Engenharia empírica: Padrões das issues do Github

Elaine C. Sangali, Noemi
Department of Computer Science
Federal University of Technology - Paraná / UTFPR
Campo Mourão, Paraná, Brazil
{e.nani92, noemischerer13}@gmail.com

July 2018

1 Introdução

Muitos projetos coletam *feedback* do usuário por meio de um rastreador de bugs central. O rastreador do GitHub é chamado de Issues e pode ser usado com todos os repositórios [1].

As issues também podem ser atribuídos a outros usuários, marcados com rótulos para uma pesquisa mais rápida e agrupados com *milestones*. Também é possível rastrear problemas duplicados usando uma resposta salva ou visitar o painel para encontrar rapidamente *links* para os problemas atualizados recentemente [1].

O objetivo desse trabalho é analisar issues de cinco projetos de cinco linguagens diferentes do Github a fim de verificar se elas estão padronizadas, de acordo com as especificações do próprio projeto.

2 Os projetos

Pra escolha das linguagens, considerou-se um artigo que especificava as mais utilizadas no Github. Já para os projetos, considerou-se os mais populares. A Figura 1 demostra os projetos escolhidos.

LINGUAGEM	NOME PROJET	ESTRELAS	HAKING
JAVASCRIPT	Free Code Cam	292k	1º
	Bootstrap	132k	2º
	React	92.6k	3º
	JavaScript	68.9k	6º
	Electron	58.5k	7º
PYTHON	Httpie	34.7k	3º
	The Fuk	34.6k	4º
	Flask	34.4k	5º
		33k	9º
	Requests	31.5k	7º
JAVA	RxJava	32k	1º
	Elasticsearch	29.9k	3º
	Guava	23.2k	4º
	Interviews	21.4k	6º
	Zxing	18k	7º
RUBY	Rails	39.2k	1º
	Jekyll	33.8k	2º
	Capistrano	10.3k	9º
	Homebrew	11.8k	7º
	Devise	18.5k	5º
PHP	Symfony	17.2k	3º
	Laravel - Frame	11.6k	9º
	HHVM	15.2k	5º
	Yii2	11.8k	8º
	Composer	13.9k	7º

Figura 1: As linguagens e projetos escolhidos.

Duas contas foram feitas para determinar o número de issues que deveriam ser analisadas. Primeiro foi utilizada uma calculadora amostral[1], cuja a população é a quantidade total de issues de todos os projetos e o um nível de confiança de 99%. Depois, para cada projeto, foi calculada a quantidade de issues de acordo com a o número de amostra obtidas, por meio da fórmula `=QUOTIENT(MULTIPLY(661,total issues do projeto ), soma de todas as issues)`. A Figura 2 mostra esses valores obtidos.

[1] http://www.publicacoesdeturismo.com.br/calculoamostral/

LINGUAGEM	NOME PROJETO	Nº ISSUES	AMOSTRAGEM
JAVASCRIPT	Free Code Camp	12107	53
	Bootstrap	17047	75
	React	5002	22
	JavaScript	766	3
	Electron	8302	36
PYTHON	Httpie	474	2
	The Fuk	410	1
	Flask	1420	6
		17206	76
	Requests	2595	11
JAVA	RxJava	2388	10
	Elasticsearch	15507	68
	Guava	2742	12
	Interviews	17	0
	Zxing	765	3
RUBY	Rails	11293	50
	Jekyll	3622	16
	Capistrano	1029	4
	Homebrew	1452	6
	Devise	3698	16
PHP	Symfony	10182	45
	Laravel - Framework	10407	46
	HHVM	5398	24
	Yii2	9889	43
	Composer	4851	21
TOTAL		148569	661

Figura 2: Quantidade de issues a ser analisada de cada projeto.

Para cada um dos projetos, foi analisado o padrão de issue e anotado em uma planilha no Google Planilhas. Por exemplo, para o projeto `Httpie` de `Python`, o que é obrigatório conter numa issue é: o comportamento esperado, comportamento atual, passos para reporduzir o problema e especificações da versão do projeto, do sistema utilizado ou hardware. Cada projeto é responsável por um padrão diferente.

3 As análises

Os projetos analisados foram separados aleatoriamente utilizando o site Random[2]. A análise consistiu em verificar se a issue aleatória do projeto está pa-

[2]https://www.random.org/

dronizada.

Para cada issue, foi anotado em uma planilha:

- o nome do projeto;
- o número da issue;
- uma descrição dos requisitos atendidos;
- se a issue está padronizada;
- números de requisitos atendidos de acordo com o padrão;
- o tipo da issue (bug, feature ou other)
- status da issue (aberto, fechado ou merged)
- se a pessoa que criou a issue deixou algum comentário que atendesse aos requisitos do padrão.

