Henrique Coelho Mendes

Caracterização da Qualidade Estrutural de Software Java para Inteligência Artificial

Belo Horizonte

2023

Henrique Coelho Mendes

Caracterização da Qualidade Estrutural de Software Java para Inteligência Artificial

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG

Departamento de Computação

Curso de Engenharia da Computação

Orientadora: Kecia Aline Marques Ferreira

Belo Horizonte

2023

Espaço destinado à folha de aprovação

Dedico este trabalho aos meus pais.

Agradecimentos

Agradeço aos professores excepcionais que já tive por terem me ensinado, além do conteúdo, a amar o estudo pelo estudo.

Agradeço a Professora Doutora Kecia pelo direcionamento preciso durante o desenvolvimento deste trabalho.

Por fim, agradeço aqueles que amo pela compreensão e apoio durante a graduação.

"Do. Or do not. There is no try."
(Yoda)

Resumo

Este estudo investiga a qualidade de software em projetos de Inteligência Artificial desenvolvidos em Java, focalizando a construção, manutenibilidade e eficiência. Diante do papel crescente da IA na indústria de software, o trabalho visa avaliar práticas de desenvolvimento, identificar desafios recorrentes e oportunidades de melhoria em repositórios específicos. A metodologia envolve revisão da literatura, seguida pela coleta de dados em 50 repositórios em Java de IA/AM, comparativamente a um grupo Java não relacionado à IA. As conclusões esperadas buscam orientar a comunidade no aprimoramento de soluções de IA em Java, estimulando discussões sobre a melhoria contínua dos padrões de qualidade no desenvolvimento de software voltado à IA.

Palavras-chave: Qualidade de software. Inteligência Artificial. Desenvolvimento em Java. Métricas de software.

Abstract

This study investigates software quality in Java-based Artificial Intelligence projects, focusing on construction, maintainability, and efficiency. Given the growing role of AI in the software industry, the work aims to assess development practices, identify recurring challenges, and pinpoint improvement opportunities in specific repositories. The methodology involves literature review followed by data collection from 50 Java repositories of AI/ML, compared to a non-AI-related Java group. The anticipated conclusions seek to guide the community in enhancing AI solutions in Java, fostering discussions on the continuous improvement of quality standards in AI-oriented software development.

Keywords: Software Quality. Artificial Intelligence. Java Development. Software Metrics.

Lista de abreviaturas e siglas

AM	Aprendizado de Máquina
CBO	Coupling Between Object class
CSV	Comma Separated Values
DIT	Depth of Inheritance Tree
IA	Inteligência Artificial
LCC	Loose Class Cohesion
LCOM	Lack of Cohesion in Methods
LOC	Lines of code
NOC	Number of Children
NOSI	Number of static invocations
RFC	Response For a Class
TCC	Tight Class Cohesion
WMC	Weighted Methods Per Class

Sumário

1 Introdução

A comunidade de Engenharia de Software tem se dedicado intensamente ao estudo da qualidade do software em diversas dimensões, incluindo o enfoque na qualidade estrutural. Entende-se por qualidade estrutural os aspectos relacionados a forma como o software é construído, tais como a sua modularidade e a sua manutenibilidade. No âmbito das investigações sobre a qualidade estrutural do software, destacam-se iniciativas como a proposição de métricas de avaliação do software (KITCHENHAM, 201O) e o desenvolvimento de ferramentas destinadas a aferir sua qualidade estrutural (SOBRINHO; LUCIA; MAIA, 2021). É relevante notar que essas pesquisas tendem a ser independentes do domínio de aplicação, ou seja, concentram-se na avaliação de software genérico. Contudo, recentemente, observa-se um aumento do interesse da comunidade em analisar especificamente a qualidade dos softwares direcionados à Inteligência Artificial (GIRAY, 2021).

No cenário atual da indústria de desenvolvimento de software, a utilização de Inteligência Artificial (IA) tem se destacado como uma poderosa ferramenta para a resolução de problemas complexos e a automatização de tarefas. A linguagem de programação Java, conhecida por sua versatilidade e ampla adoção na comunidade de desenvolvedores, também tem sido empregada de forma extensiva em projetos de IA, sendo a quinta linguagem mais utilizada (GONZALEZ; ZIMMERMANN; NAGAPPAN, 2020). As outras quatro linguagens mais adotadas para o desenvolvimento de software na área de Inteligência Artificial são, nesta ordem: Python, Jupyter Notebook, C++ e JavaScript.

