INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAJAZEIRAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANÁLISE SOBRE O USO DO VOCABULÁRIO DE SOFTWARE NA DETECÇÃO DE PADRÕES DE PROJETO.

JOSÉ CARLOS RODRIGUES DE SOUSA

Cajazeiras
2021

JOSÉ CARLOS RODRIGUES DE SOUSA

ANÁLISE SOBRE O USO DO VOCABULÁRIO DE SOFTWARE NA DETECÇÃO DE PADRÕES DE PROJETO.

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Me. Ricardo de Sousa Job.

Cajazeiras
2021

Campus Cajazeiras
Coordenação de Biblioteca
Biblioteca Prof. Ribamar da Silva
Catalogação na fonte: Daniel Andrade CRB-15/593

S725a

Sousa, José Carlos Rodrigues de

Análise sobre o uso do vocabulário de software na detecção de padrões de projeto / José Carlos Rodrigues de Sousa; orientador Ricardo de Sousa Job.- 2021.
50 f.: il.

Orientador: Ricardo de Sousa Job.
TCC (Tecnólogo em Análise e Desenvolvimento) – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2021.

1. Padrão de projeto 2. Vocabulário de Software 3. Ontologia I. Título.

004.41(0.067)

SERVIÇO PÚBLICO FEDERAL
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
COORDENAÇÃO DE CURSOS - CAMPUS CAJAZEIRAS

Às **15:30** horas do dia **17** do mês de **junho** do ano de **2021**, via Google Meet, compareceu para defesa pública do **Trabalho de Conclusão de Curso**, requisito obrigatório para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, o(a) aluno(a) **JOSÉ CARLOS RODRIGUES DE SOUSA**, matrícula **201722010010**, tendo como Título do Trabalho **USO DO VOCABULÁRIO DE SOFTWARE NA DETECÇÃO DE PADRÕES DE PROJETO**. Constituíram a Banca Examinadora os professores **RICARDO DE SOUSA JOB** (orientador), **DIOGO DANTAS MOREIRA** (examinador) e **ANDRÉ LIRA ROLIM** (examinador).

Após a apresentação e as observações dos membros da Banca Examinadora, ficou definido que o trabalho foi considerado **APROVADO** com nota **100,** com a condição de que o (a) aluno (a) entregue, no prazo máximo de 30 dias, a versão final do trabalho, via processo eletrônico à coordenação de curso. A versão deve conter a ficha catalográfica e atender às sugestões feitas pelos membros da banca. O código fonte desenvolvido no trabalho (caso haja) deve ser enviado para o e-mail da coordenação do curso (cads.cz@ifpb.edu.br).

Cajazeiras-PB, 19 de junho de 2021.

Documento assinado eletronicamente por:

- **Andre Lira Rolim, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 28/06/2021 10:33:13.
- **José Carlos Rodrigues de Sousa, ALUNO (201722010010) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS**, em 24/06/2021 21:19:20.
- **Diogo Dantas Moreira, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 21/06/2021 08:19:11.
- **Ricardo de Sousa Job, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 19/06/2021 10:40:55.

Este documento foi emitido pelo SUAP em 18/06/2021. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 199003
Código de Autenticação: 7f242efb33

Dedico este trabalho a Deus, o autor de todas as coisas, como está escrito: portanto, quer comais quer bebais, ou façais qualquer outra coisa, fazei tudo para glória de Deus.

AGRADECIMENTOS

Agradeço a todos que estiveram ao meu lado e sempre me incentivaram a continuar e me dedicar, agradeço a todos que de alguma forma contribuíram para que eu pudesse chegar até aqui. Agradeço a Deus por não ter me permitido desistir nos momentos mais difíceis e agradeço a mim mesmo por continuar mesmo diante as dificuldades.

”Se você busca entender qual deve ser o sentido da vida, o que eu posso te responder é que creia você em Deus ou não, acredite em algo ou seja completamente cético, não existe nada mais nobre e gratificante do que ajudar a quem precisa. Experimente se colocar de lado pelo menos uma vez para ajudar a alguém que de fato precisa, faça de coração sem esperar nada em troca, faça de preferência sem que ninguém saiba e verá que é verdade o que eu estou falando.”

José Carlos

RESUMO

Detectar os padrões de projeto implementados em um sistema é uma importante tarefa para o auxílio em sua compreensão. Compreender o sistema é primordial para que ele possa ser mantido e sempre que necessário evoluído. Mas as ferramentas de detecção não conseguem ser completamente precisas ao realizar a detecção. Os identificadores do vocabulário aumentam a qualidade do software e são úteis na compreensão e manutenção do código-fonte. Recentemente alguns trabalhos têm indicado que algumas informações sobre o sistema só podem ser obtidas por meio do vocabulário de software. Devido a essas circunstâncias, é realizado neste trabalho uma análise com o auxílio de uma ferramenta sobre os termos do vocabulário que podem estar relacionados aos padrões de projeto. A avaliação de quatro projetos de código aberto constatou que existem sim termos fortemente relacionados aos padrões de projeto tanto nos identificadores que nomeiam as classes como nos identificadores que nomeiam métodos e atributos. Foi ainda constatado que a melhor maneira de utilizar os termos do vocabulário é fazê-lo em conjunto, utilizar os termos em conjunto resultou em maiores indícios de instância de padrão implementado, para o projeto *Jext* por exemplo, o maior valor de indício de padrão utilizando apenas um termo foi de 17% enquanto que com o uso conjunto dos termos o valor de indício de padrão foi para 100%. Porém, devido a fraca convenção de nomenclatura em relação a implementação de padrões de projeto e aos diferentes vocabulários utilizados pelas organizações, uma ferramenta que deseje utilizar-se dos termos do vocabulário no processo de detecção deve estar apta a receber o conjunto de termos de quem for utilizá-la. Como solução para essa fraca convenção de nomenclatura o autor deste trabalho propõe ainda a criação de uma ontologia para a definição dos termos que devem ser utilizados na implementação dos padrões de projeto.

Palavras-chave: Padrão de Projeto. Vocabulário de Software. Detecção. Ontologia.

ABSTRACT

Detecting the design patterns implemented in a system is an important task to aid in its understanding. Understanding the system is essential so that it can be maintained and whenever necessary evolved. But detection tools cannot be completely accurate when performing detection. Vocabulary identifiers increase the quality of the software and are useful in understanding and maintaining the source code. Recently, some studies have indicated that some information about the system can only be obtained through the software vocabulary. Due to these circumstances, an analysis is performed in this work with the aid of a tool on vocabulary terms that may be related to design patterns. The evaluation of four open source projects found that there are terms that are strongly related to the design patterns, both in the identifiers that name the classes and in the identifiers that name methods and attributes. It was also found that the best way to use the vocabulary terms is to do it together, using the terms together resulted in greater evidence of the instance of the implemented pattern, for the Jext project, for example, the highest value of pattern evidence using only one term was 17%, while using the terms together the pattern evidence value was 100%. But due to the weak naming convention in relation to the implementation of design patterns and the different vocabularies used by organizations, a tool that wishes to use vocabulary terms in the detection process must be able to receive the set of terms of whoever is going to use it. As a solution to this weak naming convention, the author of this work also proposes the creation of an ontology for the definition of the terms that should be used in the implementation of the design patterns.

Keywords: Design Pattern. Software Vocabulary. Detection. Ontology.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CCT	Classs comuns que possuem o termo
CINT	Classes de instância que não possuem o termo
CIT	Classes de instância que possuem o termo
CSV	Comma-Separated Values
LOC	Número de linhas de código
NOCL	Número de classes
VIP	Valor de indício de padrão
XMI	XML Metadata Interchange
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Para um projeto de software ser considerado bem desenvolvido, normalmente ele precisa ser reutilizável e flexível, ou seja, além de atender a requisitos específicos ele precisa estar adequado para que mudanças futuras possam ser facilmente realizadas e assim requisitos futuros venham a ser atendidos. Solucionar todos os problemas durante o desenvolvimento de um sistema e ao fim obter um sistema de qualidade, é uma tarefa muito difícil, por isso os desenvolvedores mais experientes utilizam-se de soluções já testadas e aprovadas em projetos anteriores. A essas soluções é dado o nome de padrões de projeto (GAMMA et al., 2000).

Além de desenvolvido, um projeto de software precisa ser mantido e sempre que necessário, evoluído, para isso, antes de qualquer coisa ele precisa ser devidamente compreendido. Mas, entender um código-fonte não é algo simples, por isso a tarefa de evoluir grandes sistemas é normalmente uma tarefa difícil pois devido à falta de documentação, muitas vezes o código-fonte é o único recurso para a compreensão do sistema (DONG et al., 2007).

Compreender um sistema a partir do código-fonte não é fácil, por isso a detecção dos padrões de projeto implementados no software é um dos meios utilizadas para compreendê-lo. As ferramentas de detecção de padrões de projeto são instrumentos indispensáveis aos desenvolvedores que tem a responsabilidade de manter e evoluir o projeto de software (FONTANA et al., 2012).

Uma atividade de manutenção de software que contribui para que ele esteja sempre apto a sofrer alterações é a refatoração de suas estruturas, (ANTONIOL et al., 2007) propõe que uma atividade semelhante deva ser realizada para a melhoria dos identificadores de código-fonte. Os identificadores são os nomes atribuídos pelos desenvolvedores às estruturas estáticas de código-fonte como, classes, métodos e atributos. Os desenvolvedores escolhem os identificadores que melhor identificam as estruturas, para que por meio deles ela seja rapidamente compreendida. A escolha de bons identificadores pode revelar a função no sistema, e a importância da entidade na tarefa de manutenção a ser realizada.

O identificador de uma estrutura de código pode conter um ou mais termos, os termos podem ser nomes, siglas ou abreviaturas. Ao conjunto de termos extraídos a partir dos identificadores de um código-fonte é dado o nome de vocabulário do software (QUINTANS et al., 2013).

1.1 MOTIVAÇÃO

Os padrões de projeto contribuem para a produção de software de qualidade, por meio deles são reutilizadas as experiências de especialistas para a solução de problemas recorrentes no processo de desenvolvimento de software. Além de utilizados para solucionar problemas, os padrões de projetos são utilizados para documentar parte do sistema, isso porquê ao capturar uma instância de padrão implementada captura-se também o entendimento sobre essa parte do sistema. Embora técnicas recentes melhorem a documentação dos softwares, ainda é essencial detectar os padrões para auxiliar na compreensão do sistema (DONG et al., 2009).

Recentemente, tem sido dada uma maior atenção ao vocabulário de software, visto que os identificadores do vocabulário aumentam a qualidade do software e são úteis na compreensão e manutenção do código-fonte. O identificadores são responsáveis por mais da metade das informações linguísticas no código-fonte e se escolhidos com cuidado, podem refletir a semântica e o papel das entidades de código no sistema (GUERROUJ, 2013).

