matricula	TelaMenu
54	MatriculaDao	Teste
55	MatriculaPK	ValorNegativoException
56	Mensalidade	
57	MensalidadeDao	
58	Ocorrencia	
59	Parente	
60	ParenteDao	
61	Pessoa	
62	Privilegio	
63	Requisicao	
64	RequisicaoDao	
65	RequisicaoPK	
66	Sala	
67	SalaDao	
68	Seccao	
69	SeccaoDao	
70	SistemaEscolar	
71	Telefone	
72	TelefoneDao	
73	TipoPagamento	
74	TipoPagamentoDao	
75	Trimestre	
76	Turma	
77	TurmaDao	
78	Utilizador	
79	UtilizadorDao	

Fonte: Elaborada pelo autor (2021)

4.2.2. Resultados

Em seguida da análise e aplicação do parâmetro de similaridade, obteve-se a Tabela 12, com seis classes que atendem a taxa de pelo menos 75% de semelhança. Houve a variação entre 83,33% a 100% na taxa de similaridade dos resultados, o restante das classes não atendeu a medida solicitada devido não haver repetição de nomes ou uma taxa abaixo do padrão desejado na proposta.

Tabela 12 - Tabela de classes com similaridades acima de 75%

Nº	sistemagestaoescolar	GestaoEscolar	Kaderneta	Taxa de Similaridade
1	Aluno	Aluno	-	100%
2	Classe	-	Class	83,33%
3	Contato	Contato		100%
4	Disciplina	Disciplina	Discipline	90%
5	Funcionario	Funcionario	-	100%
6	Pessoa	Pessoa	-	100%

Fonte: Elaborada pelo autor (2021)

4.3. Comparativo de Resultados

Após executar ambas as metodologias, a etapa final deste capítulo se trata do comparativo dos valores obtidos. Antes de apresentar os dados, a metodologia abordada neste trabalho com base nas interações entre classes será identificada como Metodologia (I). Enquanto que a metodologia de comparativo entre nomes terá a identificação de Metodologia (II).

Na Tabela 13, estão dispostas as quantidades de classes classificadas como potenciais componentes reutilizáveis pelas metodologias em cada projeto. Ao fim da tabela estão dispostos os valores totais de classes selecionadas, existe uma diferença de 34 classes selecionadas.

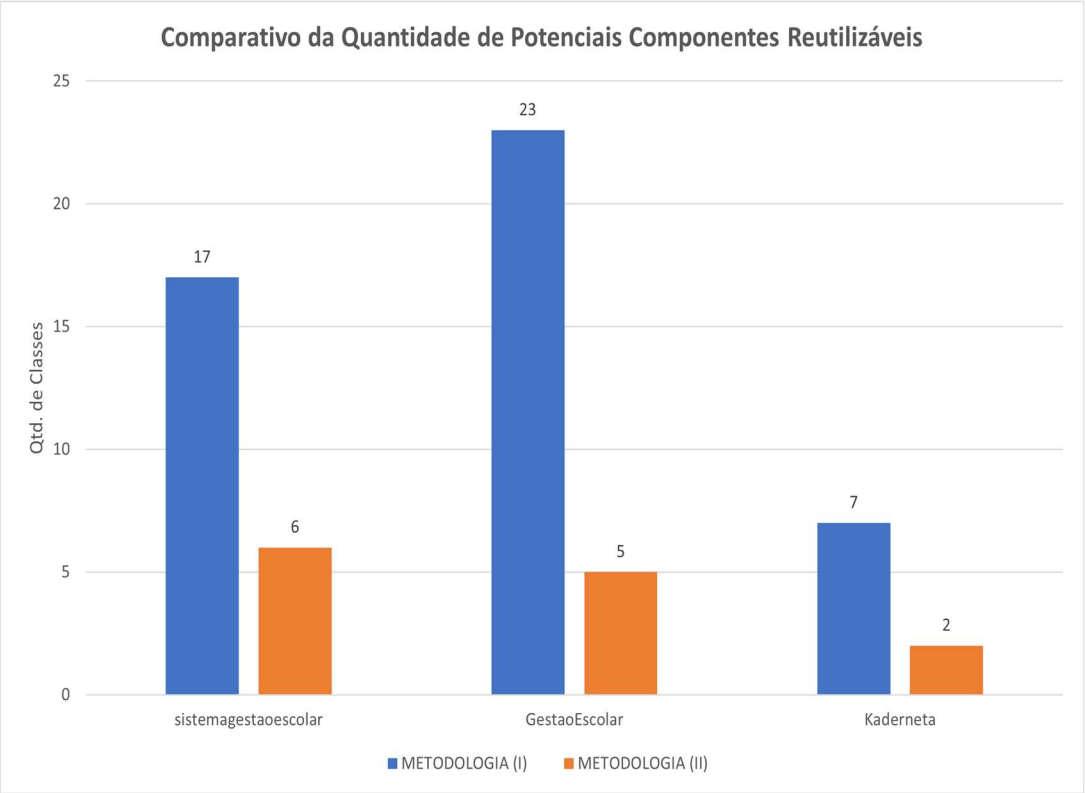
Tabela 13 - Comparativo da Quantidade de Potenciais Componentes Reutilizáveis