Entretanto, a medida que a IA se consolida como uma ferramenta tecnológica disseminada no mercado de trabalho e no dia a dia da população mundial, a necessidade de garantir a qualidade e confiabilidade do software nesse contexto torna-se crucial. Dessa forma, é importante a realização de estudos que investiguem a qualidade de softwares voltados para a Inteligência Artificial.

Alguns estudos já vem sendo realizados nesse sentido. Masuda et al. (2018) abordaram a urgência na análise da qualidade de softwares de Inteligência Artificial, especialmente em aplicações chamadas de ML-*as-a-services* (MLaaS). Eles identificaram sete principais técnicas de qualidade, incluindo a identificação de falhas, predição com base no comportamento humano e MLaaS. O estudo concluiu que há uma demanda por técnicas de qualidade para diversas aplicações de Aprendizado de Máquina. Já Lenarduzzi et al. (2021) proporam uma trilha de pesquisa para estabelecer padrões de qualidade em software de IA. Eles identificaram problemas como formação de desenvolvedores, falta de guias de qualidade, deficiências no treinamento e escassez de ferramentas. Destacam, também,

desafios no processo de desenvolvimento, carência no controle da qualidade estrutural de software, dependência de versões específicas de bibliotecas e desafios na comunicação entre desenvolvedores. Além disso, afirmam que é crucial melhorar processos, unificar terminologias e criar ferramentas para elevar a qualidade dos softwares de inteligência artificial. Entretanto, esses estudos não aprofundam na análise da qualidade estrutural dos softwares voltados para a Inteligência Artificial. Há estudos que realizam a qualidade estrutural de softwares por meio de métricas, como é o caso do trabalho de Gonçalves (2022), em que foram investigados os valores de referência de 46 métricas de software, a partir da análise de dados de 43 softwares Java de diversos domínios de aplicação. Uma métrica de software é uma medida quantitativa usada para avaliar a qualidade, eficácia e manutenabilidade do software. Ela ajuda no planejamento, estimativas de tempo e custo, e pode ajudar a identificar problemas no desenvolvimento com mais velocidade. Entretanto, investigar a qualidade estrutural de softwares específicos da área de Inteligência Artificial é importante, pois a utilização de softwares dessa natureza tem sido crescente.

1.1 Objetivo

Este trabalho visa realizar uma análise da qualidade estrutural de software em repositórios de Inteligência Artificial na linguagem Java. A análise realizada neste trabalho é baseada em métricas de software. Especificamente, serão investigados os valores de referência de métricas de software em aplicações de Inteligência Artificial, empregando-se o método proposto por Gonçalves (2022). Optou-se por se concentrar na linguagem Java porque o estudo requer a existência de ferramenta apropriada para coleta de métricas de software. Existe uma ferramenta consolidada para isso no caso de Java, o que não ocorre no caso das demais linguagens populares para o desenvolvimento de aplicações em Inteligência Artificial.

1.2 Relevância

Os resultados deste trabalho contribuirão para identificar práticas de desenvolvimento dos projetos presentes nesses repositórios, considerando as características específicas exigidas pelo contexto da IA.

Ao abordar essa temática, pretendemos contribuir para a compreensão da interseção entre o desenvolvimento de software, a linguagem Java e a Inteligência Artificial, com uma ênfase na qualidade dos projetos. Espera-se que as conclusões deste estudo possam orientar a comunidade a tomar decisões mais embasadas e eficazes na concepção, implementação e evolução de soluções de IA em Java. Além disso, a análise proposta pode servir como um ponto de partida para discussões mais amplas sobre a melhoria contínua dos padrões de qualidade na área de desenvolvimento de software orientado a IA.

2 Referencial Teórico

Este capítulo visa a compreensão das metodologias, métricas e ferramentas que desempenham um papel fundamental na avaliação e otimização da qualidade estrutural de software, consolidando assim os alicerces teóricos necessários para o desenvolvimento desta pesquisa.

2.1 Qualidade Estrutural de Software Orientado a Objetos

Os três aspectos centrais da qualidade estrutural de software são: modularidade, coesão e acoplamento (MEYER, 1997). Esta seção descreve esses aspectos.

2.1.1 Modularidade

A modularidade é um conceito em engenharia de software que se refere à organização de um sistema em módulos independentes e bem definidos. Esses módulos são unidades de código que encapsulam funcionalidades específicas e interagem uns com os outros por meio de interfaces definidas com clareza. A modularidade facilita o desenvolvimento, a manutenção e a extensão de um sistema, uma vez que permite que partes do software sejam alteradas sem que afete o restante do sistema. Ela também promove o reúso de código, melhorando a eficiência e a qualidade dos processos de desenvolvimento de software.

