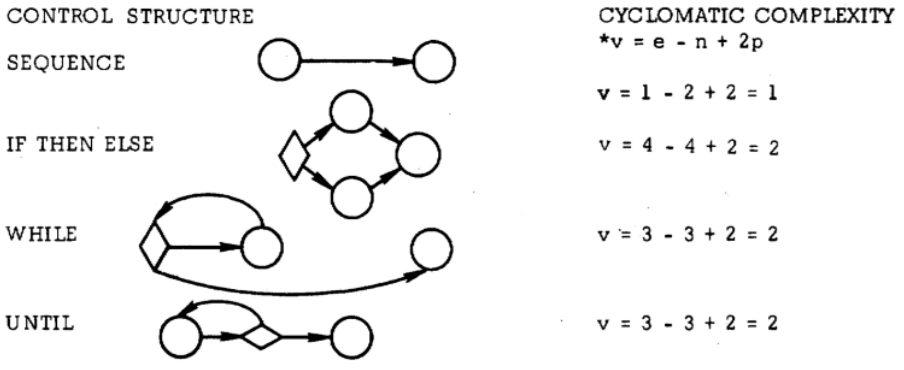
Alunos: Aureliano Maia
Marcos Paulo

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. Eh através dessa análise que tentaremos responder a questão criada no GQM, “Como a popularidade dos repositórios podem determinar a dificuldade em realizar manutenção no 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 a 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 para demonstrar melhor os resultados e chegamos à seguinte conclusão:

–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:

repositorios	Linhas	Classes	Linhas nos manuais	Condicionais	Forks	Estrelas
Elasticsearch	1914238	11017	48225	74919	13860	41477

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 do que imaginamos.

– 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.

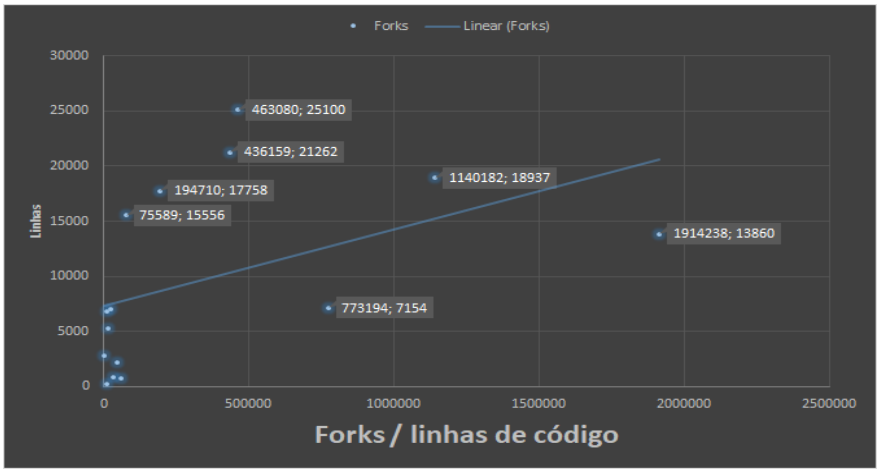
Abaixo se encontra uma tabela contendo os dados recolhidos no início do trabalho e é através desses dados que iremos trabalhar.

repositorios	Linhas	Classes	Linhas nos manuais	Condicionais	Forks	Estrelas
Elasticsearch	1914238	11017	48225	74919	13860	41477
Spring-framework	1140182	7066	4196	12768	18937	29595
Guava	773194	3155	243651	20498	7154	31967
Spring-boot	463080	4600	842	1198	25100	38573
Tutorials	436159	9274	42592	5218	21262	14491
Incubator-dubbo	194710	1631	822	9104	17758	26868
java-design-patterns	75589	1126	5663	967	15556	48044
APIJSON	61062	338	2517	470	769	6085
Fescar	46927	381	442	1403	2208	9202
XposedBridge	33435	70	64016	2680	847	2773
Interviews	22271	515	966	689	7025	34818
Java	15045	158	391	760	5297	14289
Bastillion	11304	61	949	591	291	2194
Spring-boot-examples	8117	241	174	18	6845	15727
Generator-jhipster	179	2	1096	4	2820	13781

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.