Para a análise das issues coletados, os dados foram inseridos no banco de dados, para obter a quantidade total de issues padronizadas e facilitar a a conclusão dos resultados.

4 Resultados

Foi analisado as issues de cinco projetos de cinco linguagens. Para cada linguagem foi feito uma analise dos dados utilizando a projeção de gráficos paralelos, afim de procurar padrões nos dados coletados.

Por meio da linguagem Javascript foi analisado cinco projetos: Bootstrap, Electron, Freecodecamp, Javascript e React. Os resultados dessas analises podem ser vistas na figura 3. É possível percerber que a maioria das issues não estão padronizadas e essas atingem um numero baixo de requisitos, sendo a maioria do tipo bug, fechadas e que o criador da issue não apresenta comentarios atendendo a mais requisitos.

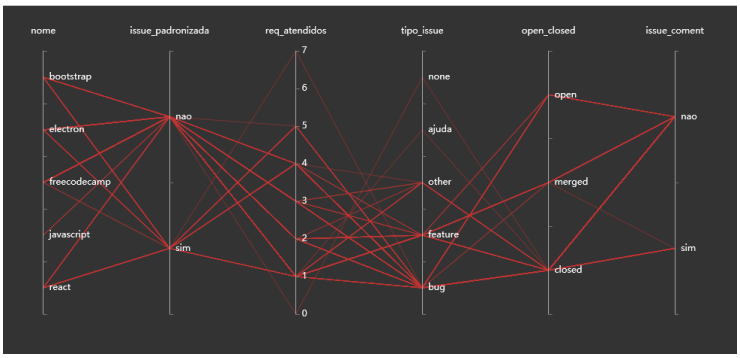

Figura 3: Análise dos projetos da linguagem Javascript.

Os cincos projetos analisados da categoria de linguagem de programação Java foram: Elasticsearch, Guava, Rx Java, Zxing. A figura 4 representa os dados coletados nesse projeto, sendo que a maioria deeles não estão padronizados, tendo uma baixa quantidade de requisitos atendidas, sendo a maioria das issues do tipo feature e bug, merged e fechada, e a maioria das issues não apresentam comentários que complementam os requisitos.

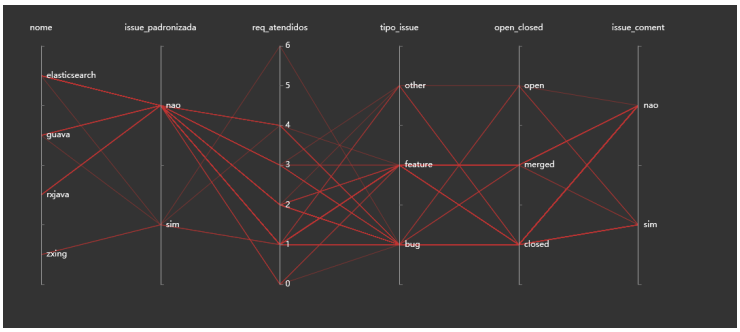

Figura 4: Análise dos projetos da linguagem Java.

Na linguagem de programação Python foram analisados os projetos: Flask, Httpie, Requests, The Fuk, como mostra a figura 5, assim como nas outras linguagens, esses projetos não estão padronizados, são bugs e features que estão fechadas ou mergiadas e que não estão comentadas com continuações de explicação do problema.

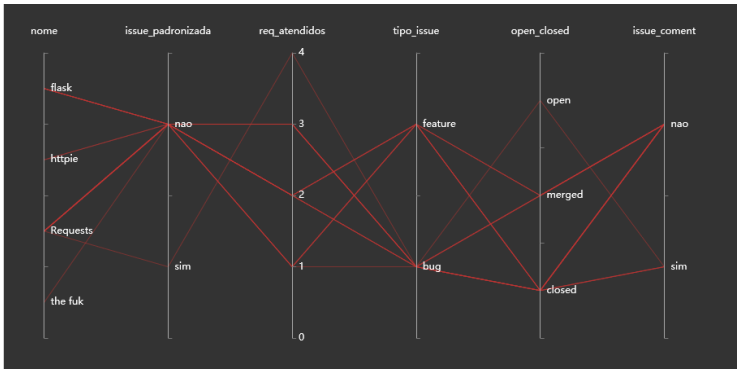

Figura 5: Análise dos projetos da linguagem Python.

Por meio da linguagem de programação Ruby foram analisados os projetos: Capistrano, Devise, Homebrew, Jeky, Rails como mostra a figura 6, assim como nas outras linguagens, os projetos não estão padronizados, são bugs e features que estão fechadas ou mergiadas e que não estão comentadas com continuações de explicação do problema.