2.1.2 Coesão

A coesão é um princípio de design de software que se concentra na relação entre os elementos dentro de um módulo ou classe. Ela mede o grau em que os membros de um módulo estão relacionados e descreve o quão bem as funcionalidades contidas em um módulo estão agrupadas logicamente. Uma alta coesão indica que os membros de um módulo estão altamente relacionados e trabalham juntos para realizar uma tarefa específica, o que é desejável. Por outro lado, baixa coesão significa que os membros de um módulo podem não estar relacionados de forma eficiente, o que pode tornar o código difícil de entender e de alta manutenibilidade.

2.1.3 Acoplamento

O acoplamento é outro princípio de design de software, mas difere da coesão, pois se concentra nas relações entre módulos ou classes. O acoplamento mede o grau de dependência entre diferentes partes de um sistema. Um baixo acoplamento é desejável,

pois indica que as partes do sistema são independentes umas das outras, facilitando a manutenção e a evolução do software. Por outro lado, um alto acoplamento indica uma forte dependência entre os módulos, o que pode tornar o sistema mais rígido e difícil de modificar sem afetar outras partes do código. Portanto, minimizar o acoplamento é um objetivo importante no design de software de qualidade.

2.2 Métricas de Software Orientado a Objetos

Métricas são medidas quantitativas utilizadas para avaliar características específicas de um objeto ou sistema. No contexto da engenharia de software, as métricas desempenham um papel fundamental na avaliação da qualidade, desempenho e manutenibilidade de um software. Elas fornecem uma base objetiva para a análise e tomada de decisões relacionadas ao desenvolvimento, aprimoramento e gerenciamento de software.

Considerando um contexto de softwares orientados a objetos, existem métricas e padrões específicos que se aplicam ao paradigma, direcionando o foco para propriedades como herança, polimorfismo, abstração e encapsulamento.

Essas métricas estão focadas nas características das classes, objetos e suas interações em um sistema orientado a objetos. Chidamber e Kemerer (1994) criaram métricas que ainda hoje são amplamente utilizadas na literatura para avaliar a qualidade de software orientado a objetos. O conjunto inclui seis métricas desenvolvidas especificamente para sistemas que utilizam o paradigma orientado a objetos. Essas métricas são definidas de acordo com critérios estabelecidos por Chidamber e Kemerer (1994), e têm como objetivo fornecer uma base para medir a qualidade do software nesse contexto.

Por serem as principais métricas propostas na literatura para software orientado a objtos, elas são descritas nesta seção.

- Weighted Methods Per Class (WMC): mede a complexidade de uma classe contando o número de métodos em uma classe ponderados por sua complexidade. É uma métrica de coesão, pois está relacionada à complexidade interna de uma classe.
- Number of Children (NOC): representa o número de classes derivadas ou subordinadas a uma classe base. É uma métrica de acoplamento, pois indica o grau de dependência entre a classe base e suas classes derivadas.
- Coupling between object classes (CBO): mede o número de outras classes às quais uma classe está acoplada. É uma métrica de acoplamento, pois avalia o grau de interconexão entre classes.
- Response for a Class (RFC): contabiliza o número de métodos que podem ser invocados por uma classe, incluindo os próprios métodos da classe. É uma métrica

de acoplamento, pois reflete a dependência entre a classe e os métodos que ela pode chamar.

- Lack of Cohesion in Methods (LCOM): mede a falta de coesão interna em uma classe, ou seja, o grau em que os métodos de uma classe estão relacionados entre si. É uma métrica de coesão, pois foca na relação entre os métodos dentro de uma classe.

2.3 Inteligência Artificial e Aprendizado de Máquina

A Inteligência Artificial (IA) é um campo interdisciplinar que busca criar sistemas e máquinas capazes de realizar tarefas que, tradicionalmente, exigiriam a inteligência humana (NORVIG; RUSSELL, 2013). Ao empregar algoritmos avançados e modelos, a IA procura simular processos cognitivos humanos, como aprendizado, raciocínio e resolução de problemas. Essa abordagem permite que as máquinas processem informações complexas, tomem decisões e aprendam com experiências anteriores, reduzindo a dependência de programação explícita para cada tarefa. Em resumo, a IA almeja criar sistemas autônomos que possam imitar e aprimorar habilidades humanas, abrindo portas para aplicações inovadoras em diversas áreas.

