Associação Propagadora Esdeva
Centro Universitário Academia – UniAcademia
Curso de Engenharia de Software
Trabalho de Conclusão de Curso – Artigo

Análise estatística da qualidade do software Drupal

Bruno Goulart Mosse[1]
Centro Universitário Academia, Juiz de Fora, MG

Tassio Ferenzini Martins Sirqueira[2]
Centro Universitário Academia, Juiz de Fora, MG

Linha de Pesquisa: Engenharia de Software

RESUMO

No início do desenvolvimento Web, era necessário que as pessoas possuíssem conhecimentos em programação para a criação de sites. Contudo, com o passar dos anos, surgiram ferramentas gerenciadoras de conteúdo, os CMSs (*Content Management System*), permitindo que até mesmo pessoas com poucos conhecimentos técnicos possam criá-los com facilidade. Em decorrência disso, houve uma ascensão em sua utilização fazendo com que novas ferramentas chegassem ao mercado. Conforme PATEL, RATHOD e PARIKH (2011), três softwares vêm se destacando, Drupal, Wordpress e Joomla. Diante dessas opções, propõe-se uma análise estatística coletando um conjunto de métricas que servem como base para a avaliação da qualidade do código fonte do software Drupal ao longo de suas versões. Fatores como usabilidade da ferramenta, oscilações no número de sites em atividade utilizando-o não foram considerados para o estudo, apenas a análise na evolução e qualidade do código do software. Este artigo, com a utilização de métodos estatísticos baseados na Engenharia de Software experimental e com um conjunto de métricas coletadas referentes a qualidade do código, apresenta indícios da confiabilidade de sua aplicação, verificando a viabilidade de sua utilização ou a necessidade de migração para outro CMS.

Palavras-chave: Avaliação estatística. Qualidade de Software. Evolução. Métrica.

[1] Discente do Curso de Engenharia de Software do Centro Universitário Academia – UniAcademia. Endereço: Rua pétala misteriosa 300, bloco 5/504. Celular: (21) 97174-0440. E-mail: m.goulartbruno@gmail.com.
[2] Docente do Curso de Engenharia de Software do Centro Universitário Academia. Orientador.

ABSTRACT

In the early days of web development, it was necessary for people to have programming skills in order to create websites. However, over the years, content management systems (CMS) have emerged, allowing even people with little technical knowledge to create them with ease. As a result, there has been a rise in its use, and new tools have come onto the market. According to PATEL, RATHOD and PARIKH (2011), three software have been standing out, Drupal, Wordpress and Joomla. Given these options, we propose a statistical analysis by collecting a set of metrics that serve as a basis for evaluating the quality of the source code of Drupal software over its versions. Factors such as usability of the tool, oscillations in the number of sites in activity using it were not considered for the study, only the analysis in the evolution and quality of the software code. This article, with the use of statistical methods based on experimental Software Engineering and with a set of metrics collected regarding the quality of the code, presents evidence of the reliability of its application, verifying the viability of its use or the need to migrate to another CMS.

1 INTRODUÇÃO

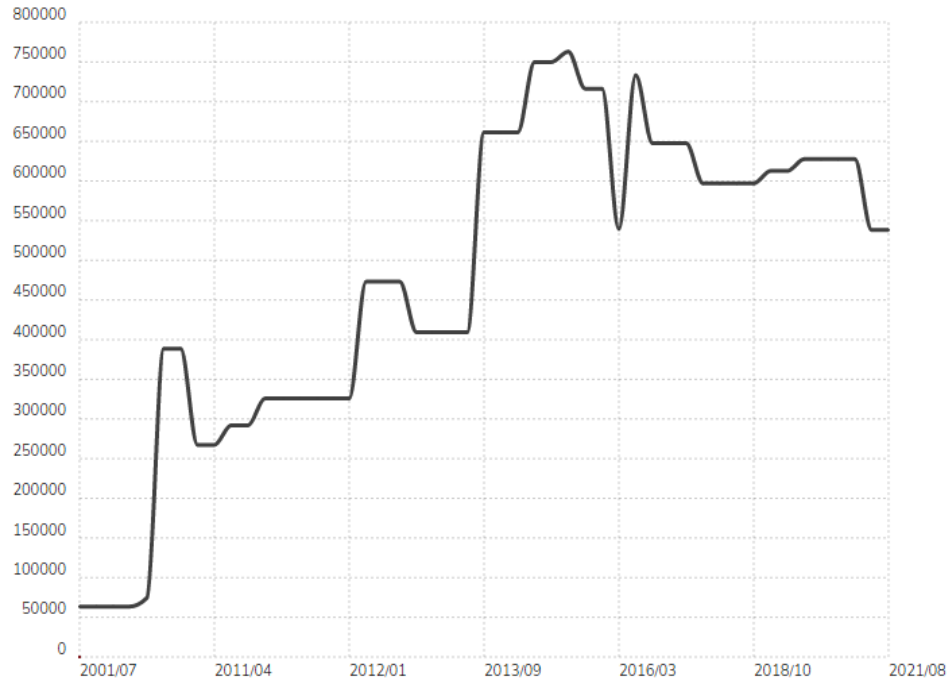
Com base no estudo (AMORIM *et al.*, 2011) o Drupal[3] é um *framework* modular e um sistema de gerenciamento de conteúdo (CMS – do inglês *Content Management System*) desenvolvido na linguagem PHP[4]. O Drupal permite criar e organizar conteúdo, manipular a aparência, automatizar tarefas administrativas e definir permissões e papéis para usuários e colaboradores. Para o seu funcionamento, é necessário um servidor HTTP (*Hypertext Transfer Protocol)*, como o Apache[5], e um servidor de banco de dados, como o MySQL[6]. O Drupal é um software livre, que, por esse motivo, facilita o desenvolvimento de novos módulos extensivos, implementações de integrações com APIs de forma fácil para os desenvolvedores. Com a chegada dos CMSs, o desenvolvimento de sites deixou de ser restrito apenas à desenvolvedores de software, e permitiu que pessoas leigas tecnicamente pudessem construir seus próprios sites. A praticidade, customização e padronização com as referências de navegabilidade estipuladas pelo W3C[7], o Consórcio da Internet que determina os padrões a serem utilizados pelas páginas da web. Desta forma, os sites que utilizam um CMS funcionam em todos os navegadores e

[3] Drupal. Disponível em: <https://www.drupal.org/>. Acessado em 22 de novembro de 2021.
[4] PHP. Disponível em: <https://www.php.net/>. Acessado em 22 de novembro de 2021.
[5] Apache. Disponível em: <https://www.apache.org/>. Acessado em 22 de novembro de 2021.
[6] MySQL. Disponível em: <https://www.mysql.com/>. Acessado em 22 de novembro de 2021.
[7] W3C. Disponível em: <https://www.w3c.br/>. Acessado em 22 de novembro de 2021.

dispositivos móveis. Com isso, o percentual de sites que são construídos utilizando os gerenciadores de conteúdo, tem crescido ao longo dos anos.

