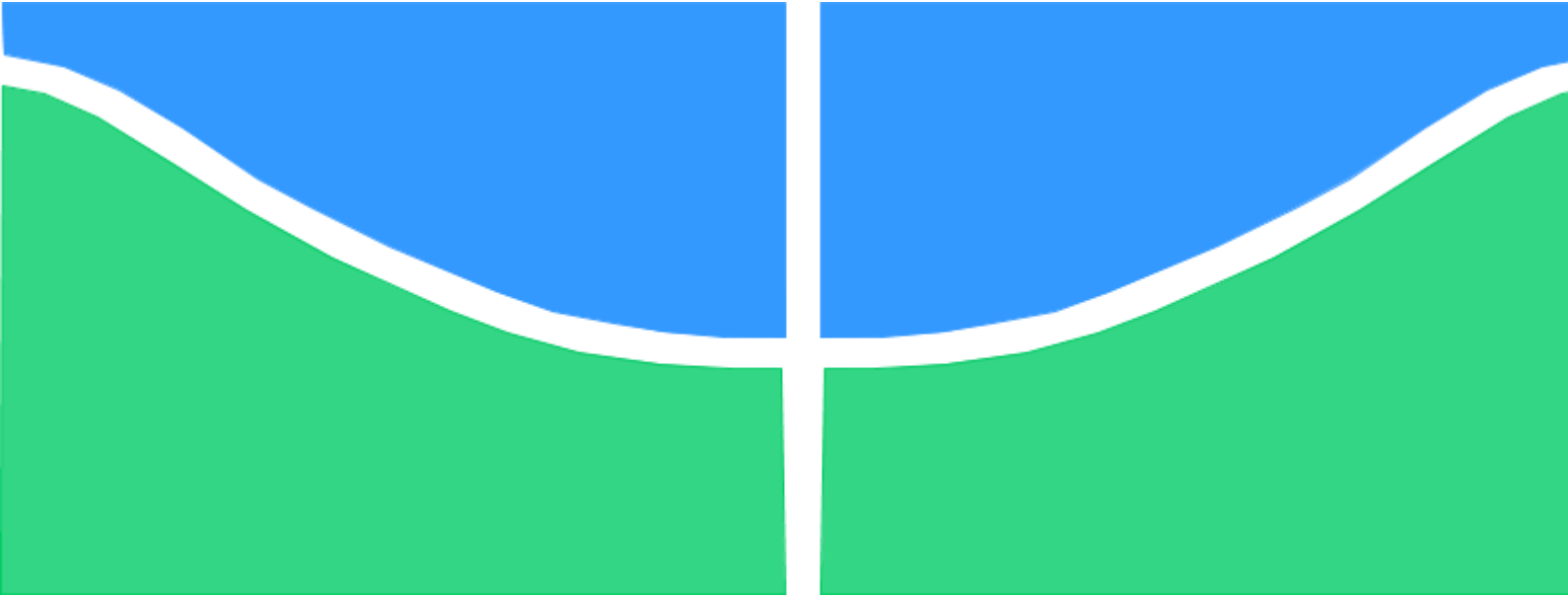


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Trabalho Extra 2
Grupo 3

Relatório

Alexandre Miguel Rodrigues Nunes Pereira - 16/0000840
Daniel Maike Mendes Gonçalves - 16/0117003
Marco Antonio de Lima Costa - 16/0135681
João Vitor Ferreira Alves - 16/0127912
Pedro Rodrigues Pereira - 17/0062686
Renan Welz Schadt - 16/0143403
Rômulo Vinícius de Souza - 15/0147601
Shayane Marques Alcântara - 16/0144949

Brasília - DF

Trabalho Extra 2 - Relatório

Grupo 3

Tabela de Contribuição

Integrantes	**Contribuição**
Alexandre Miguel Rodrigues Nunes Pereira	100
Daniel Maike Mendes Gonçalves	100
Marco Antonio de Lima Costa	100
João Vitor Ferreira Alves	100
Pedro Rodrigues Pereira	100
Renan Welz Schadt	100
Rômulo Vinícius de Souza	100
Shayane Marques Alcântara	100

SUMÁRIO

Introdução

Na atual conjuntura comercial que estamos vivenciando, é cada vez mais presente a necessidade de evolução e adaptação às novas demandas do mercado. Aliado a este fato, a qualidade de software afirma-se como diferencial em empresas que se preocupam com o produto final.