O Aprendizado de Máquina (AM), ou *Machine Learning* em inglês, é uma subárea da Inteligência Artificial que se concentra no desenvolvimento de algoritmos e modelos capazes de aprender e melhorar seu desempenho em tarefas específicas a partir da experiência e de dados. Em essência, o AM capacita as máquinas a reconhecer padrões nos dados, fazer previsões e tomar decisões a partir desses padrões, sem a necessidade de programação explícita para cada cenário.

O AM é fundamental para a avaliação de indicadores, tomadas de decisão mais acertivas, personalização de experiências de usuários e avanços em pesquisas científicas de diversas áreas além da computação.

3 Trabalhos relacionados

A investigação sobre a qualidade de softwares para Inteligência Artificial é recente. Este capítulo aborda os principais trabalhos realizados sobre esse assunto. Além disso, descreve também trabalhos relacionados a determinação de valores de referência de métricas de software.

Neste sentido, Masuda et al. (2018) realizaram um questionário considerando que a análise de qualidade de softwares de Inteligência Artificial é uma discussão com caráter de urgência. O trabalho deles, teve como objetivo descobrir técnicas para avaliar e aprimorar a qualidade de aplicações chamadas pelo autor de ML-*as-a-services* (MLaaS), focando em problemas como as formas de se verificar que um sistema retornou dados desconhecidos, se os dados que alimentam o sistema estão insuficientes ou enviesados, a qualidade de sistemas *end-to-end*, entre outros aspectos. Como alvo do estudo, foram escolhidas 16 conferências acadêmicas de Inteligência Artificial e Qualidade de Software e, nelas, questionários foram realizados com representantes de 78 artigos científicos. Como resultado, Masuda et al. (2018) encontrou sete principais técnicas de qualidade utilizadas e bem difundidas atualmente, dentre elas: identificação de falhas em um programa via apuração, predição, baseada na tentativa de prever o comportamento humano perante sistemas de software, buscando aprimorar seu design e, por fim, MLaaS, baseada no desenvolvimento de uma arquitetura de código que cria serviços escaláveis e flexíveis. Para concluir, Masuda et al. (2018) afirmam que existem demandas por técnicas de qualidade de software e/ou testes para verificar os diversos tipos de aplicações de AM.

No contexto do desenvolvimento de softwares de Aprendizado de Máquina e Inteligência Artificial, Gonzalez, Zimmermann e Nagappan (2020) apresentaram um estudo que objetivou avaliar os 10 últimos anos de desenvolvimento do tema. Um estudo empírico foi realizado em 700 ferramentas e 4.524 aplicações de IA e AM, hospedados no GitHub. Além disso, foi feita uma comparação com um grupo de 4.101 repositórios não relacionados ao tema para que se possa analisar e comparar diferentes comunidades. Os repositórios foram categorizados entre aplicações, ferramentas e estudos ou projetos pessoais, sendo estes removidos da análise pois poderiam ser apenas exercícios ou trabalhos universitários. Utilizou-se então um filtro baseado em critérios como tamanho do repositório, popularidade, atividade, disponibilidade de acesso e conteúdo. Após o processamento dos dados, aplicou-se um método quantitativo para mensurar colaboração e autonomia nos repositórios. Os resultados obtidos indicaram que a maioria dos repositórios era de propriedade de usuários (68,25%) e, analisando entre tipos, usuários possuíam a maioria das aplicações de IA e AM (79,1%), enquanto corporações possuíam a maioria das ferramentas (51,43%). Ademais, ao se filtrar a linguagem dos repositórios, encontraram as linguagens mais utilizadas, sendo o

Python em primeiro, Jupyter Notebook em segundo (considerando que Jupyter utiliza as linguagens Julia, Python e R), C++ em terceiro, Javascript em quarto e, por fim, Java em quinto. Tal resultado foi um dos motivadores desta monografia pois, considerando os outros repositórios não relacionados a IA, Java foi identificada como a quinta linguagem mais utilizada e a mesma possui uma comunidade profunda e ativa quanto a qualidade de software. O estudo mostrou, também, que os tópicos mais populares foram *deep learning*, linguagens como Python e ferramentas como Tensorflow, enquanto o conjunto de comparação possuía como tópicos mais comuns linguagens como Javascript e Python, e plataformas baseadas em Android. Por fim, o estudo concluiu que a comunidade de IA e AM no GitHub se difere do desenvolvimento tradicional de software, sendo Python a linguagem mais usada, mesmo com poucas pesquisas acadêmicas focadas na linguagem. Concluíram, também, que os projetos TensorFlow e Tesseract estão extremamente difundidos nas aplicações de IA e AM, sendo eles um referencial para o desenvolvimento da comunidade.