Informações importantes para a compreensão do sistema podem ser obtidas através dos termos contidos nos identificadores do vocabulário de um software e além disso, segundo (SANTOS et al., 2015) algumas dessas informações não são identificadas de nenhuma maneira, mas somente através dos termos contidos nos identificadores do vocabulário de software.

Devido ao alto valor semântico contido nos identificadores do vocabulário e a essa compreensão de que algumas informações são contempladas apenas ao analisar o vocabulário, o autor deste trabalho se propôs a analisar qual o papel do vocabulário de software no processo de detecção dos padrões de projeto.

1.2 PROBLEMA

Os padrões de projeto revelam importantes informações sobre um sistema de software e saber quais padrões de projeto foram implementados em um sistema auxilia em sua compreensão. O que é fundamental para tarefas como de manutenção e evolução do software.

Detectar os padrões de projeto implementados em um código-fonte é uma tarefa muito importante, mas também não é uma tarefa simples. Com esta finalidade diversas ferramentas foram e ainda são desenvolvidas atualmente, isso devido ao fato de que nenhuma delas consegue ser completamente precisa em detectar os padrões

e apresentam resultados diferentes umas das outras mesmo quando são executadas sobre os mesmos projetos (FONTANA et al., 2012).

Segundo (SANTOS et al., 2015) o vocabulário pode revelar informações que não podem ser obtidas por nenhum outro meio, devido a isso é possível concluir que o uso do vocabulário no processo de detecção de padrões de projeto, pode contribuir para que as ferramentas de detecção obtenham uma maior precisão.

Neste trabalho é averiguado se existem termos contidos nos identificadores do vocabulário que apresentem informações úteis para a detecção de padrões de projeto.

1.3 OBJETIVO

O principal objetivo deste trabalho é verificar se é possível obter do vocabulário informações úteis ao processo de detecção de padrões de projeto e com isso evidenciar o vocabulário de software como um importante recurso que pode ser utilizado. Para atender a esse objetivo, elencamos os seguintes objetivos específicos:

- Realizar um levantamento bibliográfico sobre vocabulário de software.
- Realizar um levantamento bibliográfico sobre padrões de projeto.
- Obter o vocabulário de software de alguns sistemas.
- Identificar os padrões de projeto implementados nesses sistemas.
- Verificar se existe nos identificadores do vocabulário, termos que possuem relação com os padrões de projeto implementados nos sistemas.

Para cumprir esses passos, foi realizado um Estudo de Caso utilizando quatro sistemas de software de código aberto descritos na seção 4.3 e dois dos padrões de projeto definidos por (GAMMA, 2000), descritos na seção 4.6.

1.4 SOLUÇÃO

Para avaliar a abordagem desenvolvida a fim de atingir o objetivo deste trabalho, foram levantadas as seguintes questões de pesquisa:

- QP1: O vocabulário de software é utilizado no processo de detecção de padrões de projeto?

    1. H1: O vocabulário de software é utilizado no processo de detecção de padrões de projeto.
    2. H2: O vocabulário de software ainda não é utilizado por nenhuma ferramenta no processo de detecção de padrões.

- QP2: Existem termos relacionados aos padrões de projeto presentes nos identificadores do vocabulário de software?
- QP3: Quanto os termos do vocabulário podem indicar a presença de uma instância de padrão?
- QP4: Existe alguma maneira de melhor utilizar o vocabulário e assim alcançar maiores indícios da presença de um instância de padrão?

Tomando a hipótese H1 da QP1 como verdadeira, uma resposta positiva pode ser logo atribuída à QP2, responder QP3 e QP4 podem ainda trazer importantes contribuições sobre o uso do vocabulário no processo de detecção de padrões. Em pesquisas iniciais, não foram encontrados estudos que analisassem o uso do vocabulário de software no processo de detecção de padrões. Nesse sentido, a conclusão deste trabalho contribui para evidenciar o vocabulário como um importante recurso para a obtenção de uma maior precisão na tarefa de detectar os padrões.

Tomando a hipótese H2 da QP1 como verdadeira, apenas comprovar que existe no vocabulário termos que possam auxiliar o processo de detecção já evidencia o vocabulário de software como um importante recurso que pode ser utilizado na detecção de padrões de projeto e responder as questões QP3 e QP4 tornam ainda mais significativo a realização deste trabalho.

1.5 RESULTADOS

Após a pesquisa bibliográfica sobre padrões de projeto, vocabulário de software e técnicas de detecção de padrões, foi constatado que um termo presente no identificador que nomeia classes do padrão *Strategy* foi utilizado pela ferramenta *DP-Miner* para fazer distinção entre os padrões *Strategy, State* e o *Bridge* pois mesmo utilizando análises estáticas e dinâmicas sobre os projetos a ferramenta *DP-Miner* não conseguiu distinguir entre estes padrões até utilizar-se do termo.

Durante a definição dos termos de busca que seriam utilizados neste trabalho, foi constatado que além de termos presentes nos identificadores que nomeiam as classes era possível haver também termos relacionados aos padrões de projeto presentes

nos identificadores que nomeiam métodos e atributos e isso foi constatado através de análise direta sobre o código-fonte das classes que pertenciam à instâncias de padrão.

Com o auxílio da ferramenta desenvolvida neste trabalho foi realizada uma análise sobre o uso dos termos para identificar qual a melhor maneira em utilizá-los na detecção dos padrões e os resultados obtidos comprovaram que o uso em conjunto dos termos que podem estar presentes nos identificadores que nomeiam as classes e também nos identificadores que nomeiam métodos e atributos é a maneira mais eficaz em utilizá-los, para o projeto *Jext* por exemplo o maior valor de indício de padrão utilizando apenas um termo foi de 17% enquanto que com o uso conjunto dos termos o valor de indício de padrão atingiu o valor máximo de 100%.

O uso em conjunto dos termos atribui mais exclusividade às classes de instância de padrão, por outro lado se um deles não é encontrado os resultados são comprometidos, por isso a melhor maneira do uso em conjunto deles é atribuindo um peso à classe dependendo de quantos e quais termos ela possui.

1.6 ORGANIZAÇÃO DO DOCUMENTO

Este documento está organizado nos seguintes capítulos. O capítulo 2 apresenta a fundamentação teórica deste trabalho. O capítulo 3 apresentada a ferramenta desenvolvida para a realização deste estudo de caso. O capítulo 4 apresenta a validação do estudo de caso e os resultados obtidos. O capítulo 5 apresentada as considerações finais do trabalho e as suas contribuições. Por fim são apresentadas as referências bibliográficas.

2 FUNDAMENTÇÃO TEÓRICA

Neste capítulo é descrita a fundamentação teórica deste trabalho. A seção 2.1 apresenta a definição de padrões de projeto. A seção 2.2 apresenta as principais técnicas para a detecção de padrões e ferramentas de detecção. A seção 2.3 apresenta o que é um vocabulário de software, as técnicas para extração e manipulação de termos de um vocabulário, uma ferramenta que realiza essa tarefa e um exemplo de utilização de um vocabulário para resolver problemas relacionados ao desenvolvimento de sistemas.

2.1 PADRÕES DE PROJETO

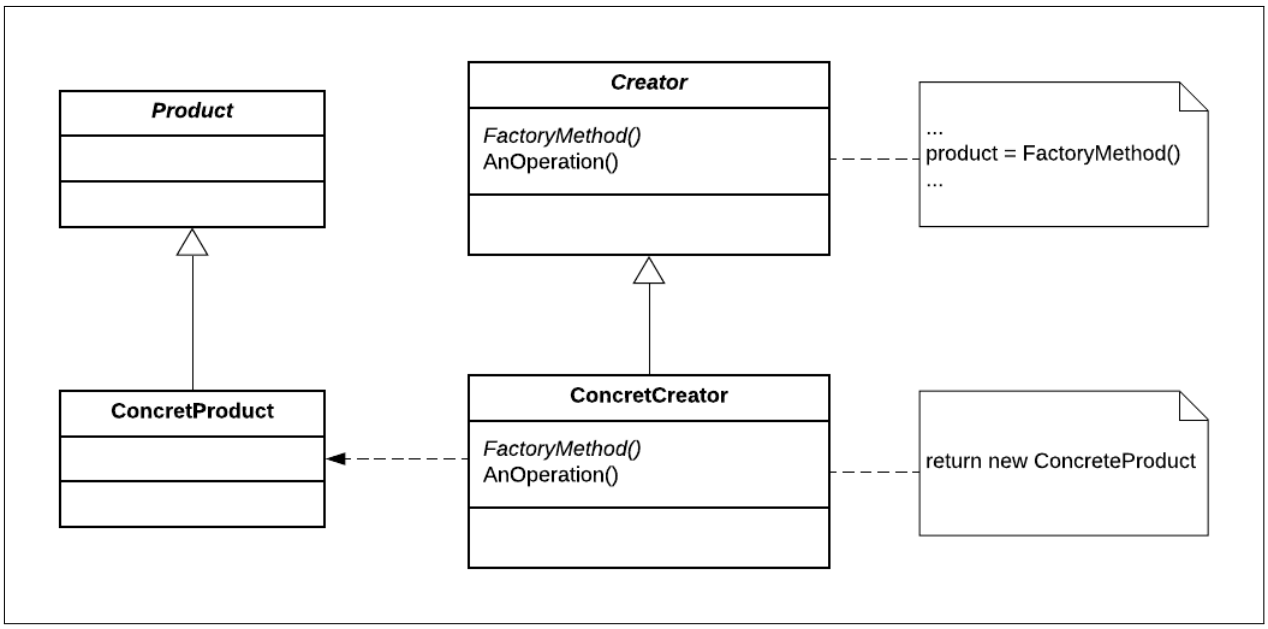
Segundo (GAMMA et al., 2000) um padrão de projeto é uma forma padrão de resolver problemas específicos que frequentemente são encontrados durante o desenvolvimento de projetos de software. Um padrão descreve classes e objetos que se comunicam para resolver um problema. Um padrão de projeto possui quatro elementos que são essenciais, são eles:

- Nome: o nome do padrão faz referência ao problema de projeto que o padrão se dispõe a resolver, ou suas soluções e consequências.
- Problema: explica o problema e em qual situação o padrão pode ser aplicado.
- Solução: descreve as responsabilidades, os relacionamentos e as colaborações entre os elementos do padrão.
- Consequências: são os resultados obtidos quando se aplica o padrão, suas vantagens e desvantagens.

Em (GAMMA et al., 2000) é utilizado um gabarito para descrição detalhada de cada um dos padrões de projeto definidos, a seguir um exemplo de como os elementos do gabarito Estrutura e Participantes são apresentados.

Estrutura: Notação baseada em OMT que representa o padrão. Nesta notação uma classe é representada por um retângulo com o seu nome na parte de cima, abaixo do nome aparecem as operações, após as operações as variáveis definidas pela classe.

Figura 1 – Estrutura do padrão Factory Method



Fonte: Elaborado pelo autor, 2021

Participantes: lista de participantes e suas funções dentro do padrão.

