Uma Análise da Co-Evolução de Teste em Projetos de Software no GitHub

Aluno: Charles José Lima de Miranda;
Orientador: Guilherme Amaral Avelino
Co-orientador: Pedro de Alcântara dos S Neto

Introdução

Cenário e motivação

O **software evolui** (LEHMAN, 1979) e essa **evolução envolve alterações** no código de produção para atividades de correções e evolução do software.

Cenário e motivação

O **software evolui** (LEHMAN, 1979) e essa **evolução envolve alterações** no código de produção para atividades de correções e evolução do software.

Como reduzir riscos?

Como garantir as funcionalidades?

Como garantir a confiabilidade?

Cenário e motivação

Teste de software

- Visa a identificação e correção de *bugs;*
- Aumenta a satisfação do usuário final;
- Garante a qualidade do Software;

Cenário e motivação

Teste de software

- Visa a identificação e correção de *bugs;*
- Aumenta a satisfação do usuário final;
- Garante a qualidade do Software;

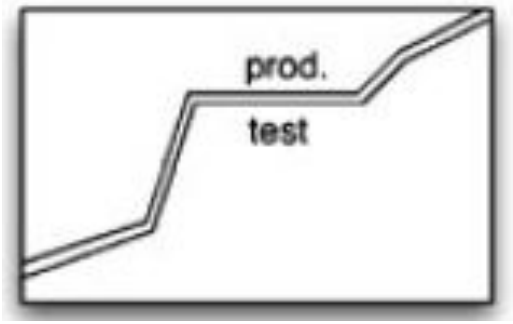
Como o teste é realizado no desenvolvimento de software?

Contexto e motivação

Teste de unidade

- É o teste de componentes de **forma isolada ou agrupamentos** coerentes de **funções** ou **classes**;
- Feito pelo desenvolvedor;
- **Existem diversos frameworks** que facilitam a automatização de testes de unidade.

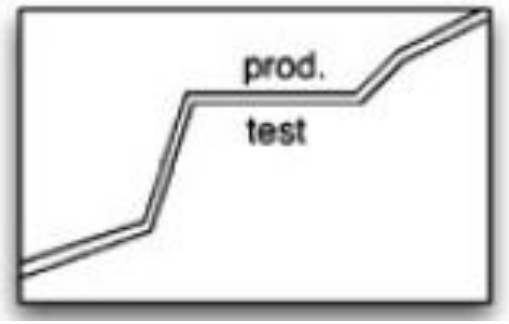
Contexto e motivação



Visto que **mudanças no código de produção** são necessárias;

Visto que **mudanças no código de teste** são necessárias;

Contexto e motivação



Visto que **mudanças no código de produção** são necessárias;

Visto que **mudanças no código de teste** são necessárias;

Para um processo de desenvolvimento visando qualidade o **código de teste também deve evoluir** junto com o código de produção (ZAIDMAN et al., 2011)

Contexto e motivação

Algumas pesquisas dessa área abordam o problema com foco **somente em Java**, e em **datasets muito pequenos**.

(ZAIDMAN et al., 2011)

(ROMANO; ZAIDMAN,2014)

(LEVIN; YEHUDAI, 2017)

(VIDÁCS; PINZGER, 2018)

Definição do problema

Como **investigar os impactos** da co-evolução de teste no desenvolvimento de projetos de software?

Questões de Pesquisa

QP1 - Como testes evoluem em projetos de software?

Questões de Pesquisa

QP1 - Como testes evoluem em projetos de software?

Motivação:

- Como evoluem?
  - Investigamos a adoção de testes;
  - Identificamos a existência de padrões de evolução de testes.

Questões de Pesquisa

QP2 - Com que frequência o código-fonte e testes co-evoluem em projetos de software?

Questões de Pesquisa

QP2 - Com que frequência o código-fonte e testes co-evoluem em projetos de software?

Motivação:

- Ocorre co-evolução?
  - Investigamos indícios de co-evolução.

Questões de Pesquisa

QP3 - Que características distinguem projetos onde há co-evolução de código de teste de projetos onde essa prática não é comum?

