PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA
UNIDADE EDUCACIONAL PRAÇA DA LIBERDADE
Bacharelado em Engenharia de Software

Mateus Santos Fonseca

Relatório de Laboratório de Experimentação de Software

Belo Horizonte
2020

1. Introdução

Tendo como contexto a análise de repositórios do GitHub, este trabalho foi proposto na disciplina de Laboratório de Experimentação de Software com a finalidade de verificar as características dos repositórios dos sistemas populares no GitHub.

Hipóteses e justificativas:

A. Sistemas populares são maduros/antigos?

Métrica: idade do repositório (calculado a partir da data de sua criação).

Hipótese: Tendo em vista que o GitHub possui 12 anos de idade (contando apenas os anos), temos como hipótese que para um repositório ser maduro ele deve ter ao menos 60% da idade do GitHub, ou seja, aproximadamente 7 anos.

B. Sistemas populares recebem muita contribuição externa?

Métrica: Total de Pull Requests aceitas.

Hipótese: A popularidade de um repositório é diretamente proporcional à quantidade de Pull Requests. Tendo isto como base, podemos dizer que a quantidade de Pull Requests de sistemas populares deve ser superior a 500.

C. Sistemas populares lançam releases com frequência?

Métrica: Total de releases

Hipótese: Tendo em vista que um software popular possui, normalmente, uma idade maior e que a idade é diretamente proporcional à quantidade de commits e, consequentemente, funcionalidades/melhorias. Tomamos como hipótese que uma quantidade razoável de releases em um sistema popular é de no mínimo 3 releases/ano.

D. Sistemas populares são atualizados com frequência?

Métrica: Tempo até a última atualização (calculado a partir da data de última atualização)

Hipótese: Como hipótese, temos que o que mantém um sistema popular é sua resposta aos feedbacks dos usuários. Tendo isto como base, torna-se necessário a manutenção

deste sistema em uma frequência, no mínimo, regular. Portanto, um repositório deve ser atualizado ao menos uma vez ao mês.

E. Sistemas populares são escritos nas linguagens mais populares?

Métrica: Linguagem primária de cada um desses repositórios.

Hipótese: Tendo em vista que as linguagens populares são as que possuem maior comunidade. Podemos afirmar que a popularidade de um sistema está diretamente proporcional à popularidade de sua linguagem primária. Neste contexto, Javascript, como base para frameworks populares atualmente como React e Angular, Java e Python devem estar figurando entre as linguagens mais utilizadas nestes repositórios mais populares.

F. Sistemas populares possuem um alto percentual de issues fechadas?

Métrica: Razão entre número de issues fechadas pelo total de issues.

Hipótese: Tendo em vista que um sistema popular busca melhorias e atualizações contínuas, torna-se essencial para estas continuarem evoluindo o software, resolvendo e fechando as issues abertas o mais rápido, cativando sua comunidade. Temos como hipótese que 75% das issues de um repositório popular devem estar fechadas.

G. Questão extra: Sistemas escritos em linguagens mais populares recebem mais contribuição externa, lançam mais releases e são atualizados com mais frequência?

Métrica: De acordo com as métricas das questões B, C e D. Replicar a fórmula filtrando pelas 4 linguagens mais populares encontrada na questão E.

Hipótese: Tendo em vista que se trata de um repositório com linguagem popular, hipoteticamente teremos um maior engajamento da comunidade para com estas linguagens, uma vez que existem mais adeptos delas na própria comunidade. Portanto, temos como hipótese que a contribuição externa, quantidade de releases e frequência de updates são diretamente proporcionais à popularidade da linguagem, uma vez que quanto mais usuários adeptos desta linguagem, maior sua comunidade e, consequentemente, maior engajamento dela nos repositórios nesta linguagem. Para referência de confirmação desta hipótese, teremos que, para cada linguagem, a

quantidade de Pull Requests Aceitas deve ser de pelo menos metade das Pull Requests Totais, a quantidade de releases/ano deve ser de pelo menos 3 e atualização deve ser no mínimo mensal.

2. Metodologia

Para elaborar este trabalho de análise, foi desenvolvido um script em Python que, através de uma query no formato da linguagem de consulta GraphQL, consumia a API do GitHub e, através de paginação, retornava os dados relevantes para a resolução das questões propostas. Ao todo, são recolhidos 1000 repositórios em uma execução do script. Estes repositórios são, posteriormente, exportados para CSV no próprio script.