- Product: define a interface do objeto que o método fábrica cria.
- ConcretProduct: implementa Product.
- Creator: declara o método FactoryMethod que retorna um objeto do tipo Product.
- ConcretCreator: implementa o método FactoryMethod para que ele retorne um ConcreteProduct.

2.2 TÉCNICAS E FERRAMENTAS

Existem diversas técnicas utilizadas no processo de detecção de padrões de projeto, de acordo com a revisão de técnicas e ferramentas realizada por (DONG et al., 2009), as diferentes abordagens para detecção podem utilizar-se de aspectos estruturais, comportamentais ou ambos, outras abordagens também incluem aspectos semânticos.

Segundo (DONG et al., 2009) a maioria das ferramentas verificam apenas os aspectos estruturais, ou seja, os relacionamentos de agregação, associação e generalização entre as classes e os atributos e métodos. A análise dos aspectos estruturais é a técnica base no processo de detecção de padrões. Mesmo as ferramentas que capturam os aspectos comportamentais ou os aspectos semânticos, precisam antes de tudo obter os resultados da análise estrutural.

Os aspectos comportamentais são geralmente a invocação de métodos e podem ser verificados de forma estática, ou seja, sem execução do sistema ou, em tempo de execução do sistema. Os aspectos comportamentais podem revelar se candidatos à instâncias de padrões são reais ou são falsos positivos, ou seja, classes e objetos que não são instância de padrão mas podem ser capturados como uma instância, implementada no sistema.

Os aspectos semânticos embora se refiram ao significado das entidades no sistema, algumas abordagens utilizam-se de aspectos semânticos diferentes, por exemplo:

- Uma ferramenta verifica se as relações entre as classes são obrigatórias ou opcionais e se são relações de um-para-um ou um-para-muitos;
- Uma outra ferramenta utiliza-se das convenções de nomenclatura, por exemplo, Vector e HashTable podem indicar a multiplicidade de uma agregação.
- E ainda outra ferramenta explora os padrões das convenções de nomenclatura buscando no nome das classes termos que indiquem o papel delas em uma instância de padrão.

As abordagens mais atuais utilizam-se de outras ferramentas para obterem uma versão intermediária do código-fonte, isso simplifica o processo de detecção. A maioria delas usa como formato intermediário a Abstract Syntax Tree (AST) ou Abstract Semantic Graph (ASG), outras ainda utilizam outras representações como matrizes e vetores. Sobre as representações intermediárias podem ser aplicadas diferentes técnicas (DONG et al., 2009).

A ferramenta ***Design Pattern Detection*** (DPD) proposta por (TSANTALIS et al., 2006) realiza uma análise estrutural do sistema, ela obtém uma versão intermediária do código-fonte através de uma estrutura que manipula bytecode Java e fornece todos os detalhes do código-fonte do sistema, através dessa estrutura são construídas as matrizes que representam o sistema. A representação do sistema a ser analisado e

dos padrões de projeto é dada em forma de matriz. A ideia é que a estrutura de um padrão é um grafo que pode ser mapeado em uma matriz quadrada.

A detecção é realizada através do cálculo da similaridade entre as vértices dos grafos. Essa abordagem busca encontrar os padrões mesmo que estejam implementados em versões modificadas. Esta metodologia quando aplicada em todo o sistema pode causar lentidão, por isso a ferramenta particiona o sistema analisado em hierarquias comunicantes para que o algoritmo seja aplicado não ao sistema inteiro de uma vez, mas à subsistemas. As classes que não fazem parte de nenhuma hierarquia são agrupadas pois em alguns padrões algumas funções podem ser realizadas por classes que não pertencem a hierarquias, esse processo resulta em um melhor desempenho. A busca por padrões é realizada em cada subsistema de forma separada e como entrada externa, informações como o nome do padrão poderiam ser facilmente fornecidas (TSANTALIS et al., 2006).

Por sua vez, a ferramenta ***DP-Miner*** proposta por (DONG et al., 2007) inicialmente averiguava os aspectos estruturais e também comportamentais de um sistema, na abordagem o sistema é representado por uma matriz e cada célula possui um valor que representa as características da classe, como métodos, atributos e relacionamentos. Os padrões são também representados por matrizes e se uma matriz correspondente à matriz que representa o padrão for encontrada no sistema, uma instância de padrão candidata é encontrada.

Assim como as ferramentas mais atuais de detecção, a busca por padrões é realizada através da análise sobre representações intermediárias de código-fonte, o padrão *XML Metadata Interchange* (XMI) é o formato utilizado, os arquivos XMI são obtidos a partir de diagramas UML mapeados para o formato XML.

Após a análise estrutural, os aspectos comportamentais são averiguados para a detecção de falsos positivos, mas devido à padrões como o *Strategy, State* e o *Bridge* possuírem aspectos estruturais e comportamentais bem semelhantes, mesmo utilizando-se dos aspectos estruturais e comportamentais a abordagem não conseguiu distinguir entre eles.

No entanto, através de uma análise direta no código-fonte foi percebido que o termo *“strategy”* estava presente no nome das classes do padrão *Strategy*, por isso além das análises estruturais e comportamentais a abordagem passou a se utilizar também dos termos que nomeiam as classes(ou seja, aspectos semânticos) e assim foi possível fazer a distinção entre esses padrões.

2.3 VOCABULÁRIO DE SOFTWARE

O vocabulário de software que também pode ser por alguns autores chamado de léxico do código-fonte é composto pelo conjunto de termos que podem ser extraídos dos identificadores do código e também dos comentários. Os identificadores são usados para nomear estruturas estáticas de código-fonte como classes, atributos e métodos. Durante a manutenção de um sistema, o desenvolvedor passa a maior parte do tempo buscando compreender o código-fonte. Por isso, os desenvolvedores buscam escolher os identificadores que melhor identificam as estruturas, para que por meio deles ela seja rapidamente compreendida (SANTOS et al., 2015).

Os identificadores reduzem o tempo e o esforço empreendidos na compreensão do código-fonte, eles tornam desnecessária a leitura de todo o código de uma estrutura para a compreensão do papel dela dentro do sistema (ANTONIOL et al., 2007).

Em (ANTONIOL et al., 2007) foi realizada uma análise sobre a evolução do léxico/vocabulário ao longo da evolução de três grandes sistemas de software, a saber, Eclipse, Mozilla e Alice. Essa análise constatou que as estruturas lexicais e estruturais tendem a atingir níveis cada vez mais altos de estabilidade ao longo do tempo, ou seja, tendem a sofrer menos mudanças, mas a estabilidade lexical se manteve sempre superior que a estabilidade estrutural. Sendo o léxico sempre mais estável durante a evolução de um sistema. Com isso concluiu que os identificadores são essenciais para que os programadores compreendam as estruturas do sistema, por esse motivo a produtividade dos programadores com a função de manter e evoluir o software é diretamente afetada pela escolha de bons identificadores.

Definindo a qualidade dos identificadores como a possibilidade de compreendê-lo a partir de palavras do dicionário ou de abreviaturas conhecidas, (LAWRIE et al., 2007) realizou uma avaliação sobre a evolução da qualidade dos identificadores ao longo de três décadas, cobrindo 186 programas desenvolvidos em quatro linguagens de programação. O estudo constatou que a qualidade dos identificadores têm aumentado com o passar dos anos, concluiu ainda que código-fonte aberto possui uma quantidade maior de palavras do dicionário enquanto que softwares privados possuem mais abreviações, por isso ferramentas que visam código privado devem estar preparadas para aceitar as abreviações definidas localmente ou para aprender e incluir essas abreviações.

Além dessas informações, estudos têm indicado que algumas informações extraídas dos vocabulários de software não são obtidas através de análises estruturais ou comportamentais. Essas informações podem ser úteis para entendimento e apri-

moramento de sistemas. A literatura ainda indica que a qualidade dos identificadores também contribuem para uma maior qualidade de software e ajudam a prolongar a vida útil dos sistemas (QUINTANS et al., 2013).

Para realizar a tarefa de manutenção de um sistema, identificar o desenvolvedor especialista, ou seja, o mais apto a manter determinadas entidades de código é uma tarefa que pode poupar tempo e esforço. Com base na importância do vocabulário de software e nas informações que podem ser reveladas pelos identificadores que não são contempladas por outros meios, como análises estáticas ou dinâmicas, (SANTOS et al., 2015) faz uso dos identificadores do vocabulário para obter uma melhor precisão na realização da tarefa de identificação de especialistas. Para isso utiliza-se dos *commits* em um VCS (*Version Control System*) para capturar os identificadores utilizados por cada desenvolvedor e assim obter os seus vocabulários que são armazenados em matrizes, também o vocabulário do código-fonte que é composto pelo vocabulário de cada entidade do sistema é armazenado como uma matriz. O desenvolvedor apropriado a manter determinadas entidades é aquele que possui uma maior similaridade entre o seu vocabulário e o vocabulário da entidade.

Segundo (GUERROUJ, 2013) uma boa quantidade do vocabulário é formada por acrônimos/abreviaturas ou pela concatenação de termos e por isso não podem ser identificados prejudicando a obtenção dos benefícios das convenções de nomenclatura. A fim de eliminar essas ambiguidades propõe uma abordagem para a normalização do vocabulário que se dá através da divisão e da expansão dos identificadores do vocabulário.

Ao trabalhar com vocabulário de software é preciso então normalizá-lo, extrair os termos dos identificadores, para isso é necessário usar técnicas de *Information Retrieval* (IR) sobre eles, os termos que não formam palavra nenhuma mas são abreviaturas ou siglas são denominados como *tokens*, por isso a técnica de extração dos termos do vocabulário também é conhecida como tokenização, para a sua realização é normalmente utilizado o padrão *Camelcase*, que separa as palavras através do primeiro caractere que é sempre maiúsculo (QUINTANS et al., 2013).

A ferramenta ***VocabularyTool*** descrita por (QUINTANS et al., 2013) extrai os identificadores presentes nas classes, atributos, métodos, interfaces, enums, variáveis locais e parâmetros. Após extrair os identificadores ela os grava em um arquivo no formato *Vocabulary Extended Language* (VXL) organizando-os na mesma estrutura em que são encontrados no código-fonte para que em seguida ele possa ser lido e apenas os identificadores de classes, parâmetros, atributos, variáveis locais e métodos sejam carregados para a realização da tokenização desses identificadores, ou seja,

para a extração dos termos dos identificadores. Para a tokenização a *VocabularyTool* utiliza-se de filtros como o *Camelcase* que é a separação dos termos através da letra inicial maiúscula e o *Underscore* que é a separação dos termos pelo símbolo *underscore/underline*.

Além do processo de tokenização, a ferramenta *VocabularyTool* também realiza a radicalização dos termos, ou seja, ela remove os afixos e assim extrai apenas o radical de algumas palavras.

Um exemplo de uso de um vocabulário para a resolução de problemas em desenvolvimento de sistemas pode ser observado no trabalho realizado por (FALBO; BERTOLLO, 2005) que propôs o estabelecimento de uma ontologia, um vocabulário comum sobre os processos de software.