Diante de diversos atributos de qualidade previstos na norma ISO 9126-1 (2003), como a Funcionalidade, Confiabilidade, Usabilidade, Eficiência, Manutenibilidade e a Portabilidade, é possível afirmar que todos os atributos citados têm seu grau de importância em um produto de software, de acordo com o objetivo dos *stakeholders*. Portanto, é foco deste trabalho dar ênfase ao ramo de manutenibilidade, especificamente a modificabilidade, que trata da capacidade do produto de software permitir que modificações possam ser implementadas, não somente no código fonte, mas também na documentação e projeto (ISO 9126, 2003).

Um produto com alta qualidade neste quesito pode vir a ser lido e compreendido mais facilmente, tornando o processo de manutenção mais eficiente e com menos custos (LAGERSTRÖM, 2010). Ao contrário, um produto que dificulte esta ação, poderá ter a tendência da necessidade de mais tempo e esforço de profissionais para cumprir o objetivo das modificações demandadas.

Objetivo

O trabalho tem como objetivo definir métricas e meios de testar a manutenibilidade de um código, com ênfase na sub-característica de modificabilidade, de modo a exemplificar essas definições por meio de um estudo de caso. Assim, o problema a ser resolvido é definir como se deve testar uma característica não funcional, como a manutenibilidade e quais métricas utilizar para verificar se o código está ou não manutenível.

Objetivos específicos

Dado o contexto do presente trabalho, bem como seu objetivo geral, foram definidos objetivos específicos que compõem a estratégia argumentativa e estrutura de tópicos, demonstrando quais serão as abordagens realizadas e as compreensões necessárias para o alcance do objetivo geral, sendo esses:

- Apresentar a problemática de testar a manutenibilidade de um software;
- Definir os conceitos de Manutenibilidade e Modificabilidade;
- Apresentar a metodologia de pesquisa utilizada para escopo do projeto;
- Definir as métricas que quantificam dados de manutenibilidade e modificabilidade.
- Relacionar as métricas definidas com as estratégias de validação de Manutenibilidade;
- Exemplificar as métricas através de um estudo de caso;

Metodologia

Para a estruturação do presente trabalho, foi realizada uma revisão bibliográfica a respeito da manutenibilidade e modificabilidade, em que foram consultados artigos, livros e periódicos, para correta conceituação dessas características, bem como para elaboração das estratégias de testes que essa área demanda. Em complemento à etapa de revisão bibliográfica, foi selecionado um projeto da disciplina de MDS/EPS para aplicação e exemplificação das métricas, permitindo observar sua aplicação prática com dados e resultados reais, bem como sua interpretação para o tema proposto.

Os seguintes conceitos foram necessários para o embasamento teórico e compreensão dos tópicos abordados nas etapas seguintes do relatório. Para esse fim, foram consultados materiais no Portal de periódicos Capes[1] sob as seguintes *strings* de pesquisa:

- "test*" AND "modifiability" AND "software"
- "software modifiability" AND "software maintainability"
- ("software" AND ("modifiability" OR "maintainability")) AND "test*"

A partir desta pesquisa, artigos aparentemente relacionados com o tema foram selecionados para a realização de fichamentos conforme os critérios descritos no ***apêndice A***, para embasar o trabalho proposto.

Definições

As seguintes definições conceituam aspectos e característica presentes ao longo do projeto, utilizando esses conceitos para embasamento das métricas, estratégias de testes análise do caso de estudo, sendo esses conceitos:

Manutenibilidade

Para definir manutenibilidade, é necessário, primeiramente, entender o conceito de qualidade de software como definição baseada em manufatura e produto, que, segundo a ISO 9000 (2000), pode ser definida como a totalidade de funcionalidades e características de um produto ou serviço, que se apoia em sua habilidade de satisfazer necessidades explícitas ou implícitas. Assim, partindo dessa visão de qualidade e entendendo manutenção de um software como: Processo de modificação que busca manter as funcionalidades primárias intactas, em prol de repará-lo ou atualizá-lo, pode-se conceituar a manutenibilidade como a facilidade com que um sistema ou componente de software pode ser modificado para corrigir falhas, melhorar a performance, se adaptar a um novo ambiente ou às mudanças estabelecidas nos requisitos, de forma que o desempenho dessas atividades seja de custo mais baixo possível.