A figura 1 apresenta uma representação gráfica, retirada do site BuiltWith[8], do número de sites que utilizam o Drupal desde julho de 2001. Pode-se observar o crescimento da sua utilização ao longo dos anos, chegando a sua maior marca com cerca de 750 mil sites ativos. A última informação coletada foi em agosto de 2021 com aproximadamente 550 mil sites.

Figura 1. Representação gráfica da quantidade de sites ativos que utilizam o Drupal.



Fonte: https://trends.builtwith.com/cms/Drupal.

Devido a esse crescimento da utilização desse *framework*, este trabalho propõe uma análise da qualidade de seu código fonte a fim de concluir se o software está em um crescimento ordenado ou em decaimento. Assim, verificar a segurança da utilização desta ferramenta ou então a possibilidade de uma migração para outro CMS, caso a qualidade do software esteja em declínio. Fatores como usabilidade da ferramenta, oscilações no número de sites em atividade utilizando-o não foram considerados para o estudo, apenas a análise na evolução e qualidade do código do software.

[8] BuiltWith. Disponível em: <https://builtwith.com/>. Acessado em 22 de novembro de 2021.

Este artigo é composto por mais 3 seções. Na seção 2 será apresentado o referencial teórico com o conceito das métricas coletadas e utilizadas no estudo. Já na seção 3 é apresentada a metodologia realizada para a coleta das métricas. Na seção 4, são descritos os métodos estatísticos aplicados e os resultados obtidos com o uso dessas técnicas. Por fim, na seção 5 são descritas as conclusões acerca dos resultados coletados na seção 4 e os indícios da viabilidade da utilização do Drupal.

2 REFERENCIAL TEÓRICO

Conforme (WARNARS *et al.*, 2017), em um estudo experimental é de suma importância que seja compreendido o conceito de cada métrica a ser analisada e o que ela representa para a qualidade do software. Por isso, nesta seção serão abordados os conceitos de cada métrica utilizada neste estudo.

2.1 Medidas de complexidade Halstead (*Average_bugs_by_class (Halstead)*)

Conforme AWODE (2017), essa métrica tem como objetivo determinar uma medida quantitativa da complexidade a partir dos operadores e os operandos contidos em um módulo através do código fonte. Está relacionada à dificuldade no processo de escrever ou entender uma parte do código ao realizar a sua manutenção ou revisão.

2.2 *Lack of Cohesion of Methods (average_LCOM)*

Conforme (LAKSHMINARAYANA; NEWMAN, 1999), LCOM ou Falta de coesão dos métodos mede a correlação entre os métodos e as variáveis de instância local de uma classe. Alta coesão indica boa subdivisão de classe e a baixa coesão aumenta a complexidade. As classes com baixa coesão provavelmente poderiam ser subdivididas em duas ou mais subclasses. É calculado através da proporção de métodos em uma classe que não acessam um campo de dados específico.

2.3 Linhas de código (LOC)

Baseado no estudo (GRAYLIN *et al.*, 2009), a métrica de linhas de código pode ser coletada de 3 maneiras: por método, classe e pelo projeto completo, e, dessa forma, é possível medir o tamanho do software. Com a informação da quantidade de linhas por método, pode-se estipular um valor máximo de linhas que, caso seja superior, pode-se concluir que haja uma complexidade elevada na função, sendo assim complicada a sua compreensão. O mesmo pode

ser aplicado para o número de linhas em uma classe. Existe o LOC lógico que contém apenas as linhas de código executadas, não sendo consideradas definições, importações de namespace, linhas de comentários e linhas em branco. Além disso, existe o LOC físico que contém todas as linhas de código do sistema.

2.4 Número de classes (NOC)

No trabalho (CHIDAMBER, 1994) é apresentada a métrica NOC. Essa métrica conta o número de classes do sistema. Acredita-se que quanto maior o número de classes que o sistema possua, mais difícil será a compreensão e a manutenção do mesmo. Entretanto, como nas outras métricas a NOC pode levar a conclusões errôneas. Um número maior de classes pode indicar que o sistema está mais coeso. Ou seja, um número grande de classes altamente coesas é mais desejável do que um pequeno número de classes pouco coesas.

2.5 Complexidade relativa do sistema (RSYSC - *Average_relative_System_complexity*)

Conforme (AL-FAR *et al.*, 2018), o RSYSC mede a complexidade de um projeto em termos de chamadas de métodos, passagem de parâmetros e uso de dados. Além disso, um alto RSYSC prevê um número maior de erros por linha de código. É calculada através da fórmula:

- Complexidade relativa do sistema = média da complexidade externa + complexidade interna para todos os procedimentos.

2.6 Média de métodos por classe (*Average_weighted_method_count_by_class_(CC)*)

Seguindo o estudo (CHIDAMBER, 1994) o número de métodos e a complexidade desses é um indicador do esforço de desenvolvimento e de manter a classe. Indica como a responsabilidade de cada classe está sendo definida. Dessa maneira, quanto menor for a média, menor a complexidade de manutenção e maior a coesão entre as classes, assim sendo o cenário ideal para um software.

3 METODOLOGIA

Baseado na metodologia aplicada pelo artigo (CARIGÉ; DE FIGUEIREDO CARNEIRO, 2020), esta seção apresenta os processos realizados para este estudo experimental. Com início no desenvolvimento da ferramenta Drupal Code Analysis para auxiliar na análise do código das versões do Drupal. Em seguida, a utilização da ferramenta RStudio para a geração de gráficos

e dados estatísticos das métricas coletadas. E, por fim, uma análise a respeito dos resultados obtidos.