PROJETO	METODOLOGIA (I)	METODOLOGIA (II)
sistemagestaoescolar	17 classes	6 classes
GestaoEscolar	23 classes	5 classes
Kaderneta	7 classes	2 classes
	Total = 47 classes	Total = 13 classes

Fonte: Elaborada pelo autor (2021)

Para elucidar o paralelo de valores, a Figura 37 dispõe de um gráfico representando os valores da Tabela 13. Analisando as duas representações fica evidente a diferença dos dados obtidos em ambos processos. A metodologia (I) considerou em seu procedimento uma quantidade de 34 classes acima da metodologia (II).

Figura 37 - Comparativo de Quantidade de Potenciais Componentes Reutilizáveis



Fonte: Elaborada pelo autor (2021)

Outra análise que pode ser levada em consideração são as porcentagens de potencias componentes em relação do total de classes de cada projeto.

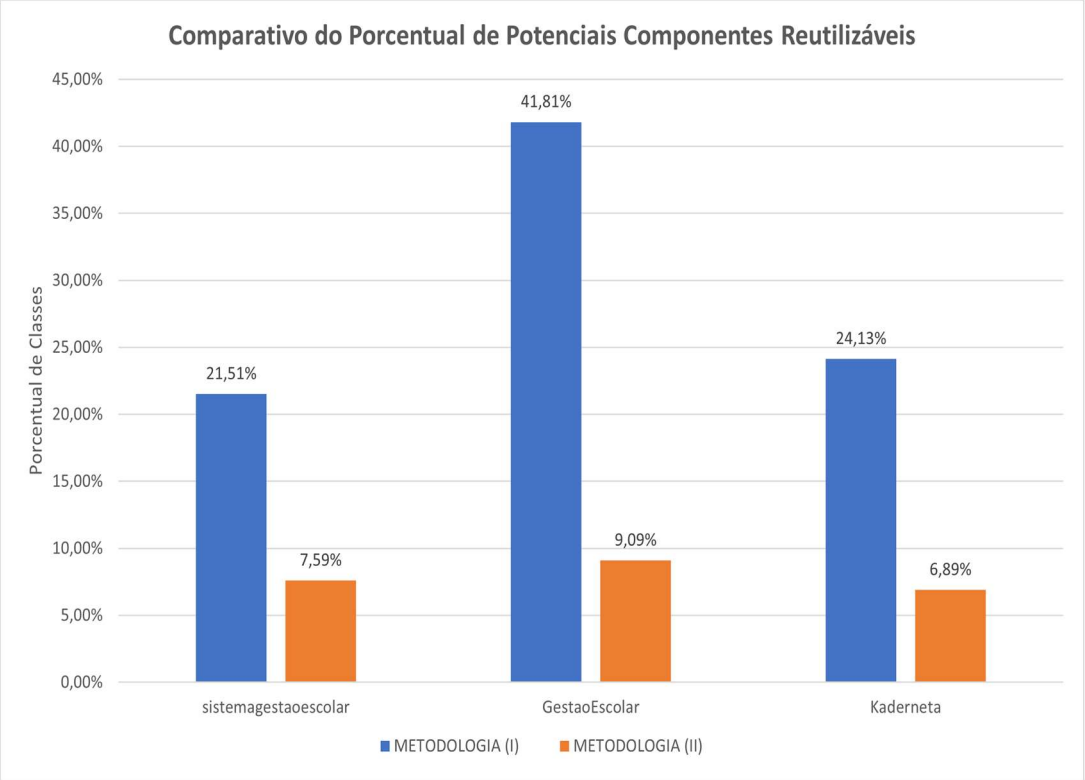
Tabela 14 - Comparativo do Porcentual de Potenciais Componentes Reutilizáveis

PROJETOS	METODOLOGIA (I)	METODOLOGIA (II)
sistemagestaoescolar	21,51%	7,59%
GestaoEscolar	41,81%	9,09%
Kaderneta	24,13%	6,89%
Conjuntos de Sistemas	28,83%	7,97%

Fonte: Elaborada pelo autor (2021)

A metodologia (I) teve um avaliou que 28,83% das classes dos três projetos têm o potencial de serem reutilizadas. Em contrapartida, a metodologia (II) alcançou um porcentual de 7,97% de possíveis componentes reutilizáveis no conjunto de sistemas. As duas pesquisas foram aplicadas nos três projetos e analisaram o total de 163 classes.