[1] Portal de periódicos Capes: http://www-periodicos-capes-gov-br.ez54.periodicos.capes.gov.br/

Com esse conceito, pode-se inferir que a manutenibilidade é uma característica relevante para garantir a qualidade de software, de forma que as ações de manutenção possuam diferentes caráteres de acordo com a intencionalidade e a motivação com que são realizadas, sendo essas: ***Corretivas***, que são atividades realizadas após a disponibilização do sistema para uso, se referindo à remoção de defeitos existentes, visando ser pontuais e atômicas; ***Adaptativas***, que podem ser entendidas como atividades que se preocupam em ajustar o software às mudanças no ambiente de execução, normalmente realizadas para se adaptar a um novo produto ou para expansão de mercado; ***Perfectivas***, que são atividades de rotina, buscando melhorar o funcionamento e a qualidade de um dado sistema (NAIK, TRIPATHY, 2008).

Modificabilidade

De acordo com a norma ISO 9126-1 (2003), a modificabilidade é a capacidade do produto de software permitir que uma modificação especificada seja implementada. A modificabilidade apresenta-se como uma sub-característica da manutenibilidade, ou seja, ela pode vir a influenciar a característica de qualidade e pode ser então analisada por meio da implementação de métricas externas e internas.

Como a sub-característica consiste em modificações, essas podem vir a incluir ações como acrescentar, deletar, adaptar ou reestruturar o código ou documentação desejados. Os esforços dessas modificações podem variar de pequenas mudanças de funcionalidades a implementação de novos serviços (LAGERSTRÖM, et. al, 2010). Além disso, alguns fatores são comumentes abordados em relação à modificabilidade de softwares, como a legibilidade, consistindo na avaliação da complexidade, da clareza, nomes de variáveis, a estrutura do código, comentários e até indentação (JØRGENSEN, 1980).

Para auxiliar na coleta dessas medidas, existem atualmente diversas ferramentas com o objetivo de análise de código, como o Code Climate[2], em sua forma automatizada mais comumente utilizada em repositórios de código aberto, em plataformas de compartilhamento e versionamento de código, como Github e Gitlab, como também a CLOC[3] e a Radon[4], ferramentas com o mesmo fim. Em suma, as ferramentas citadas utilizam métricas para contabilizar e analisar as medidas levantadas por (JØRGENSEN, 1980).

Métricas

Tendo por base os conceitos da característica em foco ─ Manutenibilidade ─ e a sub-característica abordada ─ Modificabilidade ─ pode-se identificar métricas que promovam a identificação dessas características, conforme definidas no material consultado, sendo elas:

[2] Code climate: https://docs.codeclimate.com/docs/maintainability
[3] CLOC: http://cloc.sourceforge.net/
[4] Radon: https://radon.readthedocs.io/en/latest/intro.html

Lines of Code (LOC)

Linhas de código (em inglês, *Lines of Code*), também referida como *Source Lines of Code (SLOC)* é uma métrica utilizada como indicativo do tamanho de um software, através da contagem do número de linhas no código fonte do programa. Ainda que seja uma medida baseada na ordem de magnitude de software, é um bom indicativo acerca do quão manutenível um software está, bem como quão alta é sua complexidade.

Existem dois tipos de medição a partir dessa métrica que podem ser aplicados a um projeto, sendo elas: *Physical Lines of Code (LOC,* de forma similar*)*, que abrange de forma mais geral a quantidade efetiva de linhas de código, incluindo até mesmo comentários e, a depender da proporção definida, a quantidade de linhas brancas em um código, como por exemplo, inclusão de linhas brancas até que essas atinjam a cobertura de 25% de uma seção de código; *Logical Lines of Code (LLOC)*, que corresponde ao número de definições realizadas em um código, fator que independe da formatação ou indentação utilizada, mas que é variável de acordo com a linguagem e de mais difícil implementação. Dessa forma, a ótica LLOC de se definir a métrica pode ser tomada por possuir resultados que não estão sujeitos a estilo, e buscam verificar a complexidade em termos de lógicas implementadas.