Giray (2021) buscou identificar, analisar, resumir e sintetizar sistematicamente o estado atual da pesquisa em engenharia de software no que se refere ao desenvolvimento de sistemas de AM. Para tal, foi realizada uma revisão sistemática da literatura, na qual um conjunto de 141 estudos provenientes de fontes relacionadas à engenharia de software foi selecionado. Em seguida, uma análise quantitativa e qualitativa foi conduzida com base nos dados extraídos desses estudos. Os resultados obtidos revelam que a natureza não-determinística dos sistemas de AM complexifica todos os aspectos da engenharia de sistemas de AM. Apesar do crescente interesse desde 2018, os resultados apontam que nenhum dos aspectos da ES relacionados ao desenvolvimento de sistemas de AM possui um conjunto maduro de ferramentas e técnicas. É interessante notar que a área de teste se destaca como a mais popular entre os pesquisadores. No entanto, mesmo no contexto de testes de sistemas de AM, os engenheiros dispõem apenas de alguns protótipos de ferramentas e propostas de solução com evidências experimentais limitadas. Muitos dos desafios enfrentados na engenharia de sistemas de AM foram identificados por meio de pesquisas, questionários e entrevistas. Nesse sentido, o artigo indica que há a necessidade de novas pesquisas que busquem, principalmente, identificar e reduzir débitos técnicos em sistemas de aprendizado de máquina.

Lenarduzzi et al. (2021) realizaram um estudo baseado na experiência de quatro grupos de pesquisa na área para definir uma trilha de pesquisa compartilhada que objetiva o desenvolvimento de padrões de qualidade de software para aprendizado de máquina. A partir dessa experiência coletiva, os autores encontraram problemas chaves que definem os problemas atuais no campo de qualidade de software para produtos de Inteligência Artificial. Dentre esses problemas, há a habilidade e o treinamento de desenvolvedores, considerando que a profissão do Engenheiro de AM surgiu há menos de uma década. Os autores identificaram quatro problemas em aberto nesse tópico: a compreensão de código, guias de qualidade, problemas de treinamento e falta de ferramentas de melhoria de

qualidade de software. Além disso, o processo de desenvolvimento ainda é precário, sendo frequentemente complexo introduzir desenvolvedores de IA no processo de desenvolvimento, pois muitos vieram de outras áreas como matemática e estatística e, portanto, não possuem familiaridade com desenvolvimento de software a nível de mercado. Em sequência, o prcesso de teste ainda é um ponto de falha pois não existe nos times de AM uma cultura de testes unitários, de integração e outros tipos de testes comumente utilizados na engenharia de software. Dessa forma, a maioria dos testes é realizado de forma manual, ou nem são realizados. Considerando este fato, a confiabilidade para realizar *deployments* é reduzida. Há também, carência no controle da qualidade estrutural de softwares de IA, que frequentemente não é realizado, sendo causado principalmente por falta de ferramentas que indiquem possíveis problemas estruturais em tempo de desenvolvimento. Ademais, a falta de compatibilidade de versões mais recentes para mais antigas de ferramentas e a portabilidade dos códigos faz com que os códigos se tornem muito dependentes de versões específicas de bibliotecas, o que gera códigos obsoletos e de difícil compreensão à medida que as tecnologias evoluem e as aplicações se mantém presas a versões antigas. Por fim, as terminologias e a comunicação entre desenvolvedores podem ser aprimoradas, considerando que o mercado está em constante evolução e não se têm uma unificação de termos. Lenarduzzi et al. (2021), portanto, concluem que existem diversos problemas a serem solucionados para aumentar a qualidade geral de softwares de Inteligência artificial e, para tanto, é preciso melhorar processos, unificar nomenclaturas e, principalmente, criar ferramentas que permitam analisar e indicar para os desenvolvedores possíveis problemas durante o desenvolvimento das soluções.

Gonçalves (2022) abordou a complexidade resultante do processo evolutivo dos sistemas de software, destacando a necessidade de correções e aprimoramentos contínuos. O trabalho propõe uma análise de 46 métricas de software orientado a objetos, diferenciando-se ao desenvolver uma metodologia adaptada aos padrões atuais, em contraposição a abordagens anteriores baseadas em versões mais antigas de software. Em três etapas, a pesquisa realizou uma revisão bibliográfica, derivou e avaliou os intervalos das métricas, constatando que cerca de 70% estavam associados a classes com erros em intervalos considerados "Ruins". Além de contribuir com dados e scripts disponíveis publicamente, as conclusões apontam para oportunidades de pesquisa, destacando a relevância desses avanços na compreensão e enfrentamento dos desafios da engenharia de software em ambientes dinâmicos.