Para atingir o objetivo de produzir software de qualidade, grandes organizações definem, baseados no domínio da aplicação, os meios como: tecnologias, artefatos e procedimentos para desenvolver, implantar e manter um produto de software. À esse conjunto de definições é dado o nome de processo de software, do qual depende a qualidade de um produto de software, por isso, as organizações adotam modelos e padrões de qualidade para melhorar seus processos de software. Mas, segundo (FALBO; BERTOLLO, 2005), devido ao diferente vocabulário desses modelos e padrões de qualidade surgem os problemas relacionados ao uso em conjunto deles. Por isso, ele propôs o estabelecimento de uma ontologia, que é um vocabulário que busca capturar conceitos por meio dos seus termos, é a teoria de um conteúdo, uma maneira de comunicação para um domínio específico.

3 FERRAMENTA DESENVOLVIDA

A ferramenta desenvolvida segundo a abordagem proposta na seção 4.5, tem como propósito realizar a busca por termos relacionados aos padrões de projeto no vocabulário de software. Após recuperados os termos eles são manipulados para que o valor que representa o indício de uma instância de padrão seja apresentado.

A ferramenta suporta os padrões de projeto definidos por (GAMMA et al., 2000) detalhados na seção 4.6.

3.1 FERRAMENTAS UTILIZADAS

A ferramenta de extração do vocabulário de software tem como *output* um arquivo no formato *Comma-Separated Values* (CSV) contendo o nome das classes do projeto, os termos encontrados no projeto e a frequência em que esses termos aparecem dentro de cada classe, o *output* da ferramenta de extração de vocabulário *VocabularyTool* pode ser visualizado na Figura 2.

Figura 2 – Output da ferramenta VocabulayTool

	A	B	C
1		/lib/:org.jext.project.ProjectManagement	/lib/:org.jext.console.Console$ConsoleProcess$StderrThread
2	been	1	0
3	string	1	0
4	jext	1	0
5	lt	14	0
6	project	8	0
7	separ	1	0
8	heavier	1	0
9	sourceforg	1	0
10	interfac	2	0
11	get	2	0
12	should	1	0
13	have	1	0
14	implement	2	0
15	from	1	0
16	benson	1	0
17	href	1	0
18	net	1	0
19	so	1	0
20	class	1	0
21	us	1	0
22	which	1	0
23	all	1	0
24	e	12	0
25	made	1	0

Fonte: Elaborado pelo autor, 2021

A ferramenta de detecção de padrões *Design Pattern Detection usign Similarity Scoring* (DPD) além de apresentar seus resultados em uma interface gráfica, tem como *output* um arquivo em XML contendo todos os padrões detectados no código-fonte. Para cada padrão, são listadas todas as suas instâncias e apresentadas as principais classes e métodos definindo seus papéis dentro da definição do padrão que pode ser vista na seção 2.1. Em relação ao padrão de projeto *Factory Method* a DPD apresenta a classe com o papel Creator e o método FactoryMethod com o seu retorno ConcreteProduct para cada instância desse padrão. O *output* da ferramenta pode ser visualizado na Figura 3.

Figura 3 – Output da ferramenta Design Pattern Detection

```
<?xml version="1.0" encoding="UTF-8"?>
<system>
    <pattern name="Factory Method">
        <instance>
            <role name="Creator" element="Options$DisplayIro" />
            <role name="FactoryMethod()"
element="Options$DisplayIro::getInverseOptions():Options$DisplayIro" />
        </instance>
        <instance>
            <role name="Creator" element="org.jext.misc.FindFilterFactory" />
            <role name="FactoryMethod()"
element="org.jext.misc.FindFilterFactory::createFindFilter():org.jext.misc.FindFilter" />
        </instance>
        <instance>
            <role name="Creator" element="org.jext.project.ProjectManagement" />
            <role name="FactoryMethod()"
element="org.jext.project.ProjectManagement::getProjectManager():org.jext.project.ProjectManager" />
        </instance>
    </pattern>
    <pattern name="Prototype" />
    <pattern name="Singleton">
        <instance>
            <role name="Singleton" element="UML$Type" />
            <role name="uniqueInstance" element="UML$Type::CLASS:UML$Type" />
            <role name="uniqueInstance" element="UML$Type::INTERFACE:UML$Type" />
            <role name="uniqueInstance" element="UML$Type::INNER_CLASS:UML$Type" />
            <role name="uniqueInstance" element="UML$Type::METHOD:UML$Type" />
            <role name="uniqueInstance" element="UML$Type::ATTRIBUTE:UML$Type" />
            <role name="uniqueInstance" element="UML$Type::EXTENDS:UML$Type" />
            <role name="uniqueInstance" element="UML$Type::IMPLEMENTS:UML$Type" />
            <role name="uniqueInstance" element="UML$Type::THROWS:UML$Type" />
            <role name="uniqueInstance" element="UML$Type::ERROR:UML$Type" />
        </instance>
        <instance>
```

Fonte: Elaborado pelo autor, 2021

Como pode ser visto na Figura 3, as classes de retorno dos métodos do padrão *Factory Method* representam segundo a definição demonstrada na seção 2.1 o papel de ConcreteProduct. Nessas classes o termo de busca creat não deve ser encontrado.

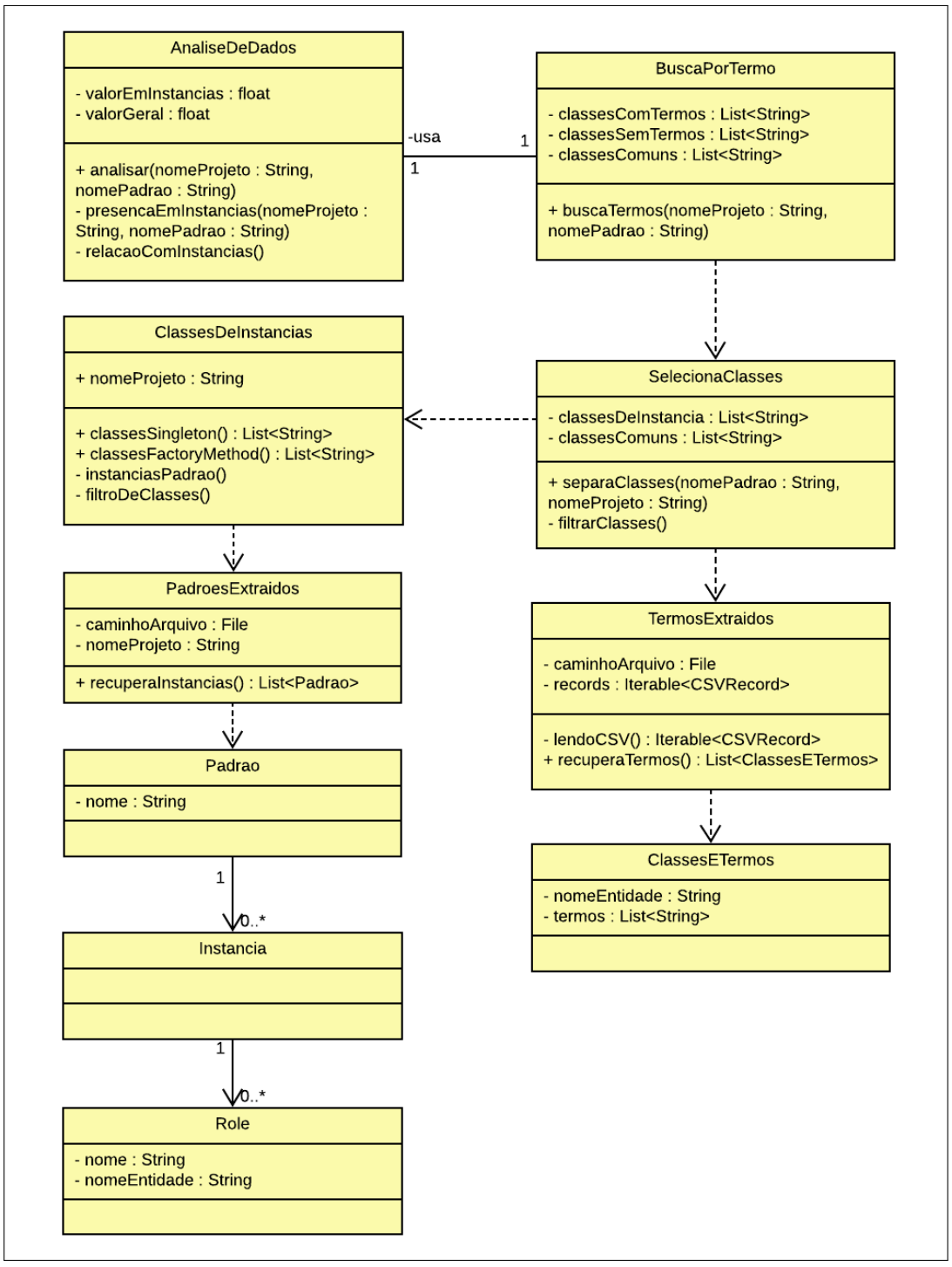
O termo creat é normalmente encontrado nos participantes Creator e ConcreteCreator, dessa forma, embora a classe ConcreteProduct faça parte de uma instância do padrão, ela não deve ser considerada no experimento. Através de observações como essa, foram selecionadas quais as classes de cada padrão utilizado deveriam fazer parte do experimento.

3.2 PRINCIPAIS FUNCIONALIDADES

As principais funcionalidades da ferramenta desenvolvida estão descritas a seguir.

Importação dos dados. Esta ferramenta tem como primeiro *input* um arquivo em formato XML contendo as informações sobre as instâncias de cada padrão detectado, este *input* está de acordo com o padrão do arquivo exportado pela ferramenta *Design Pattern Detection usign Similarity Scoring* proposta por (TSANTALIS et al., 2006), a classe responsável pela leitura dos arquivos é a classe PadroesExtraidos. O segundo *input* é um arquivo em formato CSV fornecido pela ferramenta *VocabularyTool*, ele oferece o vocabulário de software do projeto exibindo os termos pertencentes a cada classe, a classe responsável pela leitura desse arquivo é a classe TermosExtraidos. Os detalhes sobre a escolha por essas ferramentas podem ser visualizados na seção 4.1. As principais classes da ferramenta podem ser visualizadas no diagrama de classes apresentado na Figura 4.