Complexidade Ciclomática

Complexidade ciclomática é a métrica que indica o número de possíveis decisões contidas em um bloco de código, isto é, a quantidade de caminhos independentes que esse bloco contém e que podem ser percorridos. Essa métrica pode ser calculada em relação a funções, módulos, métodos ou classes em um programa, obtendo-se um valor numérico que indica a quantidade de fluxos do bloco, utilizando para isso indicadores de *loops*, estruturas condicionais ou atribuições nativas do código.

Assim, ao se identificar os possíveis caminhos de uma seção de código, é necessário montar casos de testes que contemplem essas possibilidades, garantindo a cobertura de código de um determinado sistema. Dessa forma, quanto maior a complexidade ciclomática, menos modificável e manutenível se torna o software, dado que alterações realizadas podem cascatear pelos caminhos, gerando erros, exceções, ou resultados inválidos conforme o escopo do projeto.

Halstead

As medidas de complexidade de Halstead são métricas embasadas no estudo das propriedades mensuráveis de um software de forma generalizada, isto é, a partir do código fonte, utilizar elementos comuns aos códigos em geral - como operadores e operandos - para, através de sua quantificação, encontrar indicativos de complexidade por meio de fórmulas e conceitos da linguagem. Dentre as métricas selecionadas, as medidas de Halstead não possuem uma identificação direta com relação aos aspectos de modificabilidade e

manutenibilidade em um código, performando uma análise generalista a respeito da linguagem, especificidades do escopo e de diferenças entre os módulos do sistema.

Assim, os valores obtidos através da aplicação da métrica baseada nessas medidas segue a seguinte estrutura:

$$n_1 = \textit{Número de operadores distintos}$$
$$n_2 = \textit{Número de operandos distintos}$$
$$N_1 = \textit{Número total de operadores}$$
$$N_2 = \textit{Número total de operandos}$$

Tabela 1 - Equações de medidas de Halstead

Variável	Significado	Equação
n	Vocabulário	$n = n_1 + n_2$
V	Volume	$V = N_1 + N_2 * log_2 n$
D	Dificuldade	$D = \frac{n_1}{2} * \frac{N_2}{n_2}$
E	Esforço	E = D * V
B	Erros (Bugs)	B = V / 3000

Índice de Manutenção

Esta métrica busca, por meio de uma equação envolvendo os resultados obtidos nas três métricas anteriores - Linhas de Código, Complexidade Ciclomática, Halstead - definir um valor entre 0 e 100 para qualificar o grau de manutenção de um software, isto é, o quão fácil é para oferecer suporte ao sistema e realizar alterações no código fonte. Dessa forma, essa métrica não possui variantes de interpretação quanto ao seu valor final, já que fornece essa pontuação que permite classificar, diretamente, a Manutenibilidade de um software, possuindo, entretanto, variações na estrutura da equação utilizada. O Fórmula original segue a seguinte estrutura:

$$MI = 171 - 5.2 ln\, V - 0.23\, G - 16.2\, ln\, L \quad (1)$$

Entre as variações em sua estrutura, o sistema Radon - utilizado para testar o software presente no estudo de caso - estrutura a equação da seguinte forma:

$$MI = max[0,\ 100 * \frac{171 - 5.2\ ln\ V - 0.23G - 16.2\ ln\ L + 50\ sin(\sqrt{2.4C})}{171}] \quad (2)$$

Em ambas as equações, os valores utilizados equivalem a:

- **V** - A medida de *volume* da métrica ***Halstead***
- **G -** Valor total da Complexidade Ciclomática
- **L -** Número de Linhas de Código (SLOC)
- **C -** Percentual de linhas de comentários (convertida para radianos)

Testes

Com o conceito das métricas bem definido, fez-se necessário a busca por estratégias ou casos de testes que pudessem validar essas métricas, bem como aplicá-las para qualquer software de maneira generalista. Entretanto, o aspecto sob qual esse testes foram pesquisados possui um caráter subjetivo, sem que testes automatizados de forma direta fossem definidos, de acordo com o referencial teórico utilizado. Assim, foram estipuladas mecânicas que, com o auxílio de ferramentas que realizam a obtenção dos dados das métricas, permitem a identificação dos aspectos de modificabilidade e manutenibilidade, bem como sua validação.