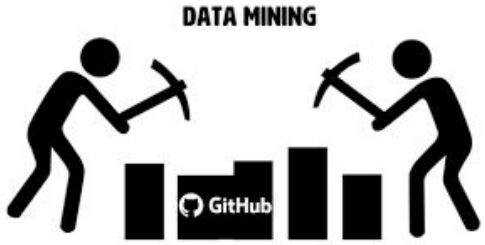
Questões de Pesquisa

QP3 - Que características distinguem projetos onde há co-evolução de código de teste de projetos onde essa prática não é comum?

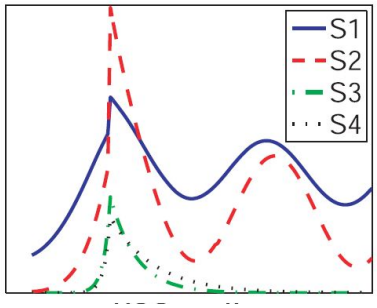
Motivação:

- São diferenças significativas?
  - Investigamos se existem diferenças

Visão geral da proposta



.[RegEx]*



Utilizamos conceitos relacionados a **mineração de repositórios de software**, busca de arquivos de teste através de **expressão regular** e realizamos a **clusterização** de dados para descoberta de padrões de crescimento.

Trabalhos relacionados

Trabalhos Relacionados

Autor	Foco da pesquisa
(MARSAVINA; ROMANO; ZAIDMAN, 2014)	Identificação de padrões de co-evolução de teste
(VIDÁCS; PINZGER, 2018)	Identificação de padrões de co-evolução de teste
(ZAIDMAN et al., 2011)	Visualização da co-evolução de teste
(LEVIN; YEHUDAI, 2017)	Co-evolução de teste e mudanças semânticas
(GONZALEZ et al., 2017)	Manutenção de software e teste de unidade

Trabalhos Relacionados

Autor	Foco da pesquisa
(MARSAVINA; ROMANO; ZAIDMAN, 2014)	Identificação de padrões de co-evolução de teste
(VIDÁCS; PINZGER, 2018)	Identificação de padrões de co-evolução de teste
(ZAIDMAN et al., 2011)	Visualização da co-evolução de teste
(LEVIN; YEHUDAI, 2017)	Co-evolução de teste e mudanças semânticas
(GONZALEZ et al., 2017)	Manutenção de software e teste de unidade

Trabalhos Relacionados

Amostra de frameworks encontrados (Gonzalez et al., 2017)

C#	NUnit	443
	Moq	164
	Rhino Mocks	30
Java	JUnit	3841
	Mockito	573
	TestNG	303
JavaScript	Mocha	6714
	JSUnit	758
	Enhance JS	737
Perl	Test::More	49
	Test::Builder	2
PHP	PHPUnit	1000
	PHP Unit Testing Framework	867
	SimpleTest	348

Trabalhos Relacionados

C#	NUnit	443
	Moq	164
	Rhino Mocks	30
Java	JUnit	3841
	Mockito	573
	TestNG	303
JavaScript	Mocha	6714
	JSUnit	758
	Enhance JS	737
Perl	Test::More	49
	Test::Builder	2
PHP	PHPUnit	1000
	PHP Unit Testing Framework	867
	SimpleTest	348

Gonzalez et al. (2017)

- A abordagem utiliza mineração de repositório de software, **identificação** de arquivos de teste e **detecção de padrões de teste**.

- Os resultados obtidos mostram que os projetos open source não adotam padrões de teste que podem ajudar com atributos de manutenibilidade. Foi verificado que **projetos menores aplicam padrões com mais frequência** e que a **adoção de padrões é uma decisão individual do desenvolvedor.**

