Alunos: Aureliano Maia

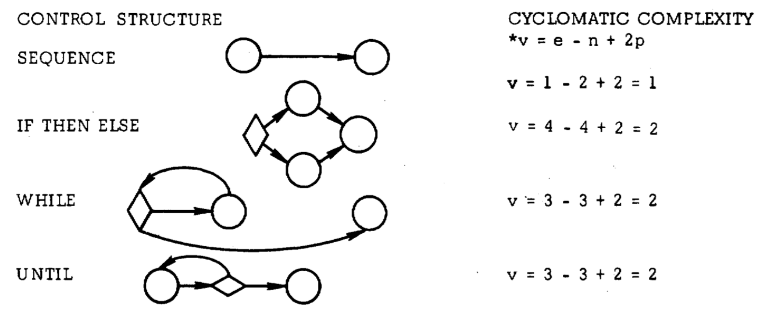
Marcos Paulo

**Analise de repositórios do github na linguagem JAVA**

A compreensão, avaliação e controle de códigos são fatores que podem influenciar na popularidade de um repositório. Podemos então entender que o grande fator que pode influenciar na popularidade do repositório é a complexidade do mesmo. Quanto maior a complexidade de um repositório, maior é o esforço para compreender, modificar e evoluir esse repositório.

Neste trabalho, adotamos uma abordagem experimental que tem como objetivo analisar 15 repositórios java dentre uma relação de uma lista dos 30 repositórios mais populares do github, conforme pesquisa realizado através do google. É através dessa análise que tentaremos responder a questão criada no GQM, “Como a popularidade pode afetar a manutenibilidade do código?”, no contexto, “*Análise da relação entre a popularidade do repositório e a facilidade na correção e refatoração de seu código”.*

Para o desenvolvimento deste trabalho, utilizamos trechos de código em java, que realiza uma análise do código do repositório, em que ao final da análise temos em resposta a soma das linhas, e a quantidade de classes do código. Utilizamos também códigos em PHP que retorna a quantidade de linhas do manual e também a complexidade ciclomática do código (if, else, switch, for, etc.).



Por fim utilizamos a API do github que nos retorna à quantidade de estrelas e a quantidade de forks, o que demonstra a quantidade de colaboradores e também popularidade do repositório.

Com isso também analisaremos a influência de desenvolvedores que contribuem na evolução do repositório através do fork sob a popularidade do mesmo.

Após análise dos resultados obtidos, criamos tabelas e gráficos:

–Todos as métricas estudadas podem influenciar na complexidade dos repositórios e com isso influenciar na sua popularidade. Porém a análise não pode ser conclusiva quando utilizamos somente uma métrica para analisar o repositório e falarmos da dificuldade em sua manutenção. Como por exemplo:

****

A quantidade de linhas sozinhas divididas pelo número de forks nos levaria a entender que o código seria de fácil manutenibilidade, pois se dividirmos a quantidade de linha pela quantidade de colaboradores- forks-, o resultado seria 138 linhas por colaboradores.

Porém se analisarmos a complexidade ciclomática demonstrada nas condicionais podemos ver que a manutenibilidade seria um pouco mais difícil.

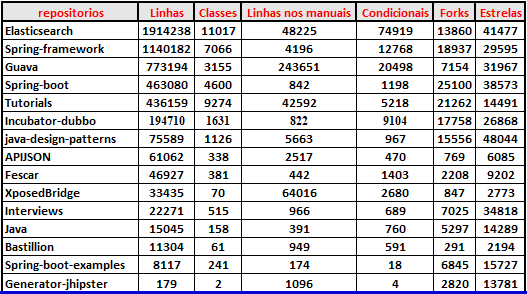
– Em cada repositório estudado, a variação na dificuldade e por consequência a variação de sua popularidade foi influenciada por um conjunto diferente de métricas, como demonstrado no exemplo acima.