Mecânica

Para realizar o teste sob a visão da manutenibilidade de um sistema, se faz necessário a definição e aferição das métricas de código LOC, Complexidade Ciclomática e Halstead. A métrica final que define o índice de manutenibilidade utiliza as outras três métricas acima para retornar o quão manutenível é cada arquivo de um projeto.

Ferramentas

A ferramenta utilizada para obter essas métricas foi o Radon, que analisa o código fonte sob a ótica das métricas descritas acima, inserindo-as em uma equação para retornar o índice de manutenibilidade.

O Radon analisa a árvore de execução de um programa em Python para avaliar sua complexidade ciclomática, de modo que cada estrutura condicional ou de laço contribui para o aumento da complexidade do código. Quanto maior a complexidade ciclomática, pior é a manutenibilidade de um código e mais difícil é modificá-lo.

Tabela 2 - Classificação de Blocos de Código em Complexidade Ciclomático, Radon

Pontuação	Classificação	Risco

1 - 5	A	Baixo - Bloco Simples
6 - 10	B	Baixo - Bloco bem estruturado e estável
11 - 20	C	Moderado - Bloco levemente complexo
21 - 30	D	Mais que Moderado - Bloco de complexidade relevante
31 - 40	E	Alto - Bloco Complexo, Alarmante
41+	F	Muito Alto - Propenso a erro, Bloco Instável

Para calcular o índice final, são usadas também as métricas de linhas totais de código fonte e porcentagem de linhas de comentário (em relação ao total de linhas do projeto). Quanto mais linhas de código um arquivo contém, menos manutenível ele é. Por outro lado, linhas de comentário ajudam na compreensão do código.

O Radon coleta todas as métricas de Halstead, dentre elas a métrica utilizada para o cálculo do índice final é o volume, que é definido a partir do número de operadores e operandos manipulados pelo código. Quanto maior o volume de um código (manipula mais operadores) pior tende a ser sua manutenção. O valor do volume depende também da linguagem utilizada, linguagens mais verbosas costumam ter valores mais altos, mas isso não indica necessariamente que sua manutenibilidade é inferior, por essa razão esta métrica somente deve ser usada em conjunto com outras, ou como critério de comparação entre dois arquivos escritos na mesma linguagem (THIRUMALAI, CHANDRASEGAR, 2017).

O índice de manutenção é obtido através das métricas mencionadas acima e usa uma escala de 1 a 100. É considerado que o código possui boa manutenibilidade quando este índice está acima de 20.

Estudo de Caso

Para o estudo de caso realizado neste trabalho, foi escolhido o projeto HubCare[5], desenvolvido na disciplina de EPS/MDS durante o primeiro semestre de 2019. O HubCare consiste em uma extensão de navegador para Chrome e Firefox, a qual desempenha o papel de analisar repositórios do Github para extrair métricas que possam definir um *ranking* de contribuição para os projetos, ou seja, o quanto um projeto está apto para receber contribuições, levando em conta frequências de *commits*, solução de *issues*, quantidade de autores e *pull requests*.

Após a escolha do projeto, foi necessário fazer uso das ferramentas escolhidas, no caso a CLOC e a Radon. As análises ocorreram com sucesso e os resultados estão listados a seguir:

[5] Repositório HubCare: https://fga-eps-mds.github.io/2019.1-hubcare-docs/

- CLOC, ao fazer uso da ferramenta CLOC, *count of lines*, foi usado o seguinte comando:
    - *cloc 2019.1-hubcare-api*
      (ferramenta) (diretório a ser analisado)

  Obtendo assim o resultado:

```
     173 text files.
     157 unique files.
      59 files ignored.

github.com/AlDanial/cloc v 1.84  T=0.08 s (1834.7 files/s, 70794.3 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Python                         132            808           1009           2920
YAML                             3             39             37            430
Markdown                         9            124              0            276
make                             5             30              5             96
Dockerfile                       1              0              0             14
-------------------------------------------------------------------------------
SUM:                           150           1001           1051           3736
-------------------------------------------------------------------------------
```