Para visualização do código fonte e do arquivo .csv base utilizado para a análise feita neste trabalho segue o link do meu repositório no GitHub: https://github.com/MateusSantosFonseca/Analise-Repositorios-GitHub-Labex.

3. Resultados obtidos

A. Mediana de 5 anos (arredondado).

B. Mediana de Pull Requests Aceitas de 284.

C. Mediana dos releases é de 37,195. Arredondando temos 37 releases. Como temos que analisar a quantidade de releases por ano, temos a mediana do ano igual 5, como calculado na questão A. Portanto 37/5 = 7,4 releases por ano.

D. A mediana do tempo entre atualizações é de menos de um dia. Portanto, os repositórios são atualizados diariamente.

E. Linguagens mais utilizadas de acordo com os repositórios populares analisados (mostrando apenas as 4 mais utilizadas): JavaScript: 302, Python: 94, Java: 71, Go: 69.

F. A mediana encontrada para a quantidade total de issues é igual a 811,5. Para issues fechadas, a mediana é igual a 635,5. A razão entre issues fechadas e issues totais é igual a 0,7831, ou seja, 78,31% das issues totais já foram resolvidas e fechadas.

G. Resultados obtidos para cada uma das quatro principais linguagens dos repositórios analisados:

JavaScript: Mediana de Pull Requests Aceitas de 303, Mediana Pull Requests Totais de 555,5, Mediana de Releases/ano de 3,4 e é atualizado diariamente.

Python: Mediana de Pull Requests Aceitas de 354, Mediana Pull Requests Totais de 481,5, Mediana de Releases/ano de 0,7, e é atualizado diariamente.

Java: Mediana de Pull Requests Aceitas de 179, Mediana Pull Requests Totais de 392, Mediana de Releases/ano de 1,2, e é atualizado diariamente.

Go: Mediana de Pull Requests Aceitas de 661 Mediana Pull Requests Totais de 903, Mediana de Releases/ano de 5,6, e é atualizado diariamente.

4. Análise dos resultados em relação à hipótese

A. A hipótese inicial foi refutada, uma vez que o resultado demonstrou que a mediana dos sistemas populares é de 5 anos, sendo dois anos inferiores aos 7 previstos. Portanto sistemas populares não são maduros.

B. A hipótese inicial foi refutada, uma vez que o resultado demonstrou que a mediana de Pull Requests Aceitas foi de 284, pouco mais da metade. Portanto sistemas populares não recebem muita contribuição externa.

C. A hipótese inicial foi confirmada, uma vez que a mediana de releases/ano é de 7,4, sendo maiores que os 3/ano previstos. Portanto sistemas populares lançam releases com maior frequência.

D. A hipótese inicial foi confirmada, uma vez que os sistemas populares são atualizados, de acordo com a mediana, diariamente. Portanto sistemas populares são atualizados frequentemente.

E. A hipótese inicial foi confirmada, uma vez que o resultado demonstrou que, em ordem decrescente, JavaScript, Python e Java estavam figurando entre as linguagens primárias mais utilizadas nos repositórios populares. Portanto, sistemas populares são escritos nas linguagens mais populares.

F. A hipótese inicial foi confirmada, uma vez que a porcentagem da razão entre as medianas das issues fechadas e issues totais é de 78%, superando os 75% da hipótese inicial. Portanto sistemas populares possuem alto percentual de issues fechadas.

G. **JavaScript:** A hipótese para esta linguagem foi confirmada, pois javascript tem mais da metade (330) de Pull Requests Aceitas em relação aos totais, possui mais releases por ano do que 3 (3,4) e é atualizada diariamente.

Python: A hipótese para esta linguagem foi refutada, pois apesar de Python ter mais da metade (354) de Pull Requests Aceitas em relação aos totais e ser atualizada diariamente, ela possui menos releases por ano do que 3 (0,7).

Java: A hipótese para esta linguagem foi refutada, pois Java tem pouco mais de 1 release por ano (1,2), portanto menor que 3 e ela tem menos da metade (179) de Pull Requests Aceitas em relação aos totais.

Go: A hipótese para esta linguagem foi confirmada, pois Go tem mais da metade (661) de Pull Requests Aceitas em relação aos totais, possui mais releases por ano do que 3 (5,6) e é atualizada diariamente.