– Além disso, dentro de um mesmo repositório, as mudanças que aumentam a dificuldade na manutenção e as mudanças que reduzem essas dificuldades, foram influenciadas por conjuntos diferentes de métricas. Como por exemplo: Quantidade de linhas de código x forks.

Quantidade de linhas de código x linhas dos manuais

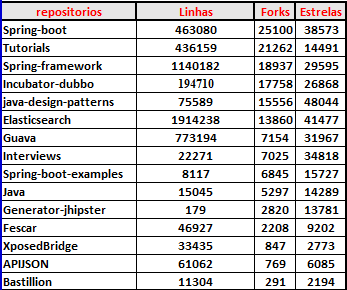
A facilidade na correção do código e na refatoração pode aumentar ou diminuir em função de diferentes fatores, como o nível de maturidade do projeto, o aumento no tamanho do seu código fonte, o tipo de manutenção ao qual o projeto é submetido, práticas de desenvolvimento como criação e controle do manual, comentários feitos no código, o que consequentemente aumenta a popularidade ou diminui.

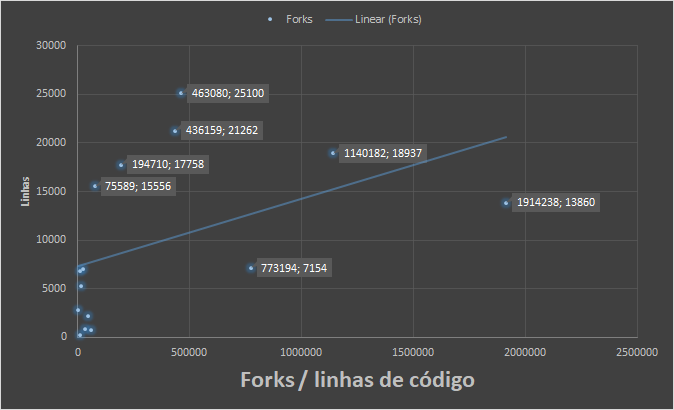
Abaixo se encontra uma tabela contendo os dados recolhidos no início do trabalho e é através desses dados que iremos trabalhar.



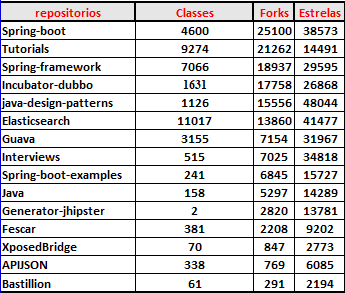
As tabelas e gráficos a seguir demonstram o teste realizado com as métricas, quantidades de linhas, quantidade de classes condicionais para analisarmos o nível de dificuldade do repositório e também as métricas forks e estrelas, onde analisamos a popularidade.