4 Metodologia

Este capítulo define a metodologia empregada nesta pesquisa, estruturando-a em seções que representam distintas fases do estudo. A Seção 4.1 discute a condução do estudo teórico relativo a esta pesquisa. A Seção 4.2 detalha o procedimento adotado para a seleção dos sistemas de software de IA analisados. A Seção 4.3 detalha como os repositórios que serão analisados foram escolhidos. A Seção 4.4 descreve como será realizada a coleta de métricas e a análise dos resultados. A Seção 4.5 descreve qual método utilizado para analisar os resultados obtidos.

4.1 Revisão da Literatura

Essa monografia possui como temáticas principais a qualidade de software e os sistemas de Inteligência Artificial e/ou Aprendizado de Máquina. Para obter um recorte referente à interseção entre os dois temas, um estudo prévio de artigos e livros da literatura foi realizado, objetivando determinar o estado da arte da área em questão.

4.2 Coleta de Dados

Os dados coletados dos repositórios Java de IA/ML seguiram a metodologia delineada no artigo (GONZALEZ; ZIMMERMANN; NAGAPPAN, 2020). Entretanto, uma limitação encontrada foi a ausência de uma lista específica dos repositórios coletados, bem como das tags do GitHub utilizadas para identificá-los. Para contornar essa lacuna, desenvolvemos um script em Python e Jupyter para efetuar a coleta necessária. Utilizando a API do GitHub, os dados dos repositórios desejados foram adquiridos por meio de uma query que inicialmente filtrava apenas códigos em Java. Posteriormente, foram realizadas requisições para cada tag, sendo "machine learning"e "Artificial Intelligence"as escolhidas. Uma lista foi empregada para armazenar os repositórios encontrados, permitindo a inclusão apenas de dados referentes a repositórios únicos, evitando duplicações. Por fim, os resultados foram registrados em um arquivo JSON.

No total, 80 repositórios primordiais foram selecionados para análise manual, classificados em ordem decrescente de estrelas e, em caso de empate, pela quantidade de commits.

4.3 Escolha dos Repositórios

A escolha dos repositórios a serem analisados visou estabelecer comparações com um grupo de repositórios Java que não necessariamente abordam projetos de IA. Nesse contexto, utilizamos como referência o trabalho de Diego Santos Gonçalves, que analisou 46 repositórios. Com base nessa análise, optamos por selecionar uma quantidade similar de repositórios para este estudo.

A partir da lista obtida durante a coleta de dados, os repositórios foram filtrados manualmente para incluir apenas sistemas que verdadeiramente abordassem o tema de Inteligência Artificial, utilizando Java como parte integrante da solução. Repositórios em que o Java era utilizado apenas como servidor para serviços de ML, estudos ou replicação de cursos, bem como repositórios destinados à coleta e persistência de dados para posterior uso em modelos, foram excluídos. Esse processo resultou em uma lista final de 50 repositórios.

4.4 Coleta de Métricas

Para a coleta de métricas, todos os repositórios escolhidos serão baixados em sua versão mais recente. Em seguida, a ferramenta *CK Tool* será utilizada para coletar métricas. Tal ferramenta foi escolhida pois foi a mesma utilizada no trabalho do Gonçalves (2022).

4.5 Análise dos Resultados

A partir das métricas de cada repositório e, seguindo o método proposto no trabalho do Gonçalves (2022), os resultados obtido serão analisados e discutidos.

5 Resultados Preliminares

Como resultado preliminar deste trabalho, foram selecionados os principais repositórios no GitHub dedicados à inteligência artificial desenvolvidos em Java, totalizando 50 repositórios. As características desses repositórios indicam que há uma diversidade de abordagens e ferramentas utilizadas na comunidade de desenvolvimento de softwares voltados para IA.

A Tabela 1 mostra os repositórios que serão analisados na sequência deste trabalho.