Figura 4 – Diagrama de classes da ferramenta

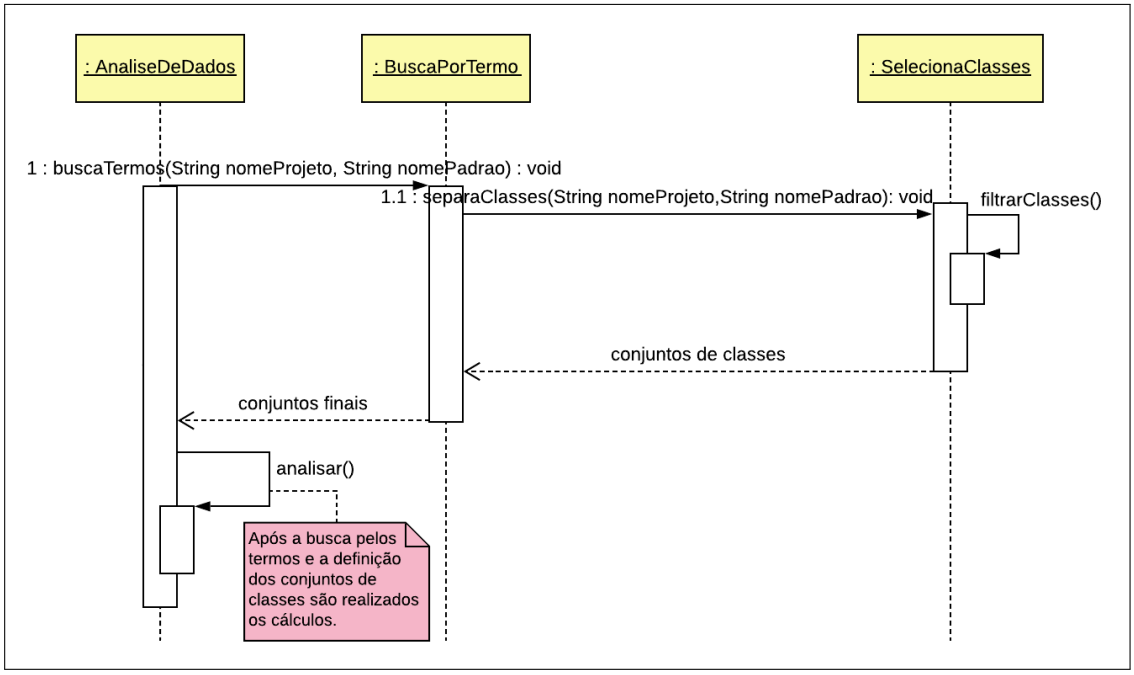


Fonte: Elaborado pelo autor, 2021

Manipulação de dados. De posse dos dados importados, a primeira etapa de manipulação é a separação das classes do projeto analisado, onde somente as

classes do próprio projeto são selecionadas, classes externas ao projeto como por exemplo bibliotecas não são consideradas no experimento, esta tarefa é realizada pela classe SelecionaClasses. Em seguida as classes são divididas em dois grupos, as que compõem alguma instância do padrão e as que não compõem nenhuma instância do padrão, a classe SelecionaClasses utiliza-se da classe ClassesDeInstancias para recuperar somente as classes de instância do padrão e assim as classes do projeto são divididas nos dois grupos de classes. O próximo passo é a realização da busca pelos termos definidos na seção 4.6 nos dois grupos de classes selecionadas, tarefa realizada pela classe BuscaPorTermo, o fim dessa tarefa resulta nos três conjuntos finais de classes. Por fim, a classe AnaliseDeDados é a responsável pela realização dos cálculos de acordo com a métrica definida na seção 4.2, após os cálculos são obtidos o valor de indício de padrão para cada projeto e a porcentagem das classes que compõem instâncias de padrão que possuem o termo. O diagrama de sequência que apresenta a troca de mensagens entre alguns objetos para a manipulação dos dados pode ser visualizado na Figura 5.

Figura 5 – Diagrama de sequência manipulação

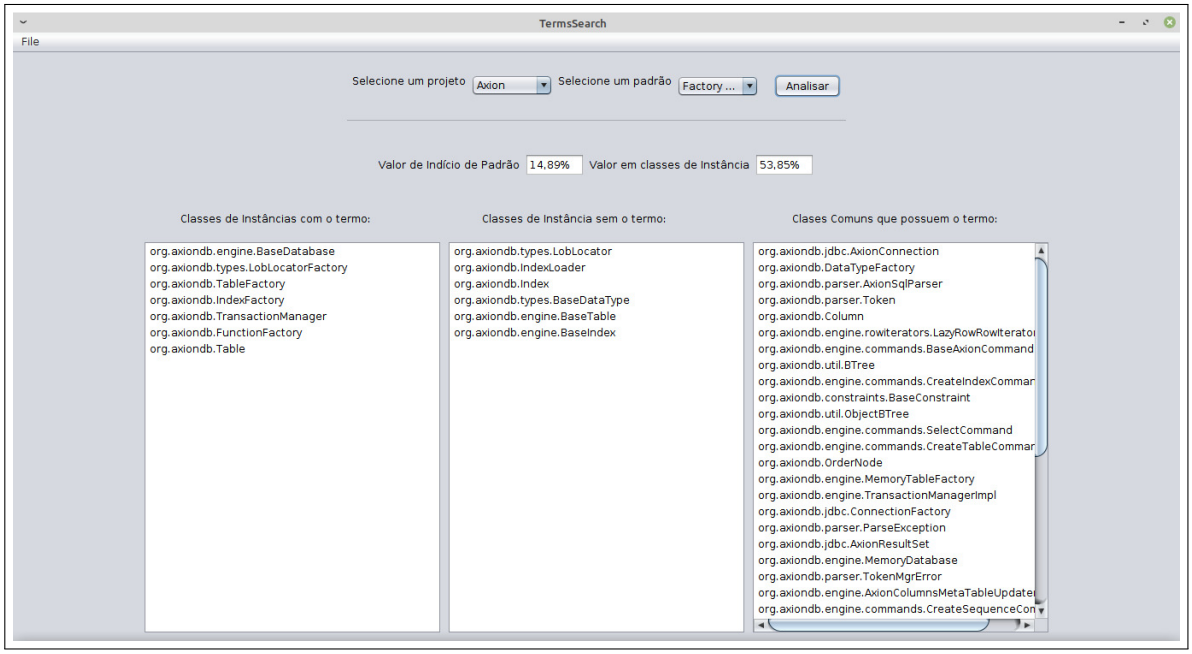


Fonte: Elaborado pelo autor, 2021

Exibição e exportação de resultados. Após a manipulação dos dados o usuário pode visualizar: Valor de indício de padrão, esse valor representa o quanto a presença

do termo está associado à instâncias de padrão implementadas; Porcentagem das classes que possuem o termo de busca e compõem instâncias do padrão, esse valor representa o quanto o termo é utilizado na implementação do padrão; Lista das classes de instância que possuem o termo; Lista das classes de instância que não possuem o termo. Lista das classes comuns que possuem o termo. A ferramenta exibe todas essas informações e as exportam em um arquivo no formato XML para facilitar análises futuras. Esses dados são apresentados graficamente através da interface gráfica que pode ser observada na Figura 6.

Figura 6 – Interface da ferramenta de apoio durante análise de dados



Fonte: Elaborado pelo autor, 2021

Caso o usuário queira extrair os resultados, ele pode no menu da ferramenta exportar esses dados para um arquivo no formato XML de acordo com a Figura 7.

Figura 7 – Output da ferramenta de apoio

```
▼<padrao>
   <valorEmInstancias valor="53.846157"/>
   <valorGeral valor="14.893617"/>
  ▼<classesComTermo>
     <classe nome="org.axiondb.engine.BaseDatabase"/>
     <classe nome="org.axiondb.types.LobLocatorFactory"/>
     <classe nome="org.axiondb.TableFactory"/>
     <classe nome="org.axiondb.IndexFactory"/>
     <classe nome="org.axiondb.TransactionManager"/>
     <classe nome="org.axiondb.FunctionFactory"/>
     <classe nome="org.axiondb.Table"/>
   </classesComTermo>
  ▼<classesSemTermo>
     <classe nome="org.axiondb.types.LobLocator"/>
     <classe nome="org.axiondb.IndexLoader"/>
     <classe nome="org.axiondb.Index"/>
     <classe nome="org.axiondb.types.BaseDataType"/>
     <classe nome="org.axiondb.engine.BaseTable"/>
     <classe nome="org.axiondb.engine.BaseIndex"/>
   </classesSemTermo>
  ▼<classesComuns>
     <classe nome="org.axiondb.jdbc.AxionConnection"/>
     <classe nome="org.axiondb.DataTypeFactory"/>
     <classe nome="org.axiondb.parser.AxionSqlParser"/>
     <classe nome="org.axiondb.parser.Token"/>
     <classe nome="org.axiondb.Column"/>
     <classe nome="org.axiondb.engine.rowiterators.LazyRowRowIterator"/>
     <classe nome="org.axiondb.engine.commands.BaseAxionCommand"/>
     <classe nome="org.axiondb.util.BTree"/>
     <classe nome="org.axiondb.engine.commands.CreateIndexCommand"/>
```

Fonte: Elaborado pelo autor, 2021

4 VALIDAÇÃO

Esta seção apresenta a validação do Estudo de Caso desenvolvido de acordo com a abordagem proposta na seção (4.5) para averiguar o valor de indício de padrão de projeto implementado no software, obtido através do uso do vocabulário de software.

4.1 INSTRUMENTAÇÃO

A maioria das ferramentas de detecção apresentam apenas o número de padrões de projeto detectados, outras apresentam apenas graficamente a localização do padrão, por esse motivo a ferramenta de detecção de padrões escolhida foi a *Design Pattern Detection usign Similarity Scoring* (DPD)[1] pois ela também apresenta o nome das classes que compõem as instâncias de padrão detectadas. É uma ferramenta desenvolvida em linguagem de programação Java que detecta padrões de projeto em projetos Java, o método de detecção usado por essa ferramenta é descrito na seção 2.2.

A ferramenta de extração de vocabulário escolhida para a execução desse experimento foi a *VocabularyTool*[2], ela foi desenvolvida para extrair o vocabulário de software de projetos em linguagem Java, mais detalhes sobre essa ferramenta podem ser lidos na seção 2.3.

A linguagem de programação utilizada no desenvolvimento da ferramenta de apoio para a análise dos resultados denominada como *TermsPatterns*[3] foi a mesma que as ferramentas de detecção de padrões e extração de vocabulário, ou seja, a linguagem Java e com o uso do JDK 1.8. O IDE utilizado para o desenvolvimento foi o NetBeans 8.2.

Para a realização da leitura dos arquivos em formato XML fornecidos pela ferramenta *Design Pattern Detection usign Similarity Scoring* e escrita dos arquivos de *output* dessa ferramenta foi utilizada a API JDOM². Para a leitura do arquivo em formato CSV fornecido pela ferramenta *VocabularyTool* foi utilizado a biblioteca Apache Commons CSV.

[1] http://users.encs.concordia.ca/∼nikolaos/pattern_detection.html
[2] https://sites.google.com/site/softwarevocabularies/vocabulary-tool/download
[3] https://github.com/gpes/terms-patterns

4.2 MÉTRICAS

Após as classes do projeto serem divididas em, classes de instância com termo (CIT), classes de instância que não possuem o termo (CINT) e classes comuns que possuem o termo (CCT) é possível a realização de cálculos para a obtenção de métricas.