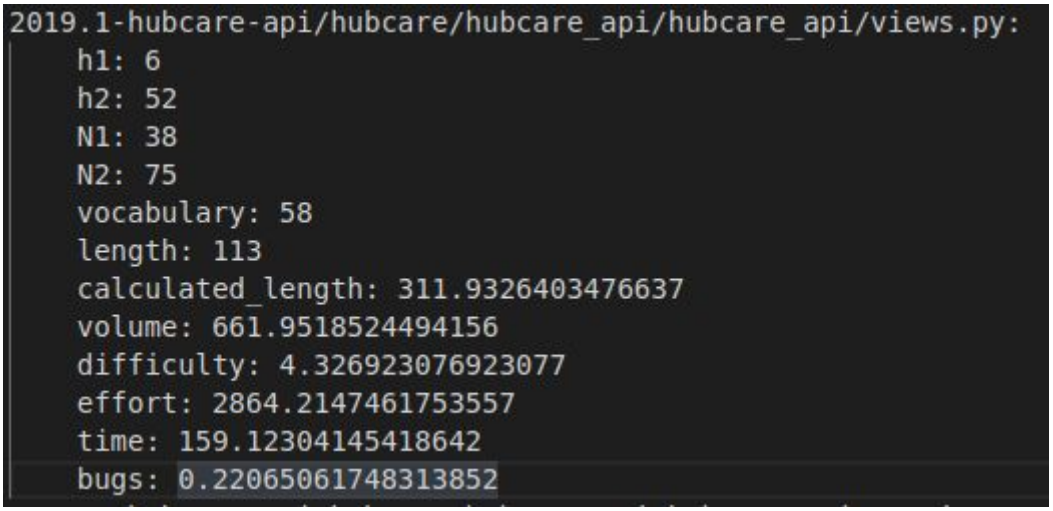
- Ao fazer uso da ferramenta Radon, foram usados os seguintes comandos:
    - *radon cc -a 2019.1-hubcare-api*
      (ferramenta)(Cyclomatic Complexity)(average)(diretório)

  O resultado final obtido foi de que a complexidade ciclomática do sistema (média entre os arquivos) é de aproximadamente 2.10, o que configura baixa complexidade, se enquadrando na mais alta classificação possível (A):

```
168 blocks (classes, functions, methods) analyzed.
Average complexity: A (2.1011904761904763)
```

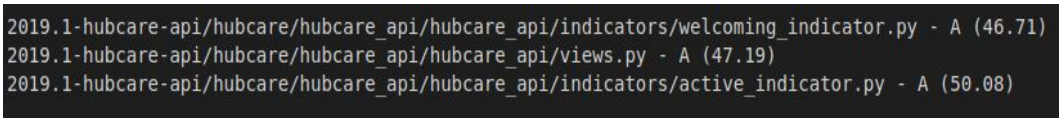
- Foi usado o seguinte comando para obter as métricas de Halstead:
    - *radon hal 2019.1-hubcare-api*
      (ferramenta)(Halstead Metrics)(diretório)

O arquivo mais complexo de acordo com as métricas de Halstead está ilustrado abaixo.



- Foi usado o seguinte comando na ferramenta Radon para obter o índice de manutenibilidade dos arquivos:
    - *radon mi -s 2019.1-hubcare-api*
      (ferramenta)(Manutenability Index)(score)(diretório)

  O resultado obtido, de acordo com as métricas utilizadas, foi de que todos arquivos do projeto possuem manutenibilidade adequada.
  O índice de manutenibilidade médio do projeto é de 90.
  As três menores pontuações estão ilustradas na figura a seguir.



Conclusão

O índice de manutenibilidade obtido na média entre os arquivos do projeto foi de 90, considerado um índice alto na escala que vai de 0 a 100. Uma das possíveis explicações encontradas para essa alta manutenibilidade, obtida pelas métricas utilizadas, é o fato do Django ser um framework que tem seus módulos claros e bem definidos, de maneira que a divisão por aplicativos, cada um contendo a estrutura padrão do Django (admin, apps, models, serializers, urls, views) propicia arquivos curtos, com pouca complexidade e baixo acoplamento, auxiliando estruturas como testes de unidade e testes de integração.

