PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Campus Barreiro - Curso de Sistema de informação

Disciplina: Processo e Qualidade de Software Período: 6º - 2º sem./2018

Aluno(s): Caio César e Carlos Eduardo

Relatório do desenvolvimento do Projeto de Processo e Qualidade de Software

1. Planejamento do projeto

1.1 Ideia proposta

A ideia inicial do trabalho é fazer uma análise mais profunda entre as métricas documentação e repositório,

1.2 GQM

Ao conversarmos e utilizarmos o método *goal question e metric* (GQM) chegamos ao seguinte modelo para realizar nossa análise:

Objetivo: O Objetivo deste estudo é caracterizar a relação entre um repositório de qualidade e a cobertura de linhas comentadas no código

Questão: A questão principal a ser respondida é: Qual a relação entre um repositório de qualidade e a cobertura de linhas comentadas do seu código?

Métricas: As métricas a serem analisadas serão: Quantidade de linhas de código, quantidade de linhas comentadas, quantidade de contribuintes no repositório e avaliação do repositório.

2. Desenvolvimento

2.1 Escolha dos repositórios

Para realizar a análise foi selecionado os repositórios com mais contribuintes no GitHub, analisado pela métrica “quantidade de

contribuintes” os projetos com mais contribuidores segundo o https://octoverse.github.com/ :

1. https://github.com/Microsoft/vscode
2. https://github.com/facebook/react-native
3. https://github.com/npm/cli
4. https://github.com/angular/angular-cli
5. https://github.com/tensorflow/tensorflow
6. https://github.com/FortAwesome/Font-Awesome
7. https://github.com/angular/angular
8. https://github.com/moby/moby
9. https://github.com/jlord/patchwork
10. https://github.com/ansible/ansible

2.2 Busca dos dados

Após escolher os repositórios, teríamos que pensar em como iriamos buscar os dados do repositório, após algumas pesquisas, descobrimos uma ferramenta chamada CLOC, uma ferramenta que roda no diretório raiz do repositório, e então devolve alguns dados, como a quantidade de linhas de código, a quantidade de linhas em branco, a quantidade de linhas comentadas, e entre outras diversas funções da ferramenta.

```
C:\Users\Caio\Downloads>cloc-1.78.exe C:\Users\Caio\Downloads\cloc-master\cloc-master
     387 text files.
     375 unique files.
      78 files ignored.

github.com/AlDanial/cloc v 1.78  T=0.50 s (728.0 files/s, 95728.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Perl                             6           1472           2451          22191
YAML                           189              6            189           4304
Markdown                         1            232             26           2261
ANTLR Grammar                    2            200             59           1012
R                                3             95            312            698
C/C++ Header                     2            191            780            618
C++                              4            132            173            570
Forth                            2             17             84            529
TypeScript                       4             53             39            416
Logtalk                          1             59             57            368
Windows Message File             2             89              9            348
```

2.3 Analise dos dados

Com os dados em mãos, então partimos para a análise dos dados brutos, para essa análise, fizemos uma planilha no Excel, onde continha uma tabela com as informações buscadas do CLOC, dos repositórios selecionados.