Para este estudo, foi utilizada a versão 7.4.16 do PHP, 2.1.3 do Composer[9], 2.7.4 do PhpMetrics[10], 12.16.3 do Node.js[11] e 2.10.1 do Php Depend[12]. Foram selecionas estas versões por motivos de compatibilidade entre elas. A análise foi feita da versão 7.0 a 9.2.4 do Drupal, com um total de 239 versões e 6 métricas coletadas para cada uma delas. Versões beta, alpha, rc e unstable não foram analisadas. Para a coleta das métricas, foi desenvolvida a ferramenta Drupal Code Analysis, disponível no GitHub[13], utilizando a linguagem Node.js. As etapas do processo de análise para cada versão podem ser vistas na figura 2, na qual consistem em:

1. Buscar a lista de todas as versões do Drupal através do link na plataforma do GitHub[14];
2. Após feita a busca, o Drupal Code Analysis acessa a planilha onde estão sendo salvas as métricas e busca a última versão analisada, pois assim, o sistema pode selecionar a versão subsequente que será analisada. Caso não encontre, ou seja, esteja na última versão, o programa finaliza;
3. A terceira etapa consiste em clonar o repositório do Drupal disponível GitHub[15], que será baixado em formato zip;
4. Após o projeto baixado, é feita a extração do arquivo zipado e em seguida é removido o zip para economizar espaço em disco;
5. Com o projeto na máquina local, o programa executa o comando de análise da ferramenta PhpMetrics. Ao término deste procedimento, a ferramenta cria uma pasta report na raiz do projeto com todas as métricas coletadas disponibilizadas através de páginas HTML. A figura 4 representa os dados gerados da versão 7.0 analisada pela ferramenta. Após as métricas geradas, estes dados são salvos em memória para que sejam utilizados na etapa 7;
6. Neste momento, o programa irá realizar a última análise do código utilizando a ferramenta PHP Depend. Ao término deste procedimento, a ferramenta cria um arquivo chamado

[9] Composer. Disponível em: <https://getcomposer.org/>. Acessado em 22 de novembro de 2021.
[10] PhpMetrics. Disponível em: <https://phpmetrics.org/>. Acessado em 22 de novembro de 2021.
[11] Node.js. Disponível em: <https://nodejs.org/en/>. Acessado em 22 de novembro de 2021.
[12] Php Depend. Disponível em: <https://pdepend.org/>. Acessado em 22 de novembro de 2021.
[13] Drupal Code Analysis. Disponível em: <https://github.com/BrunoGrM/drupal-code-analysis>. Acessado em 22 de novembro de 2021.
[14] Histórico de versões do Drupal. Disponível em: <https://api.github.com/repos/drupal/drupal/git/refs/tags>. Acessado em 22 de novembro de 2021.
[15] Código fonte do Drupal. Disponível em: <https://github.com/drupal/drupal>. Acessado em 22 de novembro de 2021.

jdepend.xml na raiz do projeto com todas as métricas coletadas, exemplo na figura 5. Com o arquivo gerado, o programa faz a sua leitura e armazena as métricas em memória;

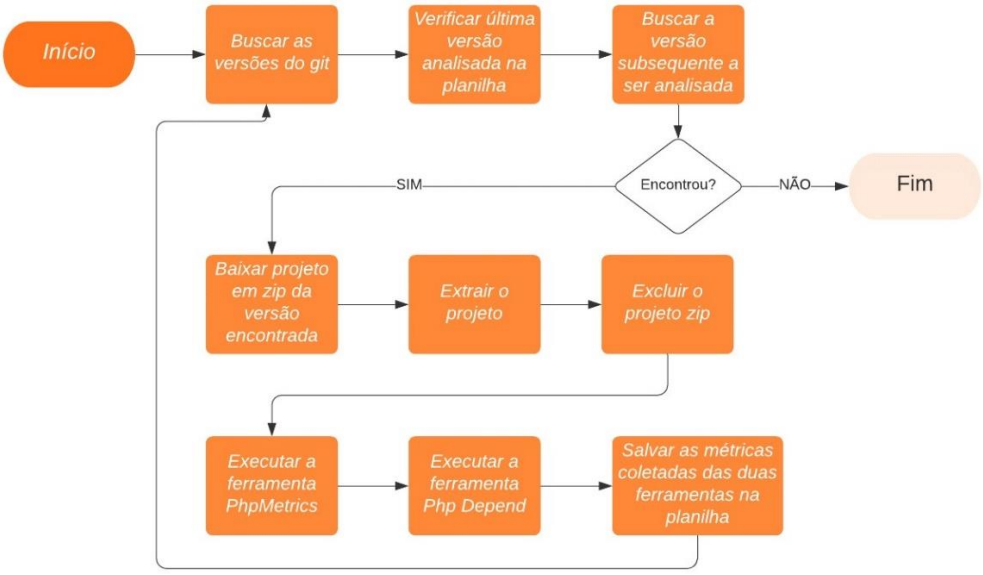
7. Por fim, com todas as métricas coletadas nos processos 6 e 7, a última etapa é fazer a conexão com a planilha do Google[16] e salvar todos os dados. A planilha está disponível no repositório do Drupal Code Analysis na pasta docs > métricas.xlsx.

Para auxiliar nas etapas de coleta dos dados, foram utilizadas algumas bibliotecas.