É importante saber se os termos que se relacionam com os padrões são constantemente utilizados nos identificadores de código-fonte das classes que compõem as instâncias dos padrões. Para saber qual a porcentagem das classes de instância que possuem o termo o seguinte cálculo é efetuado.

$$USO = \frac{CIT}{CIT + CINT}.$$

Um outro valor muito importante é o valor da presença do termo em relação à uma instância do padrão (VIP), esse valor tem grande importância quanto ao objetivo deste trabalho e é útil para a resposta da QP3. O valor VIP é o valor que podemos atribuir à uma classe que possui o termo de busca, esse valor indica a probabilidade desta classe compor uma instância de padrão implementada. O VIP pode ser obtido através do seguinte cálculo.

$$VIP = \frac{CIT}{CIT + CCT}.$$

4.3 DEFINIÇÃO DA AMOSTRA

Os projetos escolhidos para a análise foram extraídos através do *Qualita.Class Corpus* proposto por (TERRA et al., 2013), são projetos *open-source* disponíveis para a execução de experimentos desenvolvidos em linguagem Java que é a linguagem suportada pelas ferramentas de detecção de padrões e extração de vocabulário. Foram selecionados cinco projetos que possuíam em seu código-fonte instâncias dos padrões escolhidos para análise. Os projetos avaliados e as informações sobre eles podem ser vistos na Tabela 1.

4.4 METODOLOGIA

Inicialmente foi realizado um levantamento bibliográfico sobre o tema de estudo para a obtenção de mais conhecimento sobre o problema, esse levantamento foi em

Tabela 1 – Projetos selecionados do Qualitas.Class Corpus

	Versão	KLOC	NOCL
axion	1.0-M2	24.2	257
Jext	5.0	60.2	761
Quickserver	1.4.7	18.3	196
Sandmark	3.4	93.2	1045

Fonte: Elaborado pelo autor, 2021

primeiro momento realizado a partir da revisão de literaturas realizadas por (MAYVAN et al., 2017). Após decidido os objetivos deste trabalho, foi definida a metodologia utilizada no Estudo de Caso a ser realizado.

Os métodos para a realização do trabalho são os seguintes: pesquisa bibliográfica sobre padrões de projeto, métodos de detecção de padrões utilizados pelas ferramentas e vocabulário de software; detecção dos padrões implementados no software; extração do vocabulário de software; realização da busca pelos termos relacionados aos padrões de projeto dentro do vocabulário de software extraído; realização de cálculos para a obtenção do indício de instância de padrão implementada nos projetos, onde todas as classes que possuem os termos são selecionadas para a realização dos cálculos. A seguir é apresentada a abordagem proposta por este trabalho e a métrica para a obtenção dos valores de indício.

4.5 ABORDAGEM PROPOSTA

Esta seção apresenta a abordagem proposta por este trabalho para identificar os temos que possuem relação com os padrões de projeto tendo como resultado principal um valor que se refere a ligação entre o termo de busca encontrado no vocabulário do código-fonte e a presença de uma instância do padrão implementada no sistema.

Na abordagem proposta é preciso primeiramente de: (i) a extração do vocabulário de software; (ii) a detecção dos padrões de projeto presentes no código-fonte. Estas duas atividades são realizadas pelas ferramentas descritas na seção 4.1 e detalhadas nas seções 2.2 e 2.3.

Para extração do vocabulário de software utilizamos a *VocabularyTool*, que permite a identificação dos identificadores das classes do projeto, os termos encontrados no projeto e a frequência em que esses termos aparecem dentro de cada classe.

Por sua vez, para a detecção de padrões de projeto utilizamos a ferramenta

Design Pattern Detection usign Similarity Scoring(DPD), pois possui como *output* um arquivo em XML contendo todos os padrões detectados no código-fonte. Para cada padrão, são listadas todas as suas instâncias e apresentadas as principais classes e métodos definindo seus papéis dentro da definição do padrão que pode ser vista na seção 2.1.

Depois de receber como *input* os arquivos de *output* das ferramentas *VocabularyTool* e *Design Pattern Detection usign Similarity Scoring*(DPD), a ferramenta proposta deve relacionar as informações contidas nesses arquivos. Como os dois arquivos apresentam o nome das classes, esse é o meio utilizado para relacioná-los. As classes do projeto são então divididas em dois conjuntos, classes que compõem instância de padrão e classes comuns(classes que não compõem instâncias de padrão), após essa separação das classes a busca pelo termo é efetuada.

Após a busca pelo termo o conjunto de classes que compõem instâncias do padrão é dividido em outros dois conjuntos, as classes de instância com termo (CIT) e as classes de instância que não possuem o termo (CINT) enquanto que do conjunto das classes comuns são selecionadas apenas aquelas que possuem o termo (CCT).

Em seguida os cálculos de métrica definidos na seção 4.2 são realizados e os resultados são exibidos em tela, podendo ser salvos em arquivos no formato XML.

4.6 PADRÕES E TERMOS DE BUSCA

Os padrões foram selecionados após uma análise sobre os projetos do *Qualita.Class Corpus*, os padrões *Singleton* e *Factory Method* por serem comuns a maioria dos projetos foram os escolhidos e considerados suficientes para o experimento.

Para a seleção dos termos, os termos levantados pelo trabalho realizado por (SILVA, 2019) foram considerados, além disso, o autor deste trabalho realizou ainda uma análise sobre a estrutura e os participantes que compõem a descrição de um padrão como detalhada na seção 2.1, e em seguida, uma análise sobre o código fonte de algumas classes de instância de padrão para a verificação dos termos, assim o padrão de nomenclatura utilizada pelos projetos do *Qualita.Class Corpus* foi identificado e foram enfim determinados os termos para o experimento que podem ser vistos no Quadro 1.

Quadro 1 – Padrões e Termos

Padrões	**Termos**
Factory Method	factory, create
Singleton	singleton, instance

Fonte: Elaborado pelo autor, 2021

O passo seguinte foi realizar uma análise manual no vocabulário dos projetos, através dessa análise foi identificado que a ferramenta *VocabularyTool*, por meio do uso das técnicas de *Information Retrieval* (IR) descritas na seção 2.3 aplicadas sobre os identificadores, reduz o termo *instance* relacionado ao padrão de projeto *Singleton* ao radical *instanc* e reduz o termo *create* que é relacionado ao padrão *Factory Method* ao radical *creat*, por esse motivo, ao buscar pelos termos na matriz CSV fornecida pela ferramenta *VocabularyTool* foram utilizados como termos de busca os radicais descritos no Quadro 2.

Quadro 2 – Padrões e Radicais

Padrões	**Termos**
Factory Method	factory, creat
Singleton	singleton, instanc

Fonte: Elaborado pelo autor, 2021

4.7 ANÁLISE DOS RESULTADOS

Esta seção apresenta a análise dos resultados e com isso as respostas para as questões de pesquisa levantadas para este Estudo de Caso.

4.7.1 Pesquisa bibliográfica

Para responder a QP1 formulada neste trabalho o autor precisou realizar uma fina pesquisa bibliográfica.

QP1: O vocabulário de software é utilizado no processo de detecção de padrões de projeto?

1. H1: O vocabulário de software é utilizado no processo de detecção de padrões de projeto.
2. H2: O vocabulário de software ainda não é utilizado por nenhuma ferramenta no processo de detecção de padrões.

Através da revisão de técnicas e ferramentas realizada por (DONG et al., 2009) descrita na seção 2.2, a hipótese H1 da QP1 mostrou-se verdadeira. A ferramenta *DP-Miner* também descrita na seção 2.2 utiliza-se de termos do vocabulário referentes ao nome dos padrões de projeto para distinguir entre projetos com características semelhantes como *Strategy, State* e o *Bridge*.

O uso dos termos do vocabulário por meio da *DP-Miner* **corrobora** que o vocabulário de software pode prestar uma grande contribuição às ferramentas de detecção para que possam atingir maiores valores de precisão ao detectar os padrões de projeto.

QP2: Existem termos relacionados aos padrões de projeto presentes nos identificadores do vocabulário de software?

Com uma resposta verdadeira sendo atribuída a hipótese H1 da QP1 compreende-se que uma resposta verdadeira logo deve ser atribuída a QP2, sim, existem termos presentes nos identificadores que possuem relação com os padrões de projeto implementados.

4.7.2 Análise manual dos dados.

Uma primeira análise dos dados foi realizada manualmente, esta primeira análise foi realizada sobre o código-fonte das classes que compõem instâncias de padrões implementados, essas classes foram identificadas a partir da ferramenta de identificação de padrões utilizada neste experimento. Essa primeira análise de dados mostrou que não existe nenhum termo específico utilizado na implementação dos padrões diferente daqueles que podem ser observados na descrição deles em (GAMMA et al., 2000), essa análise foi útil para a observação do padrão de nomenclatura utilizado pelos projetos do *Qualita.Class Corpus* e definição dos termos de busca.

4.7.3 Segunda análise de dados

A segunda análise de dados foi realizada através da ferramenta de apoio. Nesta análise, apenas os termos referentes aos nomes dos padrões de projeto foram considerados, isso porque essa foi a maneira que a ferramenta *DP-Miner* se utilizou do vocabulário e conseguiu distinguir entre diferentes padrões. Os termos então foram pesquisados dentro dos identificadores que nomeiam as classes, os resultados obtidos utilizando-se apenas os termos *“singleton”* e *“factory”* são apresentados nas Tabelas 2 e 3 respectivamente.

Como pode ser visto na coluna USO da Tabela 2, o termo *singleton* não foi

Tabela 2 – Padrão Singleton com o termo singleton.

	CIT	CINT	CCT	USO	VIP
axion-1.0-M2	0	3	0	0,00	0,00
Jext-5.0.	0	1	0	0,00	0,00
Quickserver-1.4.7	0	5	0	0,00	0,00
Sandmark-3.4	0	49	0	0,00	0,00

Fonte: Elaborado pelo autor, 2021

encontrado em nenhuma das classes dos projetos analisados. Isso significa que embora seja um termo diretamente relacionado ao nome do padrão, não é possível utilizar-se dele no processo de detecção de padrões para os projetos do *Qualita.Class Corpus*, porque é um termo que não faz parte do vocabulário utilizado.

Tabela 3 – Padrão Factory Method com o termo factory.

	CIT	CINT	CCT	USO	VIP
axion-1.0-M2	4	9	8	0,31	0,33
Jext-5.0.	1	1	5	0,50	0,17
Quickserver-1.4.7	0	1	3	0,00	0,00
Sandmark-3.4	0	12	5	0,00	0,00

Fonte: Elaborado pelo autor, 2021

Analisando a coluna USO da Tabela 3 é possível verificar que o termo *factory* diferentemente do termo *singleton*, foi utilizado em alguns projetos para nomear classes de instância do padrão. Analisando a coluna VIP, é visto que o valor de indício de padrão obtido através do termo *factory* para o projeto *Axion* é de 33%, ou seja, se uma classe desse projeto possui o termo *factory* em seu nome ela tem uma probabilidade de 33% de fazer parte de uma instância do padrão *Factory Method*.