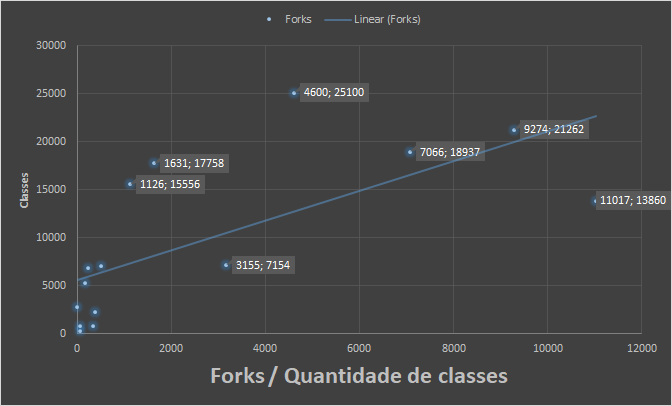
Quantidade de forks por linhas



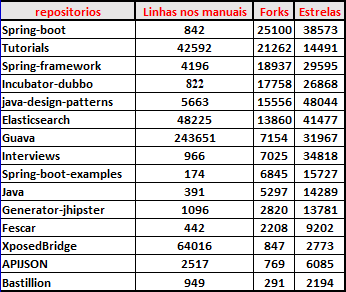


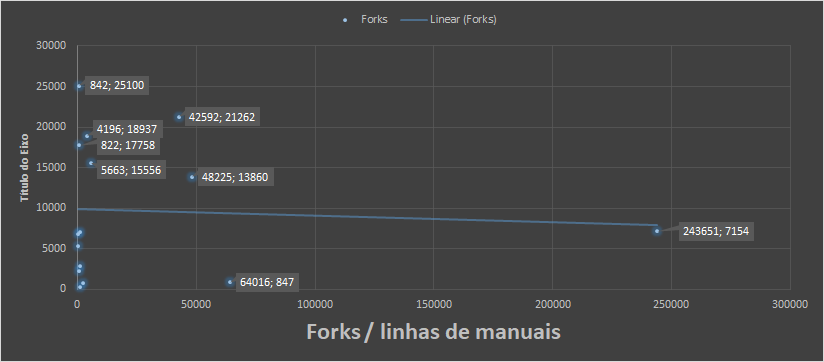
Quantidade de forks por classes



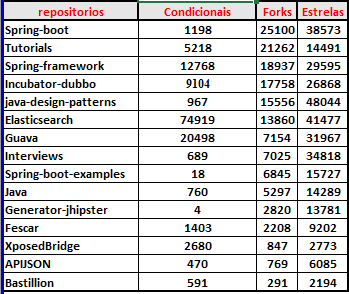


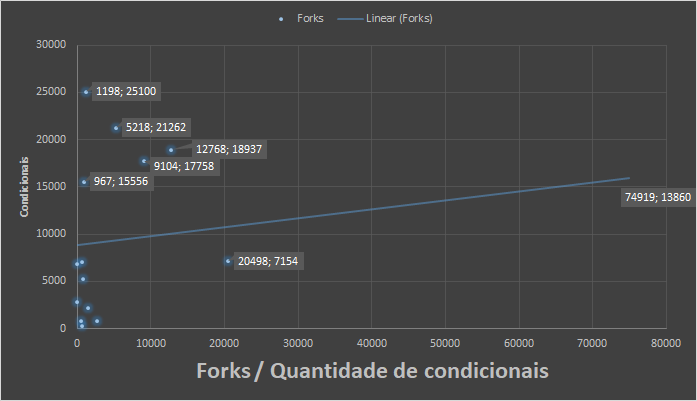
Quantidade de forks por linhas no manual





Quantidade de forks por condicionais





Ao analisarmos os dados em conjunto, concluímos que enquanto o repositório está no seu no master inicial, a popularidade do mesmo só tem a crescer, com isso surgem novos colaboradores forks, e todos os dias recebem estrelas e sua popularidade vai crescendo porém, quando o repositório atinge uma quantidade grandiosa de colaboradores e seu tamanho em linhas aumenta, junto aumenta também sua complexidade ciclomática e classes, necessitando assim de um maior detalhamento do código atraves de comentarios e por consequência cria-se a necessidade de incrementar ainda mais o manual ou manuais do repositório.

Fizemos também um experimento, que consistia em observar um período de tempo e analisar o crescimento na popularidade dos repositósitorios. Observarmos que os repositórios que estão com o menor tempo no github crescem mais rápido e recebem mais estrelas e forks pois ainda estão com uma complexidade baixa, já os repositórios que estão a mais tempo no github e já atingiram um nível de complexidade maior em sua complexidade, tende a receber menos estrelas e forks.



O que podemos observar nos gráficos e com os dados obtidos é que a medida que o código vai crescendo, cresce também os colaboradores e os seguidores, o que também poderá fazer com que até mesmo o seu criador deixe de incrementar o código, e passe somente a analisar as solicitações de aceite das alterações. O projeto pode até perder sua essência deixando de ser o que o proprietário imaginava e com isso o número de colaboradores e seguidores diminuem em velocidade e quantidade ao seu crescimento.

Todos os arquivos e informações sobre este trabalho podem ser encontradas no repositório github: https://github.com/marcospauloferrereira/analisederepositorioII