- axios[17] - buscar as versões do Drupal no site do GitHub e realizar o download do repositório (etapas 1 e 3);
- progress[18] e progress-extract[19] - extrair o repositório em formato zip (etapa 4);
- child_process[20] - rodar os comandos de execução da análise do código das ferramentas PhpMetrics e Php Depend (etapas 5 e 6);
- xml-js[21] - leitura das métricas geradas pelo arquivo xml da ferramenta PHP Depend (etapa 6);
- google-spreadsheet[22] - API para conectar com a planilha do Google (etapas 2 e 7)

[16] Google Planilhas. Disponível em: <https://workspace.google.com/intl/pt-BR/products/sheets/>. Acessado em 22 de novembro de 2021.
[17] Axios. Disponível em: <https://www.npmjs.com/package/axios>. Acessado em 22 de novembro de 2021.
[18] Progress. Disponível em: <https://www.npmjs.com/package/progress>. Acessado em 22 de novembro de 2021.
[19] ProgressExtract. Disponível em: <https://www.npmjs.com/package/progress-extract>. Acessado em 22 de novembro de 2021.
[20] ChildProcess. Disponível em: <https://www.npmjs.com/package/child_process>. Acessado em 22 de novembro de 2021.
[21] XmlJS. Disponível em: <https://www.npmjs.com/package/xml-js>. Acessado em 22 de novembro de 2021.
[22] Google Spreadsheet. Disponível em: <https://www.npmjs.com/package/google-spreadsheet>. Acessado em 22 de novembro de 2021.

Figura 2. Fluxograma dos processos do programa.



Fonte: Elaboração própria.

A figura 3 representa o log do resultado final da execução da ferramenta Drupal Code Analysis para a versão 7.0 do Drupal.

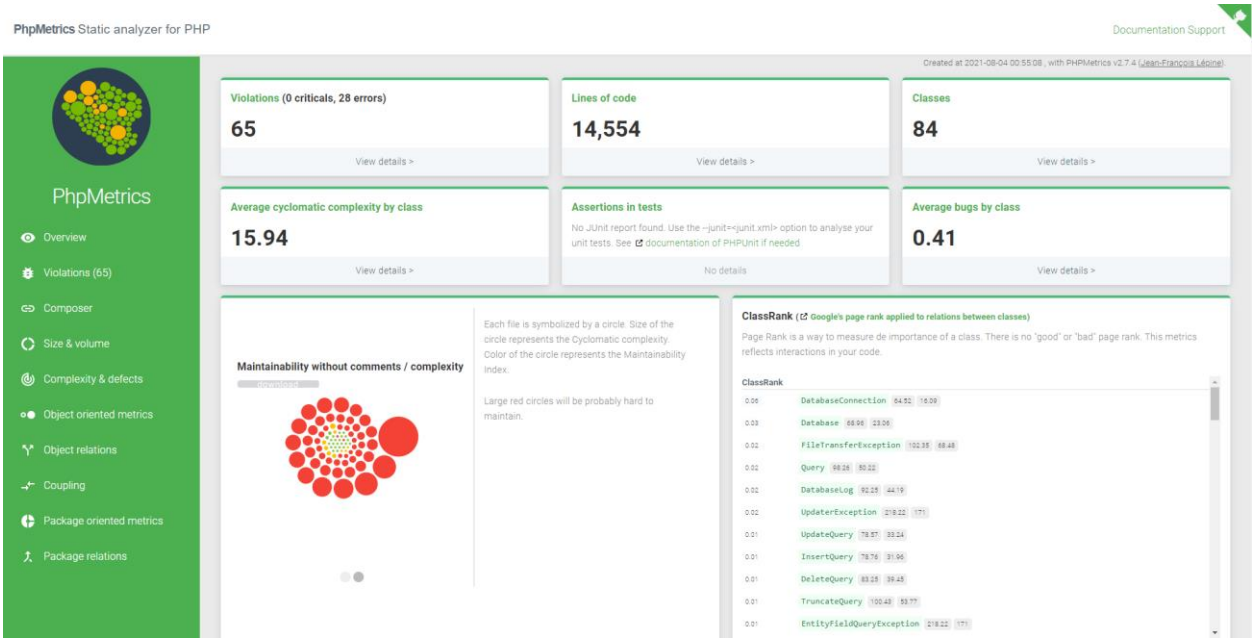
Figura 3. Log do final da execução da ferramenta.

```
Calculating Class Level metrics:
.............................................................  1200
.................                                              1547

Calculating Cohesion metrics:
.............................................................  1200
...........................                                    1794

Calculating Halstead metrics:
........................                                        560

Calculating Maintainability Index metrics:
........................                                        560

Calculating Dependency metrics:
......                                                          121

Generating pdepend log files, this may take a moment.

Time: 0:00:03; Memory: 132.00Mb

- Temp files deleted!

- Analysis completed!
(node:10892) [DEP0147] DeprecationWarning: In future versions of Node.js, fs.rmdir(path, { recursive: true }) will be removed.
 Use fs.rm(path, { recursive: true }) instead
(Use `node --trace-deprecation ...` to show where the warning was created)

- Saving data on spreadsheet...

- Analysis Drupal 7.0 completed!
- Process finished in: 15.07s!
- Access data in: https://docs.google.com/spreadsheets/d/1mv-LQO4OOxzYmwD4uqwgbOFP5jrfGa-Slm1hIj7bOfo/edit#gid=674073863
[nodemon] clean exit - waiting for changes before restart
```

Fonte: Elaboração própria.

A figura 4 apresenta a tela inicial do relatório em HTML gerado pela ferramenta PhpMetrics. Nessa tela é possível visualizar algumas métricas, como o número de violações, quantidade linhas de código, número de classes, média de complexidade ciclomática por classe, média de bugs por classe. Pelo menu lateral esquerdo é possível visualizar outras métricas geradas pela ferramenta. Este relatório está disponível por completo no repositório do Drupal Code Analysis na pasta docs > report.

Figura 4. Tela inicial do relatório gerado pela ferramenta PhpMetrics.



Fonte: Elaboração própria.

A figura 5 apresenta o conteúdo do arquivo xml gerado pela ferramenta Php Depend. As métricas coletadas estão todas disponíveis neste arquivo através da tag *metrics*. Este arquivo está disponível por completo no repositório do *Drupal Code Analysis* na pasta docs > jdepend.xml.