Trabalhos Relacionados

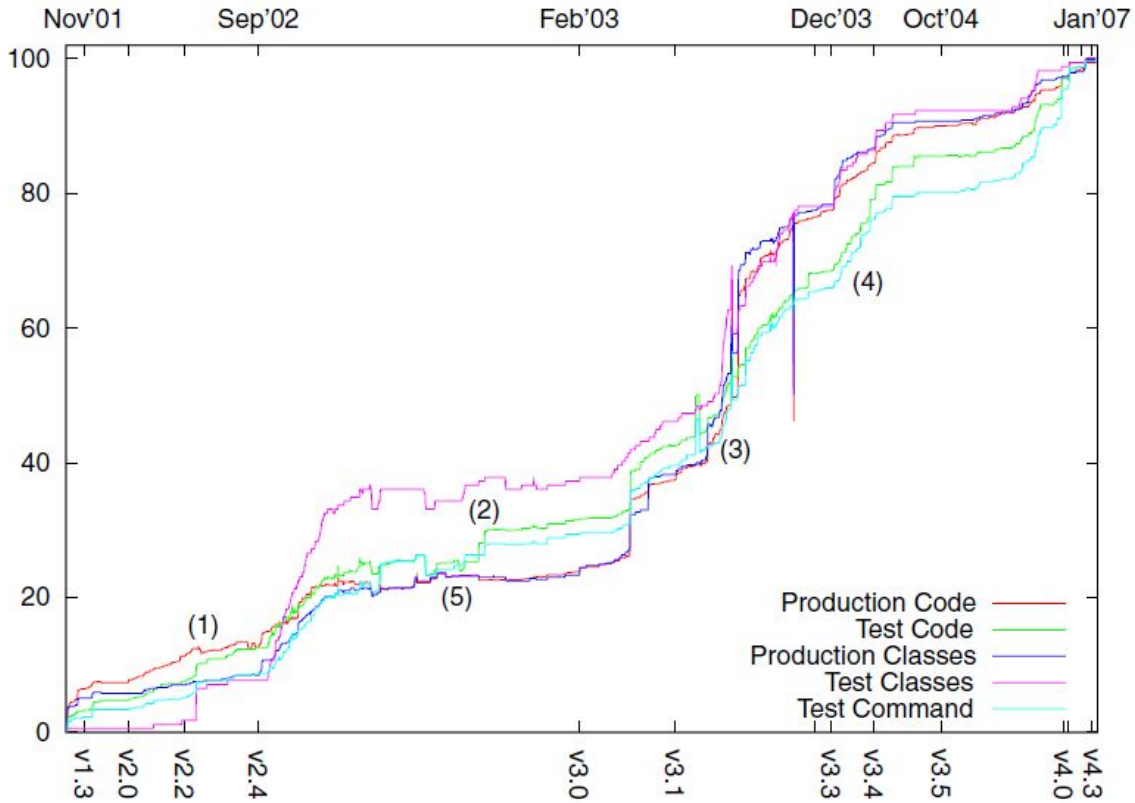


Fig. 5 Checkstyle Growth History View

Visualização do Histórico de crescimento (Zaidman et al., 2011)