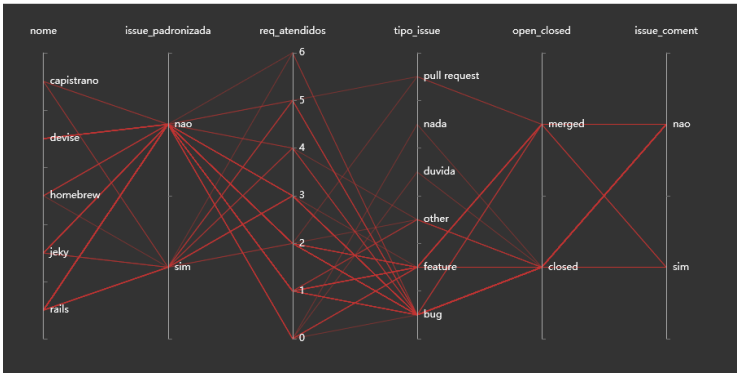

Figura 6: Análise dos projetos da linguagem Ruby.

Os projetos analisados da linguagem PHP foram: Composer, Framework, Hhvm, Symfony, Yii, é possível perceber que há uma diferença em relação as outras linguagens, a taxa de projetos padronizados e não padronizados está divida, assim como os outros dados como o tipo da issue, se o projeto está aberto ou fechado, e se a issue está comentada com resquisitos do padrão ou não (figura 7).

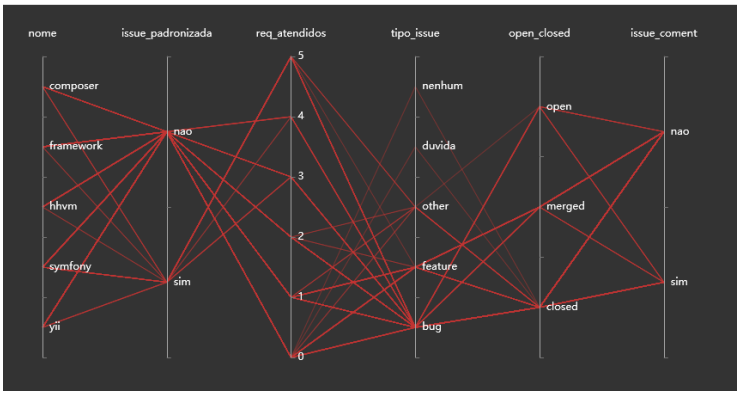

Figura 7: Análise dos projetos da linguagem PHP.

Por fim, foi realizada uma analise geral de todas as linguagens e, assim como na maioria das analises anteriores, grande parte das issues não estão padronizadas, o tipo das issues são de bugs, features e outras, a maioria das issues analisadas estavam fechadas, mas contem uma quantidade significativa aberta e *merged*, os comentários de requisitos nas issues estão divididos meio a meio (como mostra a figura 8).

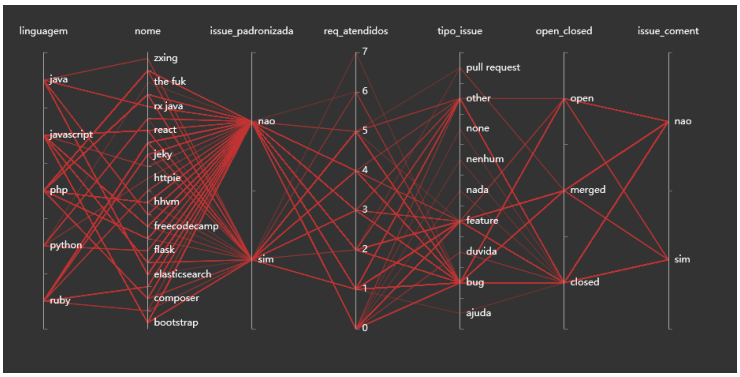

Figura 8: Análise das cinco linguagens.

5 Conclusão

Com a nossa pesquisa podemos comprovar que os padrões de issues não são seguidos pela maioria dos usuários, o próximo passo do trabalho seria investigar o porque de os usuários não seguirem esses padrões, realizando algumas perguntas com os usuários do tipo: Você sabe que existe um padrão de issue?, Você acha muito complicado seguir os padrões especificados? Ajudaria você se existisse um padrão global de issues para todos os projetos?. Com essas perguntas respondidas poderemos conseguir sugerir uma solução para o problema da falta de padrão nas issues.

Referências

[1] Github. About issues. https://help.github.com/articles/about-issues/.