Figura 5. Arquivo xml gerado pela ferramenta Php Depend.


```
<?xml version="1.0" encoding="UTF-8"?>
- <metrics roots="1" nop="1" nom="107" noi="0" nof="400" noc="3" ncloc="6746" maxDIT="1" loc="16695" lloc="2988" leafs="2"
fanout="22" eloc="5676" clsc="2" clsa="1" cloc="9949" ccn2="1290" ccn="1173" calls="1292" andc="0.66666666666667" ahh="1"
pdepend="2.9.0snapshot202103110923" generated="2021-08-04T00:56:15">
  - <files>
      <file ncloc="1726" loc="3391" lloc="996" eloc="1462" cloc="1665" name="E:\Faculdade\TCC_FINAL\drupal-
        versions\drupal-7.0\modules\simpletest\drupal_web_test_case.php"/>
      <file ncloc="110" loc="177" lloc="60" eloc="85" cloc="67" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \authorize.php"/>
      <file ncloc="48" loc="233" lloc="18" eloc="37" cloc="185" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\aggregator\aggregator.api.php"/>
      <file ncloc="105" loc="354" lloc="49" eloc="83" cloc="249" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\block\block.api.php"/>
      <file ncloc="48" loc="147" lloc="22" eloc="34" cloc="99" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\comment\comment.api.php"/>
      <file ncloc="9" loc="42" lloc="2" eloc="4" cloc="33" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\contextual\contextual.api.php"/>
      <file ncloc="13" loc="44" lloc="4" eloc="7" cloc="31" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\dashboard\dashboard.api.php"/>
      <file ncloc="1040" loc="2567" lloc="445" eloc="903" cloc="1527" name="E:\Faculdade\TCC_FINAL\drupal-
        versions\drupal-7.0\modules\field\field.api.php"/>
      <file ncloc="35" loc="66" lloc="10" eloc="29" cloc="31" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\field\modules\options\options.api.php"/>
      <file ncloc="107" loc="206" lloc="31" eloc="86" cloc="99" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\field_ui\field_ui.api.php"/>
      <file ncloc="15" loc="68" lloc="6" eloc="11" cloc="53" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\file\file.api.php"/>
      <file ncloc="81" loc="325" lloc="28" eloc="63" cloc="244" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\filter\filter.api.php"/>
      <file ncloc="15" loc="64" lloc="6" eloc="9" cloc="49" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\help\help.api.php"/>
      <file ncloc="72" loc="195" lloc="22" eloc="57" cloc="123" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
        \modules\image\image.api.php"/>
      <file ncloc="86" loc="198" lloc="32" eloc="70" cloc="112" name="E:\Faculdade\TCC_FINAL\drupal-versions\drupal-7.0
```


Fonte: Elaboração própria.

A tabela 1 apresenta as métricas que foram utilizadas no estudo e de qual ferramenta, PhpMetrics ou Php Depend, foi coletada.

Tabela 1. Métricas coletadas por ferramenta

Métrica	Ferramenta
Average_bugs_by_class (Halstead)	PhpMetrics
average_LCOM	PhpMetrics
LOC	Php Depend
NOC	Php Depend
Average_relative_System_complexity	PhpMetrics
Average_weighted_method_count_by_class_(CC)	PhpMetrics

Fonte: Elaboração própria.

Ao término do processo de coleta das métricas de todas as versões, foi utilizada a ferramenta RStudio[23] para a análise dos dados. O RStudio é um software *Open Source* de

[23] RStudio. Disponível em: <https://www.rstudio.com/>. Acessado em 22 de novembro de 2021.

ambiente de desenvolvimento integrado para a linguagem R, comumente utilizada para a geração de gráficos e cálculos estatísticos. Com o auxílio de um script compilado nesta ferramenta, chegou-se ao resultado final do estudo experimental.

4 RESULTADOS E DISCUSSÃO

Seguindo a orientação descrita no artigo (SIRQUEIRA *et al.,* 2020), o primeiro passo para a análise estatística dos dados é a identificação do número de amostras a serem analisadas. Caso este número seja maior que 30, deve-se utilizar o Teste de Kolmogorov-Smirnov (K-S) e caso seja menor, deve-se utilizar o Teste de Shapiro-Wilk para verificar a normalidade dos dados. O Teste de Kolmogorov-Smirnov é um teste não paramétrico que possibilita a avaliação quanto às semelhanças entre a distribuição de duas amostras. Também pode indicar a similaridade na distribuição de uma amostra em relação a uma distribuição clássica. Para este estudo serão analisadas 239 amostras, logo será aplicado o Teste de K-S considerando um nível de significância de 5%, sendo:

- H0: A distribuição é normal
- H1: A distribuição não é normal

Após aplicado o teste para todas as métricas, todos os dados não seguem uma distribuição normal. Portanto, foi feito o teste não paramétrico Kruskal-Wallis pois foi utilizado um fator e 3 tratamentos (versões 7, 8 e 9), considerando um nível de significância de 5%, sendo:

- H0: Os grupos são semelhantes
- H1: Pelo menos um grupo é diferente dos demais

Após aplicado o teste para todas as métricas, chegou-se à conclusão que não possuem dados semelhantes entre as amostras. Este resultado é esperado devido a grande quantidade de amostras analisadas.

Tabela 2. Análise dos dados da métrica *Average_bugs_by_class (Halstead)*.

Medidas	Valor
Min	0,23
1º Quartil	0,23
Mediana	0,23

Média	0,29
3º Quartil	0,40
Máximo	0,44
Não encontrados	-
Média amostral	0,2899582
Desvio padrão amostral	0,08416919

Fonte: Elaboração própria.

Tabela 3. Análise dos dados da métrica *average_LCOM*.

Medidas	Valor
Min	2,130
1º Quartil	2,140
Mediana	2,170
Média	2,196
3º Quartil	2,260
Máximo	2,390
Não encontrados	-
Média amostral	2,195774
Desvio padrão amostral	0,07531535

Fonte: Elaboração própria.

Tabela 4. Análise dos dados da métrica LOC.

Medidas	Valor
Min	16695
1º Quartil	18348
Mediana	860602
Média	613843
3º Quartil	1024472
Máximo	1181129
Não encontrados	40
Média amostral	613843,3
Desvio padrão amostral	489492,8

Fonte: Elaboração própria.

Tabela 5. Análise dos dados da métrica NOC.

Medidas	Valor
Min	3
1º Quartil	9
Mediana	6043
Média	4335
3º Quartil	7508
Máximo	8425
Não encontrados	40
Média amostral	4335,322
Desvio padrão amostral	3557,041

Fonte: Elaboração própria.

Tabela 6. Análise dos dados da métrica *Average_relative_System_complexity*.