Figura 38 - Comparativo do Porcentual de Potenciais Componentes Reutilizáveis



Fonte: Elaborada pelo autor (2021)

Avaliando a quantidade total de classes potenciais alvitrada pela proposta de baseada em interações entre classes, teve o valor de 72% superior à proposta de comparativo de nomes. Analisando os resultados de cada projeto aferido, as quantidades de potenciais componente propostas têm uma diferença que varia entre 28%, (Kaderneta) e 78%, (GestaoEscolar) em suas contagens. Deixando claro a diferença entre os dados obtidos por cada uma das pesquisas.

Um ponto percebido durante a análise, foi a queda de valores, por parte da Metodologia (II), na seleção de classes no projeto Kaderneta, alcançando apenas 2 resultados, talvez por suas classes utilizarem o idioma inglês. Possivelmente, este fator dificulta o comparativo num conjunto de diversos projetos escritos em diversos idiomas.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como principal objetivo propor uma alternativa para obtenção de componentes reutilizáveis, sugestionando o processo de seleção através de analisando as classes com maior destaque em seu conjunto utilizando o conceito do desvio padrão.

A Engenharia de Software Baseada em Componentes contribui para desenvolvimento possibilitando maior praticidade e economia de recursos, devido construir o software através da montagem de módulos e otimizando o reuso (SZPERSKI, 1997).

Tendo esse conceito como foco, essa pesquisa visa avaliar o software e selecionar classes com potencial para serem reutilizadas em um próximo projeto de mesmo domínio. Sendo necessário apenas poucos ajustes, poupando o processo de desenvolver as funções desde o começo, agilizando o processo da produção de sistemas.

5.1. Resultados

Os resultados obtidos através da aplicação da metodologia em três projetos de mesmo domínio foram positivos. O procedimento selecionou as principais classes dos projetos, como por exemplo: aluno, professor, pessoa, matricula e disciplina, essas que são partes fundamentais para sistemas de gestão escolar.

Outro ponto positivo que a metodologia pode ser aplicada e obter resultados independentemente do idioma do projeto e da nomenclatura adotada pelo desenvolvedor. Tendo em vista que seleção dos nomes das classes não segue um padrão em todos os projetos, sendo subjetivo de quem está produzindo o software.

Logo, a aplicação do conceito estatístico do desvio padrão presume-se ser eficaz, em comparação a metodologia de comparativo dos nomes das classes de um software. Este que caso analise um conjunto misto de idiomas utilizados na nomenclatura das classes, pode ocorrer uma margem de erro maior na comparação dos nomes.

Embora a metodologia tenha alcançado um resultado satisfatório, sendo aplicada em três projetos, esta necessita mais pesquisa. Além de ser aplicada em um cenário com um conjunto amostral maior.

5.2. Trabalhos Futuros

Ainda que os resultados alcançados com a aplicação da metodologia nos três projetos utilizados neste trabalho. Contudo, em função da indisponibilidade de mais dados, existe a necessidade de aplicar o procedimento em um cenário com uma maior quantidade de projetos avaliados. Além que este trabalho carece de ser comparado com mais metodologias de identificação de componentes reutilizáveis

Por fim, outra etapa que somará a esta pesquisa, será a produção de um software para automatizar o processo de seleção das classes, possibilitando a importação dos projetos e em seguida aplicar a metodologia e exibir os resultados. Desta forma acelerando o processo de análise, e evitando a contabilização manual das interações, que demandam tempo.

REFERÊNCIAS BIBLIOGRÁFICAS

AMPATZOGLOU, A.; STAMELOS, I.; GKORTZIS, A.; DELIGIANNIS, I. A Methodology On Extracting Reusable Software Candidate Components From Open Source Games. 8 f. Artigo científico – World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering Vol:2, No:11, 2008.

BAUER, F.L. Dissent On Development. In NATO Conference Science Committee, 1 f. 1969.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML Guia do Usuário. 2ª Ed. São Paulo. Elsevier, 2006. 463p.