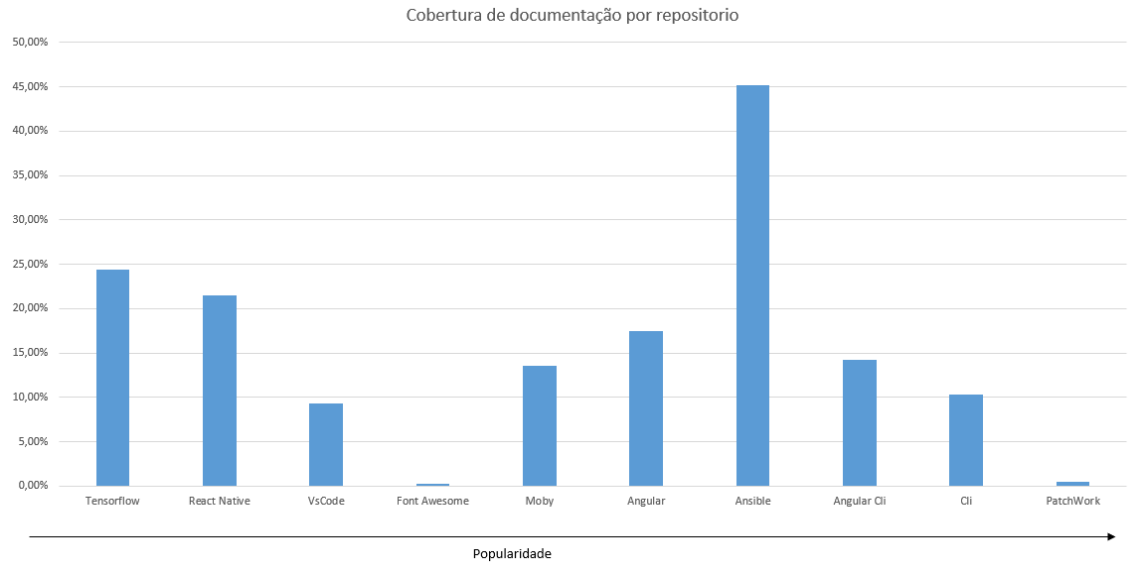
Repo	Arq	Language	Branco	Comentadas	Code	Cobertura	Stars	Forks
Angular Cli	800	TypeScript	9537	9999	51258	19,51%	20326	5135
Angular Cli	135	Markdown	2315	0	9978	0,00%	20326	5135
Angular Cli	192	JSON	46	0	9469	0,00%	20326	5135
Angular Cli	32	JavaScript	109	169	786	21,50%	20326	5135
Angular Cli	16	HTML	43	30	323	9,29%	20326	5135
Angular Cli	3	YAML	32	43	257	16,73%	20326	5135
Angular Cli	5	EJS	47	47	224	20,98%	20326	5135
Angular Cli	1	Skylark	10	16	54	29,63%	20326	5135
Angular Cli	2	Windows Resource File	7	0	21	0,00%	20326	5135
Angular Cli	2	Sass	2	4	12	33,33%	20326	5135
Angular Cli	4	CSS	0	9	3	300,00%	20326	5135
Angular	3544	TypeScript	55383	63437	306037	20,73%	42880	10896
Angular	184	Markdown	20817	0	35450	0,00%	42880	10896
Angular	314	JSON	281	0	19796	0,00%	42880	10896
Angular	372	JavaScript	2936	3644	16991	21,45%	42880	10896
Angular	359	HTML	1209	985	8004	12,31%	42880	10896
Angular	79	CSS	823	150	4590	3,27%	42880	10896
Angular	49	Sass	657	70	3474	2,01%	42880	10896
Angular	73	Bourne Shell	567	483	2129	22,69%	42880	10896
Angular	16	Skylark	300	734	1392	52,73%	42880	10896
Angular	4	YAML	80	159	781	20,36%	42880	10896
Angular	6	Bourne Again Shell	38	17	204	8,33%	42880	10896
Angular	2	Dockerfile	43	44	156	28,21%	42880	10896
Angular	4	Windows Resource File	36	0	149	0,00%	42880	10896
Angular	1	Pascal	40	130	109	119,27%	42880	10896
Angular	2	XML	0	0	16	0,00%	42880	10896
Ansible	3671	Python	150234	357053	543088	65,74%	33906	13552
Ansible	2842	YAML	30597	11326	188259	6,02%	33906	13552
Ansible	685	JSON	7	0	37812	0,00%	33906	13552
Ansible	159	PowerShell	3522	2169	18229	11,90%	33906	13552
Ansible	8	CSS	3063	12	13044	0,09%	33906	13552
Ansible	30	XML	7	0	12583	0,00%	33906	13552
Ansible	52	Markdown	1587	0	3679	0,00%	33906	13552
Ansible	102	Bourne Shell	691	464	1779	26,08%	33906	13552
Ansible	19	HTML	176	59	1240	4,76%	33906	13552
Ansible	37	INI	247	0	1193	0,00%	33906	13552

Para analisar os dados o primeiro passo foi gerar uma tabela com os dados brutos, porém divido por repositório, e então ordenar ele, por avaliação, que no caso seria a coluna “Popularidade”.

	Valores						
Repositorio	Arquivos	Linguagens no Repositorio	Linhas de Código	Linhas Comentadas	Popularidade	Média de Forks	Cobertura de Documentacao
Tensorflow	9.067	31	1.904.537	464.010	115.103	69.923	24,36%
React Native	2.938	23	269.578	58.060	71.216	15.949	21,54%
VsCode	3.034	44	748.728	69.814	64.480	8.563	9,32%
Font Awesome	2.689	8	149.121	394	57.973	9.866	0,26%
Moby	5.031	16	975.408	132.313	51.222	14.883	13,56%
Angular	5.009	15	399.278	69.853	42.880	10.896	17,49%
Ansible	7.630	19	822.278	371.180	33.906	13.552	45,14%
Angular Cli	1.192	11	72.385	10.317	20.326	5.135	14,25%
Cli	3.080	19	327.424	33.921	842	200	10,36%
PatchWork	7	4	403	2	794	21.936	0,50%
Total Geral	**39677**	**190**	**5669140**	**1209864**	**57.141**	**19.985**	**21,34%**

Com isso já tínhamos os repositórios ordenados por avaliação, o que seria a métrica “avaliação do repositório”, e teríamos também a “quantidade de linhas comentadas”, e “quantidade de linhas de código”, ao fazer uma relação dessas métricas, geramos uma coluna chamada “Cobertura de documentação” que mostra quantos porcentos as linhas de código comentada representam naquele repositório.

Após gerar a tabela, para facilitar a analise e chegarmos a uma conclusão foi gerado um gráfico de barras, onde as linhas horizontais estão ordenadas por avaliação do repositório



3. Conclusão

Com o gráfico gerado e a analise feita, chegamos a uma conclusão, de que, com base repositórios analisados, para um repositório ter uma avaliação alta, não necessariamente ele precisa de ter uma cobertura de código comentado muito grande, outros fatores que acabam levando ele a ter uma qualidade alta, mas, nos casos analisados, a cobertura de linha de código, não interfere nessa avaliação.

4. Recomendações

Uma das coisas que percebemos ao desenvolver essa análise, é que, talvez, a analise teria um resultado melhor, analisando os repositórios com maior avaliação do GitHub, ao invés dos repositórios com mais contribuintes