Medidas	Valor
Min	107,8
1º Quartil	122,8
Mediana	200,5
Média	181,6
3º Quartil	222,5
Máximo	226,7
Não encontrados	-
Média amostral	181,6111
Desvio padrão amostral	45,13194

Fonte: Elaboração própria.

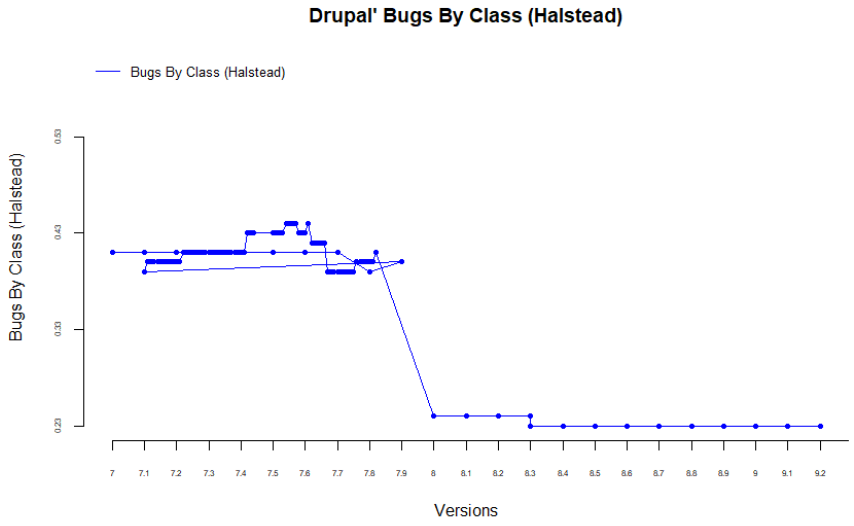
Tabela 7. Análise dos dados da métrica *Average_weighted_method_count_by_class_(CC)*.

Medidas	Valor
Min	12,69
1º Quartil	12,77
Mediana	13,00
Média	16,42
3º Quartil	23,18

Máximo	24,98
Não encontrados	-
Média amostral	16,41845
Desvio padrão amostral	5,128651

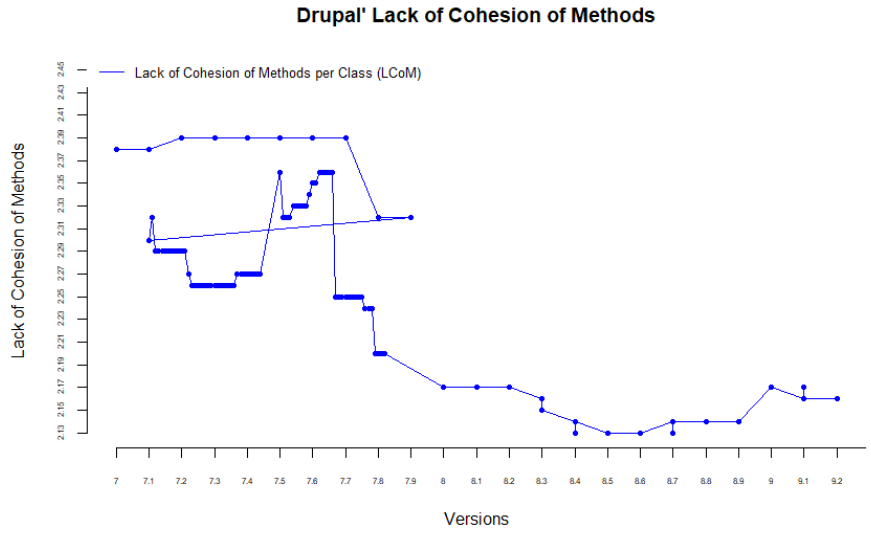
Fonte: Elaboração própria.

Figura 6. Gráfico de linha da métrica *Average_bugs_by_class (Halstead)*.



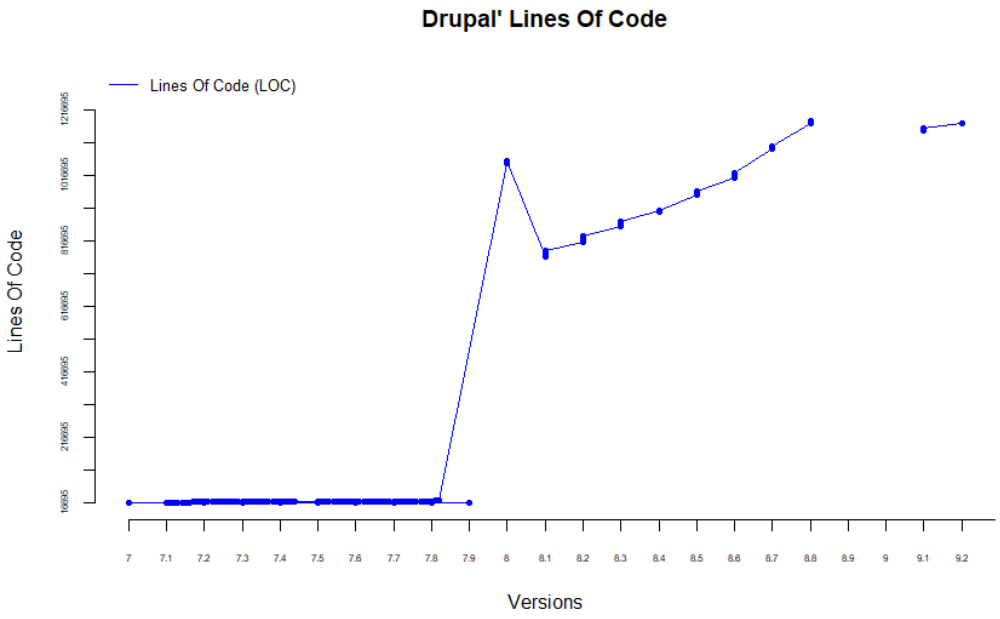
Fonte: Elaboração própria.

Figura 7. Gráfico de linha da métrica *average_LCOM*.