CALDIERA, G.; BASILI, V.R. Identifying and qualifying reusable software componets. 10 f. Artigo científico – University of Maryland, 1991.

COSTA, C. A Aplicação da Linguagem de Modelagem Unificada (UML) para o Suporte ao Projeto de Sistemas Computacionais dentro de um Modelo de Referência. 18 f. Artigo científico – Universidade de Caxias do Sul, 2001.

FEIJOO, A. Parte I – Estatística Descritiva. 8 f. Artigo - Centro Edelstein de Pesquisas Sociais, 2010.

FILHO, C.B. Reuso de Software Baseado nas Diretrizes do MPS.BR Nível E. 68 f. Trabalho de Conclusão de Curso – Universidade Estadual de Londrina, 2013.

FLAKES, W.B. Practical Software Reuse (Panel Position Paper). 2 f. Artigo científico – Computer Science Department Virginia Tech, Falls Church, 2014.

GUERRA, P.M.; SANTOS, S.R. Reutilização de Componentes de Software com Base em Identificadores Hierárquicos. 269 f. Dissertação de Doutorado - Universidade Técnica de Lisboa - Instituto Superior Técnico, 1999.

JHA, M; O'BRIEN, L. A comparison of software reuse in software development communities. In Software Engineering (MySEC), 2011, 5th Malaysian Conference in, 5 f. 2011.

JONES, C. Software return on investment preliminary analysis. In Software Productivity Research, Inc., 1993.

LUCRÉDIO, D. Uma Abordagem Orientada a Modelos para Reutilização de Software. 287 f. Dissertação de Doutorado - Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo, 2009.

LUNET, N.; SEVERO, M.; BARROS, H. Desvio Padrão ou Erro Padrão. 5 f. Artigo científico - Serviço de Higiene e Epidemiologia da Faculdade de Medicina da Universidade do Porto, 2006.

MARTINS, M. Desvio Padrão Amostral. 2 f. Revista científica - Faculdade de Ciências da Universidade de Lisboa, 2013.

OLIVEIRA, J.; FERNANDES, E.; SOUZA, M.; FIGUEIREDO, E. A method based on naming similarity to identify reuse opportunities. In: IEEE INTERNATIONAL CONFERENCE ON INFORMATION REUSE AND INTEGRATION, 2017, Las Vegas. p.10.

OLIVEIRA, J. A. A Method Based On Naming Similarity To Identify Reuse Opportunities. 80 f. Dissertação de Mestrado – Programa de Pós-Graduação em Ciência da Computação - Universidade Federal de Minas Gerais, 2016.

OLIVEIRA, M.; GONÇALVES, E.M.; BACILI, K.R. Automatic Identification of Reusable Software Development Assets: Methodology and Tool. In: XII BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS. 305., 2016, Florianópolis. p.6.

PRESSMAN, ROGER S. Engenharia de Software. 7ª Ed. Rio de Janeiro. McGraw- Hill, 2011. 843p.

ROSSI, A. C. Representação de Software na FARCSOFT: Ferramenta de Apoio à Reutilização de Componentes de Software. 253 f. Dissertação de Mestrado - Poli/USP, 2004.

SAMETINGER, J. Software Engineering with Reusable Components. Springer-Verlag Berlin Heidelberg, 1997. 272p.

SOMMEVILLE, IAN. Engenharia de Software. 9ª Edição. São Paulo. Pearson, 2011. 548p.

SOFTEX, Perspectivas de desenvolvimento e uso de componentes na Indústria Brasileira de Software e Serviços, 2007.

SOUZA, L.S. Engenharia de Software. 6 f. Artigo científico – Universidade Federal do Mato Grosso do Sul, Campo Grande, 2014.

SPAGNOLI, L.A,; BECKER, K. Um Estudo sobre o Desenvolvimento Baseado em Componentes. 48 f. Dissertação de Pós-Graduação – Faculdade de Informática - PUCRS, 2003.

SPAGNOLI, L.A,; REDOLFI, G.; BASTOS, R.M.; CRISTAL, M.; ESPINDOLA, A.P. Especificando Informações para Componentes Reutilizáveis. 59 f. Dissertação de Pós-Graduação – Faculdade de Informática - PUCRS, 2004.

TIOBE Index. (2021). TIOBE Quality Indicator change history. Acessado em 09 de julho de 2021, https://www.tiobe.com/tiobe-index/.

WANG, Z., XU, X. ZHAN, D. A Survey of Business Component Identification Methods and Related Techniques. 10 f. Artigo científico – In: International Journal of Information Technology Volume 2 Number 4, 2005.