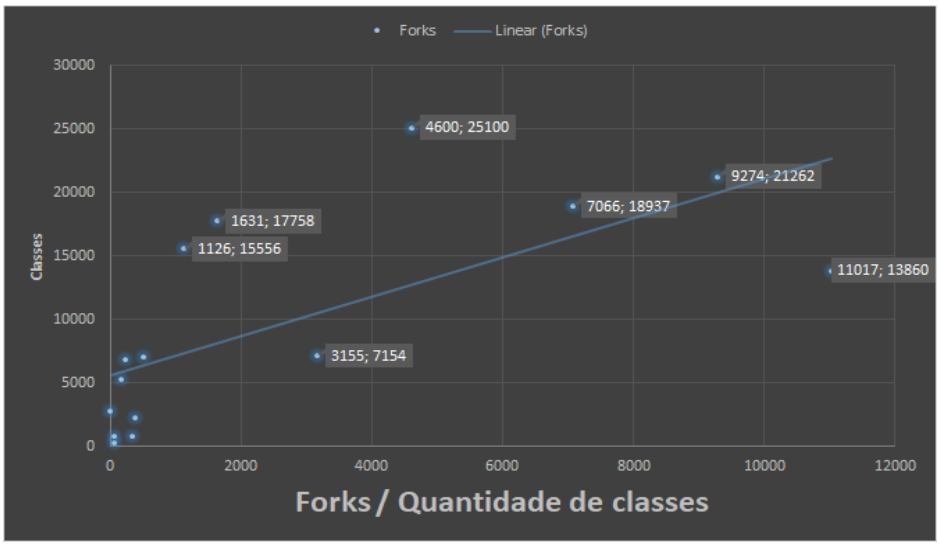
Quantidade de linhas x forks

repositorios	Linhas	Forks	Estrelas
Spring-boot	463080	25100	38573
Tutorials	436159	21262	14491
Spring-framework	1140182	18937	29595
Incubator-dubbo	194710	17758	26868
java-design-patterns	75589	15556	48044
Elasticsearch	1914238	13860	41477
Guava	773194	7154	31967
Interviews	22271	7025	34818
Spring-boot-examples	8117	6845	15727
Java	15045	5297	14289
Generator-jhipster	179	2820	13781
Fescar	46927	2208	9202
XposedBridge	33435	847	2773
APIJSON	61062	769	6085
Bastillion	11304	291	2194