Tabela 1 - Repositórios coletados

Id Github	Título	Descrição
14734876	deeplearning4j	Conjunto de ferramentas para implantação e treinamento de modelos de aprendizagem profunda usando JVM.
89322848	angel	Servidor de parâmetros flexível e poderoso para aprendizado de máquina em larga escala.
26921116	smile	Motor de aprendizado estatístico.
60377070	vespa	IA e Dados online.
218396611	djl	*Framework* de aprendizado profundo em Java, independente de mecanismo.
150938406	alink	Plataforma de algoritmo de aprendizado de máquina baseada no Flink.
48880766	tablesaw	Biblioteca Java para *dataframe* e visualização.
1370858	incubator-kie-optaplanner	Solucionador de restrições de IA em Java para otimizar problemas de roteamento, escalas, atribuições e programação.
5797013	grobid	Software de aprendizado de máquina para extrair informações de documentos acadêmicos.
41310523	deeplearning4j-examples	Exemplos de Deeplearning4j (DL4J, DL4J Spark, DataVec).
213321771	alldata	Produto de dados personalizável com plataforma, mercado, aprendizado de máquina e aplicação de produtos.

87506995	EasyML	Sistema de fluxo de dados para facilitar o uso de algoritmos de aprendizado de máquina em tarefas do mundo real.
22269384	oryx	Arquitetura lambda no Apache Spark, Apache Kafka para aprendizado de máquina em grande escala em tempo real.
71305435	modeldb	Versionamento de modelo de aprendizado de máquina de código aberto, gerenciamento de experimentos.
30649762	seldon-server	Plataforma de aprendizado de máquina e mecanismo de recomendação construído em Kubernetes.
77308564	elasticsearch-learning-to-rank	*Plugin* para integrar aprendizado para classificação com Elasticsearch.
84030164	AndroidTensorFlowMachineLearningExample	Exemplo de aprendizado de máquina TensorFlow para Android.
272672735	tribuo	Biblioteca de aprendizado de máquina em Java.
23766587	gdx-ai	Framework de IA para jogos baseado em libGDX.
25405542	datumbox-framework	Framework de aprendizado de máquina de código aberto escrito em Java.
682834027	jvector	Motor de busca de vetor incorporado mais avançado.
22690513	data-algorithms-book	MapReduce, Spark, Java e Scala para Data Algorithms Book.
147803317	TornadoVM	Estrutura prática e eficiente para programação heterogênea em linguagens gerenciadas.
142410331	hopsworks	Plataforma de IA intensiva em dados com Armazém de Recursos.
66772740	qupath	Análise de bioimagem e patologia digital.
100483465	kafka-streams-machine-learning-examples	Exemplos de implantação de modelos analíticos em ambientes de produção escaláveis com Apache Kafka.
15365897	jenetics	Algoritmo genético, programação genética, evolução e otimização multiobjetivo em Java.
32772468	JSAT	Biblioteca de aprendizado de máquina para análise estatística em Java.
31661940	elki	Kit de ferramentas de mineração de dados ELKI.
353992692	MLKit	Caixa de ferramentas para reconhecimento de texto, código de barras, marcação de imagem, detecção facial, dentre outros.

125595640	Android-TensorFlow-Lite-Example	Exemplo de aprendizado de máquina TensorFlow Lite para Android.
124719143	TensorFlowAndroidDemo	Demonstração TensorFlow Android para reconhecimento de imagem e outros recursos.
148223447	TonY	Framework para executar nativamente frameworks de aprendizado profundo no Apache Hadoop.
209459144	submarine	Plataforma de aprendizado de máquina nativa para a nuvem.
194897	jblas	Álgebra Linear para Java.
19367053	moa	*Framework* de código aberto para mineração de *Big Data* em fluxo. Inclui algoritmos de aprendizado de máquina e ferramentas de avaliação.
43274159	botlibre	Plataforma aberta para inteligência artificial, *chat bots*, agentes virtuais e automação em redes sociais.
84253677	knime-core	Plataforma de análise KNIME.
85210314	android-tensorflow-mnist	Exemplo TensorFlow para Android (construção de modelo com TensorFlow para Android).
13365821	ojalgo	Biblioteca Java de código aberto para matemática, álgebra linear e otimização.
620363420	timefold-solver	Otimizador de problemas de roteamento de veículos, alocação de funcionários, atribuição de tarefas, programação de manutenção e outros problemas de planejamento.
25651414	blinkid-android	Digitalização de ID impulsionada por IA à aplicativos nativos Android.
249609964	dl-inference	Ferramenta de inferência de aprendizado profundo.
69562331	artemis	Aprendizado interativo com *feedback* automatizado.
65136727	onyx	Biblioteca Android que utiliza tecnologias como Inteligência Artificial, aprendizado de máquina e *Deep Learning* para ajudar os desenvolvedores a entenderem o conteúdo que estão exibindo em aplicativos.
91575491	ytk-learn	Biblioteca de aprendizado de máquina distribuída que implementa a maioria dos algoritmos populares de AM.