Através dessa primeira análise com o auxílio da ferramenta de apoio, fica claro que o uso dos termos deve ser feito apenas como suporte à uma ferramenta de detecção, visto que os termos não são amplamente utilizados e não são exclusivos das classes que compõem instâncias de padrões, eles podem ser facilmente encontrados em classes comuns. Mas uma resposta à questão de pesquisa QP3 já pode ser atribuída.

QP3: Quanto os termos do vocabulário podem indicar a presença de uma instância de padrão?

Como pôde ser visto nas tabelas 2 e 3, o valor de indício de padrão varia tanto quanto a própria presença do termo nas classes que compõem as instâncias do padrão, mas depende também da presença deles nas classes comuns, ou seja, daquelas que

não fazem parte de nenhuma instância do padrão. Quanto mais utilizado e mais restrito às classes de instância, maior o valor de indício.

4.7.4 Terceira análise de dados

Durante o processo de definição dos termos de busca, além dos termos relacionados ao nome dos padrões, termos que poderiam ser encontrados no interior das classes como em métodos e atributos também foram selecionados. Nesta terceira análise de dados apenas os termos que podem ser encontrados nas estruturas internas foram considerados. Os resultados da busca pelos termos *instanc* e *creat* para os padrões *Singleton* e *Factory Method* respectivamente, são apresentados nas Tabelas 4 e 5.

Tabela 4 – Padrão singleton com o termo instanc

	CIT	CINT	CCT	USO	VIP
axion-1.0-M2	3	0	47	1,00	0,06
Jext-5.0.	1	0	23	1,00	0,04
Quickserver-1.4.7	1	4	15	0,20	0,06
Sandmark-3.4	44	5	32	0,90	0,58

Fonte: Elaborado pelo autor, 2021

Analisando a Tabela 4 e comparando com a Tabela 2 é possível observar uma grande diferença entre o uso dos termos, enquanto que o termo *singleton* não foi encontrado em nenhum dos projetos, o termo *instanc* é amplamente utilizado, chegando a ser encontrado em 100% das classes que compõem instâncias do padrão nos projetos *Axion* e *Jext*, mas também é encontrado em classes comuns o que faz com que o valor de indício de padrão não seja tão alto mesmo o termo sendo utilizado em todas as classes de instância desses dois projetos.

No projeto *Sandmark* por exemplo, pode ser observado na coluna USO da Tabela 4 que em 90% de suas 49 instâncias do padrão *Singleton* o termo *instanc* foi utilizado e isso mostra que de fato o termo *instanc* possui forte relação com o padrão *Singleton*. Embora seja também um termo facilmente encontrado em classes comuns do projeto *Sandmark*, nele o valor de instância de padrão que pode ser visualizado na coluna VIP da Tabela 4 foi o maior valor detectado com a utilização de um termo que pode ser encontrado na estrutura interna de uma classe de instância de padrão, que foi 58%.

Semelhantemente aos resultados do padrão *Singleton* para os termos *singleton* e *instanc*, na tabela 5 é visto que o termo *creat* também é muito mais utilizado do que o termo *factory* apresentado na Tabela 3, por outro lado, por ser também um termo mais

Tabela 5 – Padrão Factory Method com o termo creat

	CIT	CINT	CCT	USO	VIP
axion-1.0-M2	7	6	40	0,54	0,15
Jext-5.0.	1	1	67	0,50	0,01
Quickserver-1.4.7	1	0	18	1,00	0,05
Sandmark-3.4	5	7	112	0,42	0,04

Fonte: Elaborado pelo autor, 2021

comum o valor de indício de padrão encontrado com o uso do termo *factory* foi maior que o valor encontrado com o uso do termo *creat*.

Com essa terceira análise de dados é possível concluir que sim, termos que podem ser encontrados nas estruturas internas de uma classe podem possuir relação com os padrões implementados, isso ficou muito claro com os resultados atingidos principalmente pelo projeto *Sandmark* na Tabela 4 com a presença do termo *instanc* em 90% de suas 49 instâncias do padrão *Singleton*.

4.7.5 Quarta análise de dados

Como pôde ser visto nas duas análises anteriores, é possível utilizar-se de termos que nomeiam as classes e também de termos que podem ser encontrados internamente nas classes para obter um indício de padrão. Nesta quarta análise de dados é verificado o uso conjunto dos termos, ou seja, o uso tanto dos termos que nomeiam as classes quanto dos termos que podem nomear as estruturas internas da classe. O uso em conjunto dos termos pode atribuir mais exclusividade às classes de instância de padrão e exclusividade é o ponto chave para o uso dos termos do vocabulário, quanto mais exclusivo forem os termos maior a chance de se tratar de uma instância de padrão implementada ao detectar uma classe que possua os termos.

As Tabelas 6 e 7 apresentam os valores de uso e de indício de padrão para o uso conjunto dos termos, ou seja, para o padrão *Singleton* os termos *singleton* e *instanc* foram pesquisados e para o padrão *Factory Method* os termos *factory* e *creat* foram pesquisados.

Tabela 6 – Padrão singleton uso conjunto dos termos

	CIT	CINT	CCT	USO	VIP
axion-1.0-M2	0	3	0	0,00	0,00
Jext-5.0.	0	1	0	0,00	0,00
Quickserver-1.4.7	0	5	0	0,00	0,00
Sandmark-3.4	0	49	0	0,00	0,00

Fonte: Elaborado pelo autor, 2021

Como pode ser observado na Tabela 6, pelo fato dos valores serem atribuídos pelo uso conjunto dos termos, os valores obtidos através da busca pelo termo *instanc* observados na Tabela 4 foram prejudicados pelo fato do termo *singleton* não ser utilizado nos projetos, observando isso é possível concluir que a melhor maneira de fazer o uso conjunto dos termos é atribuindo uma pontuação a classe dependendo de quantos e de quais dos termos de busca ela possui.

Tabela 7 – Padrão Factory Method uso conjunto dos termos

	CIT	CINT	CCT	USO	VIP
axion-1.0-M2	4	9	4	0,31	0,50
Jext-5.0.	1	1	0	0,50	1,00
Quickserver-1.4.7	0	1	3	0,00	0,00
Sandmark-3.4	0	12	5	0,00	0,00

Fonte: Elaborado pelo autor, 2021

Ao ser observada a coluna VIP da Tabela 7 é possível verificar que o valor de indício de padrão para o projeto *Axion* com o uso conjunto dos termos é de 50%, valor maior do que o maior valor para o projeto *Axion* usando apenas um dos termos que foi o valor de 33% que pode ser observado na Tabela 3, tabela que apresenta os resultados para o termo *factory*.

Ainda ao observar a Tabela 7 é possível constatar que o valor de indício de padrão para o projeto *Jext* com o uso conjunto dos termos foi o valor máximo, 100% como pode ser visto na coluna VIP, valor bem superior ao valor mais alto com o uso de apenas um dos termos que foi 17% para o termo *factory*, que pode ser observado na coluna VIP da Tabela 3.

O uso conjunto dos termos aumenta o valor de indício porque confere exclusividade às classes de instância, observando a coluna CCT da Tabela 7 é possível constatar que o número de classes comuns com o termo foi 0 para o projeto *Jext*, número que chegou a ser de 67 com o uso de um só termo como pode ser visto na coluna CCT da Tabela 5.

4.7.6 Quinta análise de dados

Após a constatação de que, além dos termos presentes nos identificadores que nomeiam as classes, também os termos presentes nos identificadores que nomeiam métodos e atributos possuem relação com os padrões de projeto implementados e verificado que o uso em conjunto deles pode acarretar em uma maior exclusividade e consequentemente um maior valor de indício, é possível atribuir uma resposta a questão de pesquisa QP4.

QP4: Existe alguma maneira de melhor utilizar o vocabulário e assim alcançar maiores indícios da presença de uma instância de padrão?

Sim. Como pôde ser visto nessas análises de dados, além dos termos dos identificadores que nomeiam as classes, método utilizado pela ferramenta *DP-Miner*, também os termos dos identificadores que nomeiam métodos e atributos possuem relação com os padrões implementados e o uso em conjunto deles pode proporcionar uma maior precisão ao indicar a presença de uma instância de padrão implementada.

O melhor uso do vocabulário de software para a detecção de padrões de projeto é então, como demonstrado por este estudo o uso conjunto dos termos do vocabulário atribuindo um peso à classe dependendo dos termos que ela possui, mas um dos problemas em relação ao uso do vocabulário é que não há uma forte convenção de nomenclatura quanto aos termos que devem ser utilizados para a implementação dos padrões de projeto, por não haver essa forte convenção os termos relacionados aos padrões não são amplamente utilizados nas classes de instância e ainda podem ser facilmente encontrados nas classes que não fazem parte de instância alguma de padrão, as classes comuns. Neste estudo foram considerados softwares de código aberto, mas como é sabido as equipes de desenvolvimento e organizações privadas possuem seus próprios padrões de nomenclatura ao desenvolver software e por não haver essa forte convenção de nomenclatura relacionado aos padrões elas utilizam-se dos seus padrões de nomenclatura também ao implementar os padrões de projeto.

Como descrito na seção 2.3, segundo (LAWRIE et al., 2007) código-fonte privado também possui muito mais abreviaturas (definidas quando se trabalha com vocabulário de software como *tokens*) do que código-fonte aberto, por esse motivo sugere que as ferramentas que visam código privado devem estar preparadas para aceitar as abreviações ou para aprender e incluir essas abreviações.

Considerando essas informações é possível concluir que uma ferramenta que deseje utilizar-se dos termos do vocabulário para o processo de detecção dos padrões deve estar apta a receber esses termos como *input* daqueles que irão utilizá-la, essa é uma boa e rápida maneira de contornar esses problemas relacionados a fraca convenção de nomenclatura e aos diferentes vocabulários utilizados pelas organizações e equipes de desenvolvimento.

Uma outra maneira de não apenas contornar esse problema relacionado a nomenclatura, mas resolvê-lo, é à exemplo do trabalho de (FALBO; BERTOLLO, 2005) descrito na seção 2.3, a criação de uma ontologia para os padrões de projeto, uma definição dos termos a serem utilizados na implementação de cada padrão.

Para a resolução definitiva em relação a convenção de nomenclatura, essa ontologia deveria ser criada e a amplamente aceita pela comunidade, ela ficaria então disponível na web à exemplo do trabalho de (FONTANA et al., 2012) e assim poderia ser facilmente consultada por qualquer equipe de desenvolvimento ou organização. Informações sobre novos padrões ou variantes poderiam ser adicionadas, dessa maneira todos os benefícios relacionados aos termos do vocabulário levantados por este e por inúmeros outros trabalhos poderiam ser colhidos em relação aos padrões de projeto, pois o problema em relação a convenção de nomenclatura dos padrões estaria resolvido.