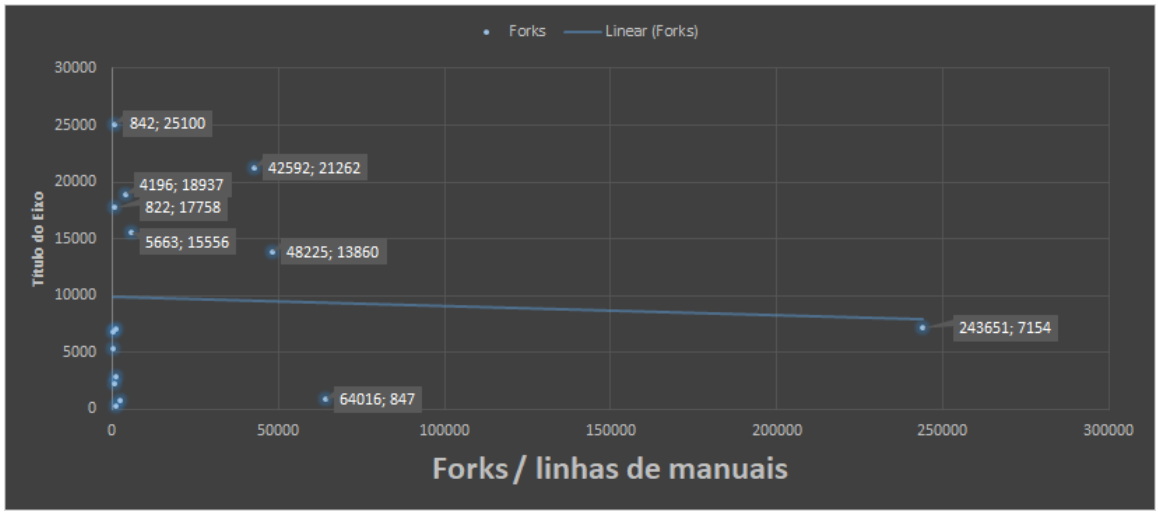
Quantidade de classes por forks

repositorios	Classes	Forks	Estrelas
Spring-boot	4600	25100	38573
Tutorials	9274	21262	14491
Spring-framework	7066	18937	29595
Incubator-dubbo	1631	17758	26868
java-design-patterns	1126	15556	48044
Elasticsearch	11017	13860	41477
Guava	3155	7154	31967
Interviews	515	7025	34818
Spring-boot-examples	241	6845	15727
Java	158	5297	14289
Generator-jhipster	2	2820	13781
Fescar	381	2208	9202
XposedBridge	70	847	2773
APIJSON	338	769	6085
Bastillion	61	291	2194



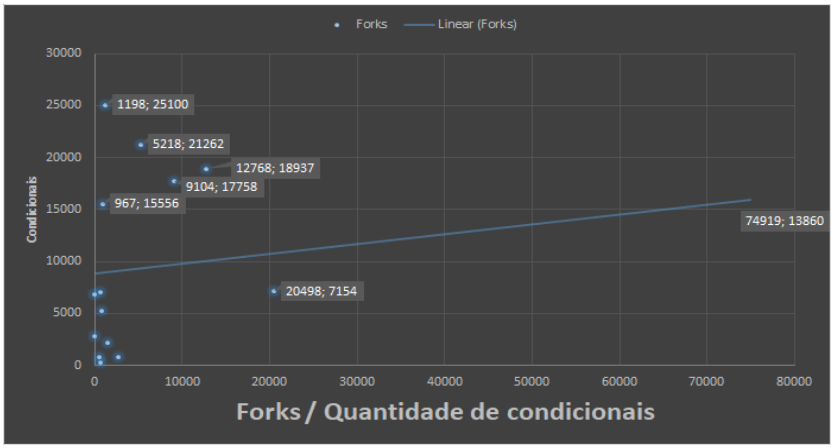
Quantidade de linhas no manual x fork

repositorios	Linhas nos manuais	Forks	Estrelas
Spring-boot	842	25100	38573
Tutorials	42592	21262	14491
Spring-framework	4196	18937	29595
Incubator-dubbo	822	17758	26868
java-design-patterns	5663	15556	48044
Elasticsearch	48225	13860	41477
Guava	243651	7154	31967
Interviews	966	7025	34818
Spring-boot-examples	174	6845	15727
Java	391	5297	14289
Generator-jhipster	1096	2820	13781
Fescar	442	2208	9202
XposedBridge	64016	847	2773
APIJSON	2517	769	6085
Bastillion	949	291	2194



Quantidade de condicionais x fork

repositorios	Condicionais	Forks	Estrelas
Spring-boot	1198	25100	38573
Tutorials	5218	21262	14491
Spring-framework	12768	18937	29595
Incubator-dubbo	9104	17758	26868
java-design-patterns	967	15556	48044
Elasticsearch	74919	13860	41477
Guava	20498	7154	31967
Interviews	689	7025	34818
Spring-boot-examples	18	6845	15727
Java	760	5297	14289
Generator-jhipster	4	2820	13781
Fescar	1403	2208	9202
XposedBridge	2680	847	2773
APIJSON	470	769	6085
Bastillion	591	291	2194



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.

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, fazendo 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.