265994353	hms-ml-demo	Demonstração do HMS ML que mostra como integrar serviços fornecidos pelo ML Kit, como detecção facial, reconhecimento de texto, segmentação de imagem, ASR e TTS.
56810670	neo4j-nlp	Capacidades de PNL (Processamento de Linguagem Natural) no Neo4j.
45342258	xgboost-predictor-java	Implementação pura em Java do preditor XGBoost para tarefas de previsão online.
75713433	tgboost	Pequena árvore de impulso de gradiente.

Fonte: Produzido pelos autores.

6 Conclusão

O desenvolvimento desta etapa do trabalho proporcionou uma estrutura sólida para a compreensão da qualidade de software em projetos que envolvem Inteligência Artificial e Aprendizado de Máquina implementados em Java. A revisão da literatura permitiu um aprofundamento nas temáticas centrais, delineando o estado da arte na interseção entre qualidade de software e sistemas de IA/AM. A coleta de dados foi realizada por meio do desenvolvimento de um script personalizado. A escolha dos repositórios, alinhada ao estudo prévio de Gonçalves (2022), resultou em uma lista de 50 repositórios Java, abordando genuinamente a temática de Inteligência Artificial. Este recorte reflete a intenção de estabelecer comparações significativas com repositórios Java não relacionados a IA. Com os repositórios identificados, a próxima etapa do trabalho englobará a coleta de métricas e a análise detalhada dos resultados, consolidando uma visão abrangente sobre a qualidade estrutural desses sistemas.

Por fim, a figura 1 mostra o cronograma para a realização deste trabalho, sendo possível observar que as atividades estão seguindo os prazos designados no pré-projeto.

Figura 1 – Cronograma

Atividades	Meses (Agosto/2023 - Julho/2024)										
	TCC I					TCC II					
	Ago	Set	Out	Nov	Dez	Fev	Mar	Abr	Mai	Jun	Jul
Definição do tema											
Entrega do Pré-projeto											
Estudo do referencial teórico											
Estudar sobre os parâmetros utilizados para a análise de qualidade de software											
Definir os parâmetros de análise											
Levantar os repositórios relacionados ao tema											
Definir os projetos a serem analisados											
Implementar scripts para coletar os dados dos projetos											
Entrega do TCC I											
Coletar os dados dos projetos											
Pesquisar ferramentas de análise de qualidade de software											
Executar as ferramentas necessárias nos projetos escolhidos											
Analisar os resultados obtidos											
Conclusão											
Elaboração da apresentação											
Redação do TCC											
Entrega do TCC II											
Defesa junto à banca											

Referências

CHIDAMBER, S.; KEMERER, C. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, v. 20, n. 6, p. 476–493, 1994.

GIRAY, G. A software engineering perspective on engineering machine learning systems: State of the art and challenges. *Journal of Systems and Software*, v. 180, p. 111031, 2021. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S016412122100128X>.

GONZALEZ, D.; ZIMMERMANN, T.; NAGAPPAN, N. The state of the ml-universe: 10 years of artificial intelligence machine learning software development on github. Association for Computing Machinery, New York, NY, USA, p. 431–442, 2020. Disponível em: <https://doi.org/10.1145/3379597.3387473>.

GONçALVES, D. S. A relação entre o surgimento de code smells e a evolução do software. 2022.

KITCHENHAM, B. What's up with software metrics?–a preliminary mapping study. Elsevier, v. 83, n. 1, p. 37–51, 201O.

LENARDUZZI, V. et al. Software quality for ai: Where we are now? In: _____. *Software Quality: Future Perspectives on Software Engineering Quality*. Cham: Springer International Publishing, 2021. p. 43–53. ISBN 978-3-030-65854-0.

MASUDA, S. et al. A survey of software quality for machine learning applications. *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, p. 279–284, 2018. Disponível em: <https://api.semanticscholar.org/CorpusID:49892619>.

MEYER, B. *Object-Oriented Software Construction*. 2. ed. Upper Saddle River, NJ: Prentice Hall, 1997. ISBN 978-0-13-629155-8.

NORVIG, P.; RUSSELL, S. *Inteligência Artificial*. [S.l.]: Elsevier Editora Ltda, 2013.

SOBRINHO, E. V. d. P.; LUCIA, A. D.; MAIA, M. d. A. A systematic literature review on bad smells-5 w's: Which, when,what,who,where. *IEEE Transactions on Software Engineering*, v. 47, n. 1, p. 17–66, 2021.