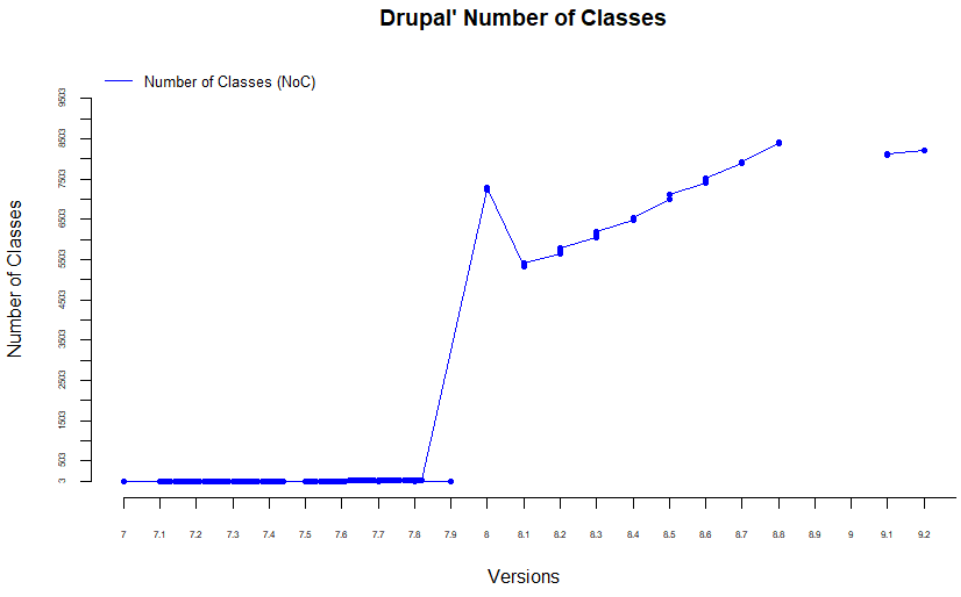
Fonte: Elaboração própria.

Figura 8. Gráfico de linha da métrica LOC.



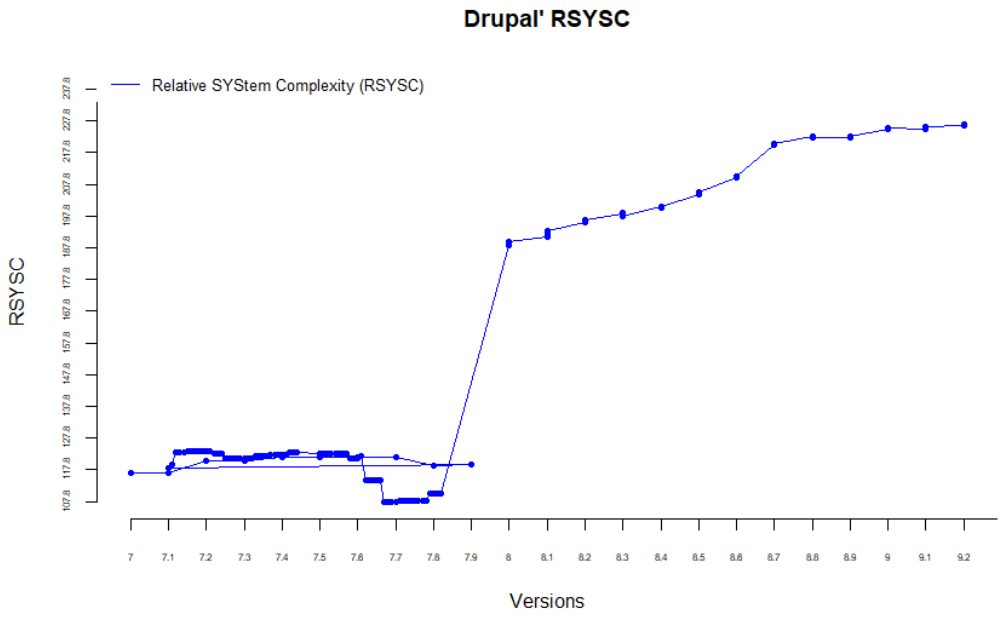
Fonte: Elaboração própria.

Figura 9. Gráfico de linha da métrica NOC.



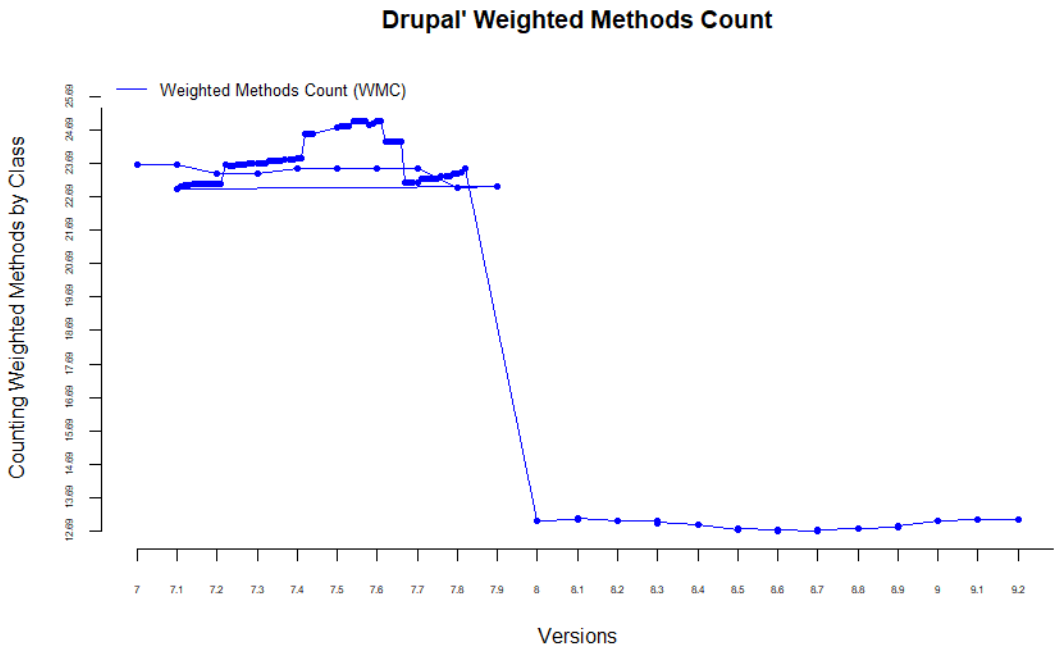
Fonte: Elaboração própria.