4.8 AMEAÇAS À VALIDADE

Algo que poderia ser uma ameaça a validade deste trabalho seria a precisão da ferramenta *Design Pattern Detection usign Similarity Scoring*(DPD) ao realizar a detecção dos padrões, mas mesmo a DPD não possuindo uma total precisão ao detectar os padrões,acreditamos que dentro do contexto dos padrões avaliados conseguimos minimizar essa problemática. Quando a DPD não detecta uma instância de padrão e essa instância possui o termo, a classe relacionada a essa instância é capturada pela ferramenta desenvolvida neste trabalho como uma classe comum com o termo e isso prejudica os resultados, quando a DPD detecta uma instância de padrão que não existe a ferramenta desenvolvida por este trabalho provavelmente captura essa classe, na maioria das vezes como uma classe de instância que não possui o termo, mais uma vez o resultado é prejudicado, contudo, foi possível observar que existem sim termos diferentes daqueles que nomeiam as classes que se relacionam com os padrões de projeto implementados, também foi observado que o uso conjunto dos termos aumenta o valor de indício de padrão, então as possíveis falhas da ferramenta DPD não ameaçam de fato a validade do trabalho.

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentado um estudo de caso que teve como objetivo verificar se existe relação entre os termos do vocabulário de software com os padrões de projeto implementados em um sistema para a análise do uso dos termos do vocabulário no processo de detecção dos padrões de projeto. Através de uma pesquisa bibliográfica sobre o tema foi constatado que os termos do vocabulário que identificam as classes foram utilizados pela ferramenta *DP-Miner* para fazer a distinção entre os padrões de projeto *Strategy, State* e *Bridge*, essa constatação por si só deixa claro que há de fato valor no vocabulário de software para a realização da tarefa de detectar padrões de projeto implementados em um sistema.

Através da análise sobre os vocabulários dos projetos analisados foi constatado também que, além dos termos que nomeiam as classes, termos presentes internamente nas classes também podem possuir relação com os padrões e projeto e com isso serem utilizados no processo de detecção.

Um dos principais problemas ao utilizar os termos do vocabulário no processo de detecção é o fato de que não há uma forte convenção de nomenclatura para implementação dos padrões, devido a isso várias instâncias de padrão detectadas não possuíam os termos em suas classes. Um outro problema ao utilizar os termos do vocabulário no processo de detecção é o fato dos termos serem facilmente encontrados em classes que não fazem parte de nenhuma instância do padrão implementada no sistema.

Para uma análise sobre como melhor utilizar os termos do vocabulário e atribuir maior exclusividade sobre os termos às classes de instância de padrão, foi desenvolvida e utilizada uma ferramenta que por meio de diferentes usos dos termos constatou que a melhor maneira de utilizar o vocabulário é com o uso em conjunto deles, os termos que nomeiam as classes e também aqueles que podem ser encontrados internamente nas classes devem ser utilizados simultaneamente.

O uso simultâneo dos termos apresentou para alguns dos projetos analisados, valores que representam um indício de instância de padrão implementada maiores do que quando usado apenas os termos que nomeiam as classes ou apenas os termos que podem ser encontrados internamente nas classes, isso significa que o uso conjunto dos termos pode atribuir maior exclusividade sobre os termos às classes que compõem instâncias de padrão, por outro lado o não uso de um dos termos pode prejudicar os

resultados que seriam obtidos utilizando apenas o outro, com isso também se conclui que a atribuição de peso à classe de acordo com os termos que ela possui é ainda uma melhor maneira de utilizar os termos do vocabulário.

O vocabulário de software é então uma importante fonte de informações sobre um sistema, que pode também ser utilizada para a detecção de padrões de projeto e poderia ser ainda mais importante nessa tarefa caso houvesse uma convenção de nomenclatura mais forte para a implementação dos padrões. Mas, mesmo que houvesse uma forte convenção de nomenclatura em relação aos padrões de projeto, uma ferramenta que utilizasse apenas o vocabulário no processo de detecção teria seus resultados sempre sujeitos ao bom ou ao mau uso do vocabulário no processo de desenvolvimento do software. Por esse motivo, utilizar apenas do vocabulário no processo de detecção não garantiria uma boa precisão, uma análise estrutural mesmo que mínima, ainda seria necessária.

É possível haver ainda uma ou outra ferramenta de detecção de padrões de projeto que faça uso do vocabulário de software durante o processo de detecção e ainda utilize-se tanto dos termos relacionados ao nome do padrão encontrados nos identificadores que nomeiam as classes como dos termos que podem ser encontrados nas estruturas internas da classe e os utilize em conjunto, mas esse ainda não é o melhor uso dos termos do vocabulário.

As equipes de desenvolvimento e organizações privadas possuem seus próprios padrões de nomenclatura ao desenvolver software e por não haver uma forte convenção de nomenclatura relacionada a implementação dos padrões de projeto, elas utilizam os seus próprios padrões de nomenclatura também ao implementar os padrões de projeto, o que é mais um problema relacionado ao uso dos termos do vocabulário, para utilizar o vocabulário é preciso saber quais os termos utilizados na implementação dos padrões, devido a isso, uma ferramenta que deseje utilizar o vocabulário deve estar apta a receber como *input* quais os termos que devem utilizados no processo, estar apta a receber esse conjunto de termos é uma boa maneira de contornar os problemas relacionados ao padrão de nomenclatura.

Uma maneira de não apenas contornar os problemas relacionados a nomenclatura, mas resolvê-los é a exemplo do trabalho de (FALBO; BERTOLLO, 2005) descrito na seção 2.3, a criação de uma ontologia para os padrões de projeto, a definição dos termos a serem utilizados na implementação de cada padrão de projeto. Essa ontologia ficaria então disponível na web a exemplo do trabalho de (FONTANA et al., 2012) e assim poderia ser facilmente consultada por toda a comunidade e informações sobre novos padrões poderiam ser adicionadas, a criação e aceitação desse ontologia

não apenas contornaria o problema relacionado a convenção de nomenclatura mas resolveria de uma vez por todas os problemas e todos os benefícios do vocabulário de software levantados por inúmeros trabalhos poderiam ser alcançados quando sempre que o assunto for padrões de projeto.

5.1 CONTRIBUIÇÕES

As principais contribuições deste trabalho são:

1. Levantamento bibliográfico sobre padrões de projeto, vocabulário de software e ferramentas de detecção de padrões.
2. Ferramenta de apoio que averigua se termos específicos estão presentes no vocabulário de software.
3. Arquivos em formato XML obtidos por meio de um Estudo de Caso realizado sobre quatro sistemas de software com informações quali-quantitativas a respeito do uso do vocabulário de software no processo de detecção de padrões de projeto implementados em um sistema.
4. Análise sobre a melhor maneira de utilizar os termos do vocabulário de software no processo de detecção de padrões de projeto.
5. Propostas para a solução dos problemas relacionados ao uso dos termos do vocabulário de software no processo de detecção de padrões de projeto.

REFERÊNCIAS

ANTONIOL, G.; GUEHENEUC, Y.-G.; MERLO, E.; TONELLA, P. Mining the lexicon used by programmers during sofware evolution. In: IEEE. **2007 IEEE International Conference on Software Maintenance**. [S.l.], 2007. p. 14–23.

DONG, J.; LAD, D. S.; ZHAO, Y. Dp-miner: Design pattern discovery using matrix. In: IEEE. **14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)**. [S.l.], 2007. p. 371–380.

DONG, J.; ZHAO, Y.; PENG, T. A review of design pattern mining techniques. **International Journal of Software Engineering and Knowledge Engineering**, World Scientific, v. 19, n. 06, p. 823–855, 2009.

FALBO, R. de A.; BERTOLLO, G. Establishing a common vocabulary for software organizations understand software processes. In: **EDOC International Workshop on Vocabularies, Ontologies and Rules for The Enterprise, VORTE**. [S.l.: s.n.], 2005.

FONTANA, F. A.; CARACCIOLO, A.; ZANONI, M. Dpb: A benchmark for design pattern detection tools. In: IEEE. **2012 16th European Conference on Software Maintenance and Reengineering**. [S.l.], 2012. p. 235–244.

GAMMA, E.; JOHNSON, R.; HELM, R.; VLISSIDES, J. **Padrões de Projetos: Soluções Reutilizáveis**. [S.l.]: Bookman, 2000.

GUERROUJ, L. Normalizing source code vocabulary to support program comprehension and software quality. In: IEEE. **2013 35th International Conference on Software Engineering (ICSE)**. [S.l.], 2013. p. 1385–1388.

LAWRIE, D.; FEILD, H.; BINKLEY, D. Quantifying identifier quality: an analysis of trends. **Empirical Software Engineering**, Springer, v. 12, n. 4, p. 359–388, 2007.

MAYVAN, B. B.; RASOOLZADEGAN, A.; YAZDI, Z. G. The state of the art on design patterns: A systematic mapping of the literature. **Journal of Systems and Software**, Elsevier, v. 125, p. 93–118, 2017.

QUINTANS, C.; SANTOS, K.; GUERRERO, D. Avaliação e melhorias de algoritmos para extração de radicais de identificadores codificados em português. 2013. Disponível em: <https://sites.google.com/site/softwarevocabularies/home>.

SANTOS, K. d. F.; GUERRERO, D. D.; FIGUEIREDO, J. C. de. Using developers contributions on software vocabularies to identify experts. In: IEEE. **2015 12th International Conference on Information Technology-New Generations**. [S.l.], 2015. p. 451–456.

SILVA, N. D. Uma abordagem para detectar falsos padrões de projeto. Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas), IFPB (Instituto Federal de Educação, Ciência e Tecnologia da Paraíba), Cajazeiras-PB, Brasil. 2019. Disponível em: <https://github.com/gpes/false-pattern>.

TERRA, R.; MIRANDA, L. F.; VALENTE, M. T.; BIGONHA, R. S. Qualitas. class corpus: A compiled version of the qualitas corpus. **ACM SIGSOFT Software Engineering Notes**, ACM New York, NY, USA, v. 38, n. 5, p. 1–4, 2013.

TSANTALIS, N.; CHATZIGEORGIOU, A.; STEPHANIDES, G.; HALKIDIS, S. T. Design pattern detection using similarity scoring. **IEEE transactions on software engineering**, IEEE, v. 32, n. 11, p. 896–909, 2006.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Cajazeiras
Rua José Antônio da Silva, 300 - Bairro Jardim Oásis, CEP 58.900-000, Cajazeiras (PB)
CNPJ: 10.783.898/0005-07 - Telefone: (83) 3532-4100

Documento Digitalizado Restrito

Versão final do TCC

Assunto: Versão final do TCC
Assinado por: José Sousa
Tipo do Documento: Anexo
Situação: Finalizado
Nível de Acesso: Restrito
Hipótese Legal: Informação Pessoal (Art. 31 da Lei no 12.527/2011)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **José Carlos Rodrigues de Sousa**, **ALUNO (201722010010) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS**, em 30/06/2021 17:57:44.

Este documento foi armazenado no SUAP em 30/06/2021. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 265744
Código de Autenticação: 759c4d4521