Com o estudo aprofundado dos conceitos de modificabilidade e manutenibilidade, foi possível constatar que, para verificar se um software possui essas características de qualidade, não são suficientes testes convencionais que indicam conformidade ou não conformidade com requisitos do sistema, porque essas características refletem métodos de programação aplicados no desenvolvimento do código, ou seja, buscam averiguar se a forma como o código foi empregado permite uma refatoração ou complementação de forma fácil e direta. Dado esse contexto, e com base no material bibliográfico consultado, foi possível constatar que as formas de verificar a existência dessas características é por meio de métricas, que se referem a

diferentes aspectos do código e obtêm indicadores, normalmente através de valores numéricos ou de fórmulas que relacionam os valores obtidos para atingir um resultado numérico interpretável. Com base nesses resultados numéricos é que se pode testar a aplicação quanto a um nível - a ser definido como satisfatório ou não - de manutenibilidade, que pode ser obtido pela métrica "Índice de Manutenibilidade".

Referências bibliográficas

ABNT - Associação Brasileira de Normas Técnicas. **ISO 9000**. Disponível em: <http://www.jvasconcellos.com.br/unijorge/wp-content/uploads/2012/09/Abnt-Nbr-Iso-9000.pdf>. Acesso em: 13 de outubro de 2019.

ABNT - Associação Brasileira de Normas Técnicas. **ISO 9126**. Disponível em: <https://jkolb.com.br/wp-content/uploads/2014/02/NBR-ISO_IEC-9126-1.pdf>. Acesso em: 13 de outubro de 2019.

CODE CLIMATE. **Documentação Code Climate (Qualidade)**. Disponível em: <https://docs.codeclimate.com/docs/maintainability>. Acesso em: 13 de outubro de 2019.

JØRGENSEN, Anker Helms. **A methodology for measuring the readability and modifiability of computer programs**. BIT Numerical Mathematics, v. 20, n. 4, p. 393-405, 1980.

LAGERSTRÖM, Robert; JOHNSON, Pontus; HÖÖK, David. **Architecture analysis of enterprise systems modifiability–models, analysis, and validation**. Journal of Systems and Software, v. 83, n. 8, p. 1387-1403, 2010.

MEYERS, G., BADGETT, T., SANDLER, C., et al., 2004, **The Art of Software Testing**. 2nd Ed. JW&S., Inc.

PROJECT CODE METER. **Definição de *Source Lines of Code (SLOC)***. Disponível em: <http://www.projectcodemeter.com/cost_estimation/help/GL_sloc.htm>. Acesso em 15 de outubro de 2019.

SOURCE FORGE. **Cloc**. Disponível em: <http://cloc.sourceforge.net/>. Acesso em 13 de outubro de 2019.

GALIN, Daniel. **Software quality assurance - From Theory to Implementation**, Pearson, Addison Wesley, 2004

Thirumalai, Chandrasegar, et al. **An assessment of halstead and COCOMO model for effort estimation**. 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), 2017.

D. Gonzalez, J. C. Santos, A. Popovich, M. Mirakhorli, and M. Nagappan, **A Large-Scale Study on the Usage of Testing Patterns That Address Maintainability Attributes: Patterns for Ease of Modification, Diagnoses, and Comprehension**, *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 2017.

Oman, P., and J. Hagemeister. **Metrics for assessing a software system's maintainability**. Proceedings Conference on Software Maintenance 1992.

RYCHKOVA, Irina et al. **A pragmatic approach for Measuring maintainability of DPRA models**. arXiv preprint arXiv:1706.02259, 2017.

NAIK, Kshirasagar. TRIPATHY, Priyadarshi. **Software Testing and Quality Assurance - Theory and Practice**, John Wiley & Sons , 2008.

HALSTEAD. **IBM.** Disponível em: <https://www.ibm.com/support/knowledgecenter/en/SSSHUF_8.0.2/com.ibm.rational.testrt.studio.doc/topics/csmhalstead.htm>. Acesso em 15 de outubro de 2019.

Apêndice A - Critérios de exclusão e inclusão para a seleção e revisão de estudos

Questões de pesquisa	Critérios de inclusão	Critérios de exclusão
Definições	Estudos que definam modificabilidade; Estudos que comparem manutenibilidade e modificabilidade. Estudos que definem métricas utilizadas para avaliar modificabilidade.	Estudos que abordam somente conceitos de outros atributos de qualidade.
Tipo de publicação	Jornais, periódicos, conferências, normas.	Estudos que não estejam na língua portuguesa, inglesa e espanhola.
Escopo do objetivo	Estudos que abordam ferramentas e métricas relacionadas a modificabilidade.	Arquitetura e metodologia muito específica em relação ao escopo.