Figura 10. Gráfico de linha da métrica *Average_relative_System_complexity*.



Fonte: Elaboração própria.

Figura 11. Gráfico de linha da métrica *Average_weighted_method_count_by_class_(CC)*.



Fonte: Elaboração própria.

Com base nos testes realizados, conforme a metodologia descrita no artigo (SIRQUEIRA *et al.,* 2020), junto a análise dos dados coletados das métricas e a leitura dos gráficos gerados pela ferramenta RStudio, pôde-se chegar a um resultado que será apresentado na seção 5.

5 CONSIDERAÇÕES FINAIS

Pelos resultados das análises efetuadas, constatou-se que estatisticamente a qualidade do código fonte do software Drupal ao longo das versões apresentou melhoras ao longo de suas versões e vêm se mantendo estável. Da versão 7 para a 8, houve um grande impacto positivo para o projeto, no qual pôde-se analisar que o sistema cresceu em questão de tamanho e diminuiu sua complexidade em relação a versão anterior. De acordo com os dados coletados da métrica *Average_bugs_by_class (Halstead)*, como pode ser vista na figura 6, mostrou-se que ao longo das versões teve uma queda neste número, e que, desde a versão 8, vêm se mantendo baixa. Sendo assim, um resultado positivo para a qualidade do sistema, pois quanto menor o número de bugs, melhor para um software. Baseado nos dados coletados da métrica *average_LCOM*, como pode ser vista na figura 7, mostrou-se que ao longo das versões ocorreu uma queda no seu valor, e que, desde a versão 8 este número vem se mantendo em baixa também. Sendo assim, um resultado positivo para o projeto, significando que as classes estão coesas. Com os dados coletados através da métrica LOC, como pode ser vista na figura 8, mostrou-se que, ao longo das versões, este número vem crescendo, e que, além disso, os dados da métrica NOC, como pode ser vista na figura 9, também cresceram, logo se tem um comportamento esperado conforme o crescimento do tamanho de um sistema. Já com os dados coletados da métrica *Average_relative_System_complexity*, como pode ser vista na figura 10, mostrou-se que ao longo das versões esse número cresceu, principalmente da versão 7.8 para a 7.9 e depois se manteve em um crescimento baixo ao longo das versões. Isto é um comportamento esperado, pois como essa métrica é baseada por linha de código e, como esse número cresceu em alta escala, essa métrica também teve o seu aumento. Entretanto, desde a versão 8.7 este número vem se mantendo, o que é um bom sinal. Por fim, de acordo com os dados coletados da métrica *Average_weighted_method_count_by_class_(CC)*, como pode ser vista na figura 11, mostrou-se que no início da versão 7, se manteve num valor alto e ao chegar na versão 8, ocorreu uma queda brusca, significando um bom resultado, e até a versão 9.2 se manteve em baixa na mesma proporção. Dessa forma, a qualidade do código do Drupal vem se mantendo em um crescimento ordenado, logo, não existem indícios, seguindo o parâmetro qualidade do código fonte, para que usuários que utilizam a ferramenta migrem o seu sistema para outro CMS e a sua utilização atualmente continua viável.

Para um trabalho futuro, pensa-se em analisar outras métricas do sistema, melhorando assim a análise estatística, e uma integração da ferramenta Drupal Code Analysis com a plataforma RStudio para a coleta do sumário de cada métrica, geração dos gráficos e execução dos testes de forma mais automatizada. Além disso, uma comparação da qualidade do código fonte dos três principais CMSs do mercado (PATEL *et al.*, 2011), Drupal, Wordpress e Joomla.

REFERÊNCIAS

AL-FAR, Anas; QUSEF, Abdallah; ALMAJALI, Sufyan. Measuring Impact Score on Confidentiality, Integrity, and Availability Using Code Metrics. In: **2018 International Arab Conference on Information Technology (ACIT)**. IEEE, 2018. p. 1-9.

AMORIM, Rafael Tavares; FERREIRA, Jean. Construção de Sites com Drupal. **Anais do Salão Internacional de Ensino, Pesquisa e Extensão**, v. 3, n. 3, 2011.

AWODE, T. R. et al. Halstead Complexity Analysis of Bubble and Insertion Sorting Algorithms. 2017.

CARIGÉ, Rui Santos; DE FIGUEIREDO CARNEIRO, Glauco. Sentiment Polarity of Programmers in an Open Source Software Project: An Exploratory Study. In: **Proceedings of the 34th Brazilian Symposium on Software Engineering**. 2020. p. 147-156.

CHIDAMBER, Shyam R.; KEMERER, Chris F. A metrics suite for object oriented design. **IEEE Transactions on software engineering**, v. 20, n. 6, p. 476-493, 1994.

GRAYLIN, J. A. Y. et al. Cyclomatic complexity and lines of code: empirical evidence of a stable linear relationship. **Journal of Software Engineering and Applications**, v. 2, n. 03, p. 137, 2009.

LAKSHMINARAYANA, Anuradha; NEWMAN, Timothy S. Principal component analysis of lack of cohesion in methods (lcom) metrics. **Technical Report TRUAH-CS-1999-01**, 1999.

PATEL, Savan K.; RATHOD, V. R; PARIKH, Satyen. Joomla, Drupal and WordPress-a statistical comparison of open source CMS. In: **3rd International Conference on Trendz in**

Information Sciences & Computing (TISC2011). IEEE, 2011. p. 182-187.

Php Depend. **Php Depend Documentation.** Online. 2021. Disponível em: <https://pdepend.org/documentation/software-metrics>

PhpMetrics. **PhpMetrics Documentation.** Online. 2021. Disponível em: https://phpmetrics.org/index.html

SIRQUEIRA, Tassio Ferenzini Martins et al. Aplicação de Métodos Estatísticos em Engenharia de Software: Teoria e Prática, 2020.

WARNARS, Harco Leslie Hendric Spits et al. Object oriented metrics to measure the quality of software upon PHP source code with PHP_depend study case request online system application. In: **2017 International Conference on Applied Computer and Communication Technologies (ComCom)**. IEEE, 2017. p. 1-5.