Trabalhos Relacionados

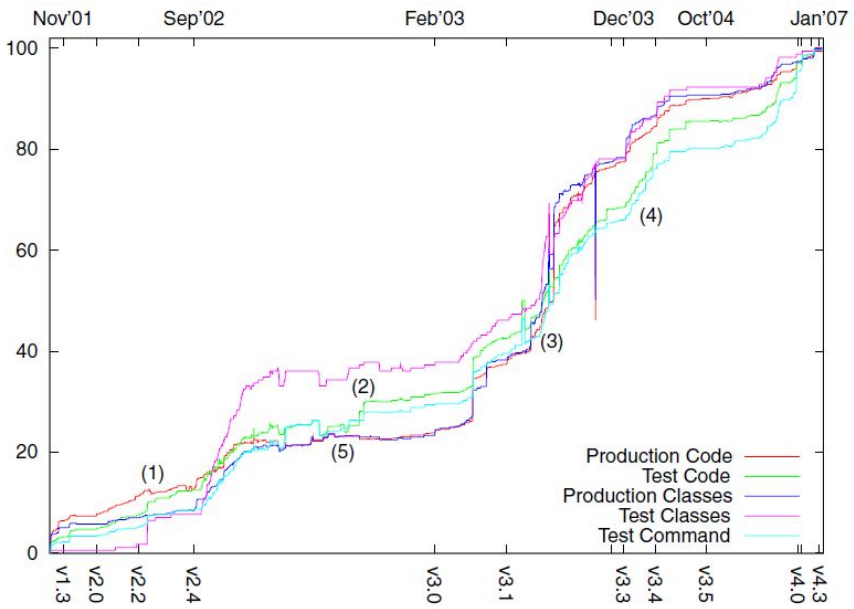


Fig. 5 Checkstyle Growth History View

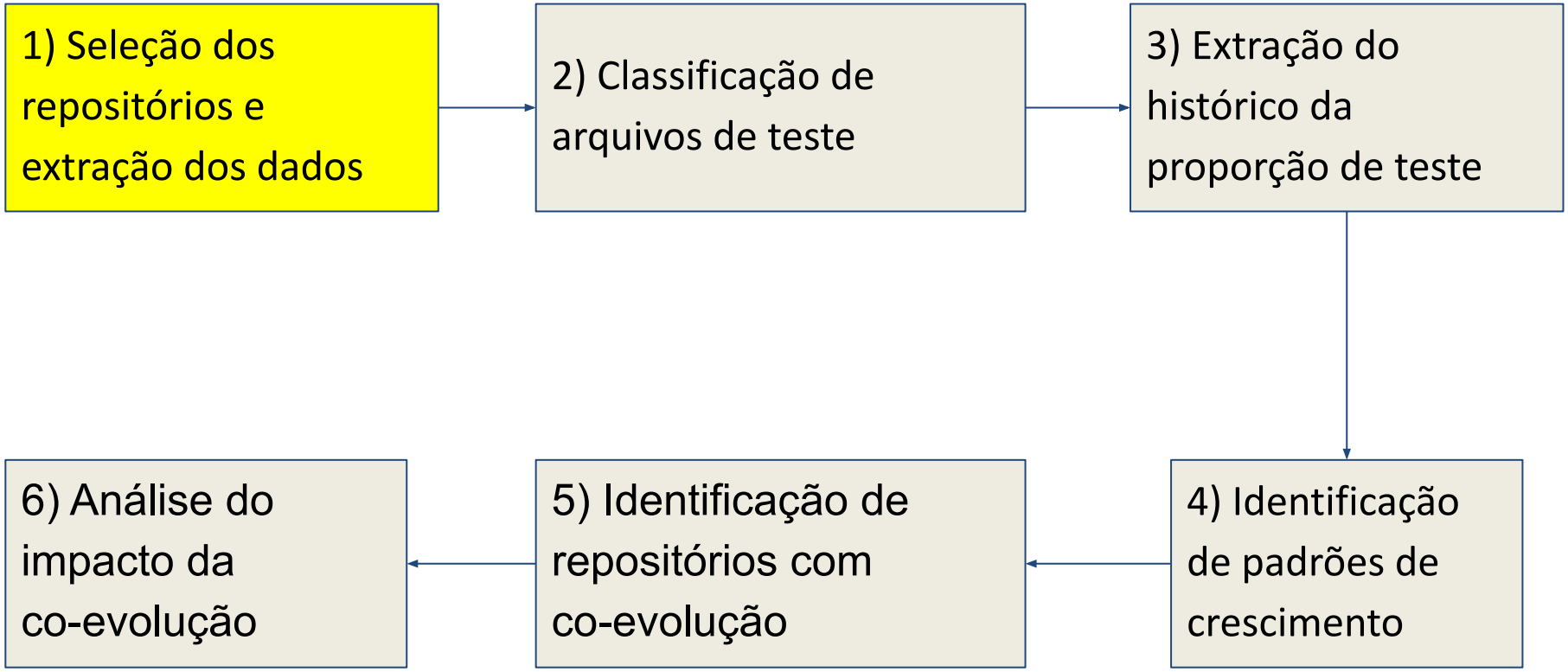
Zaidman et al (2011)

- Apresenta **três visões para o estudo da co-evolução**: i) Histórico de Mudanças, ii) Histórico de crescimento, iii) Evolução da Cobertura de Teste.

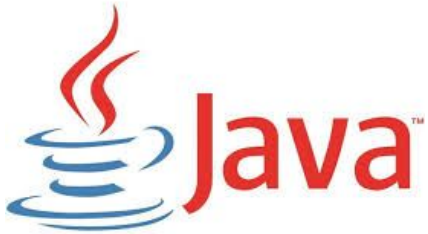
- Os resultados obtidos mostram que embora o código de teste e produção possam ser commitados em momentos diferentes (faseado) ou juntos (síncrono), o **padrão de desenvolvimento síncrono** foi encontrado com maior frequência.

Metodologia

Abordagem


1) Seleção dos repositórios e extração dos dados
2) Classificação de arquivos de teste
3) Extração do histórico da proporção de teste
4) Identificação de padrões de crescimento
5) Identificação de repositórios com co-evolução
6) Análise do impacto da co-evolução

Seleção e extração



600 de cada linguagem
3.000 repositórios públicos

Seleção e extração

Dados dos repositórios (K = milhares, M = milhões)

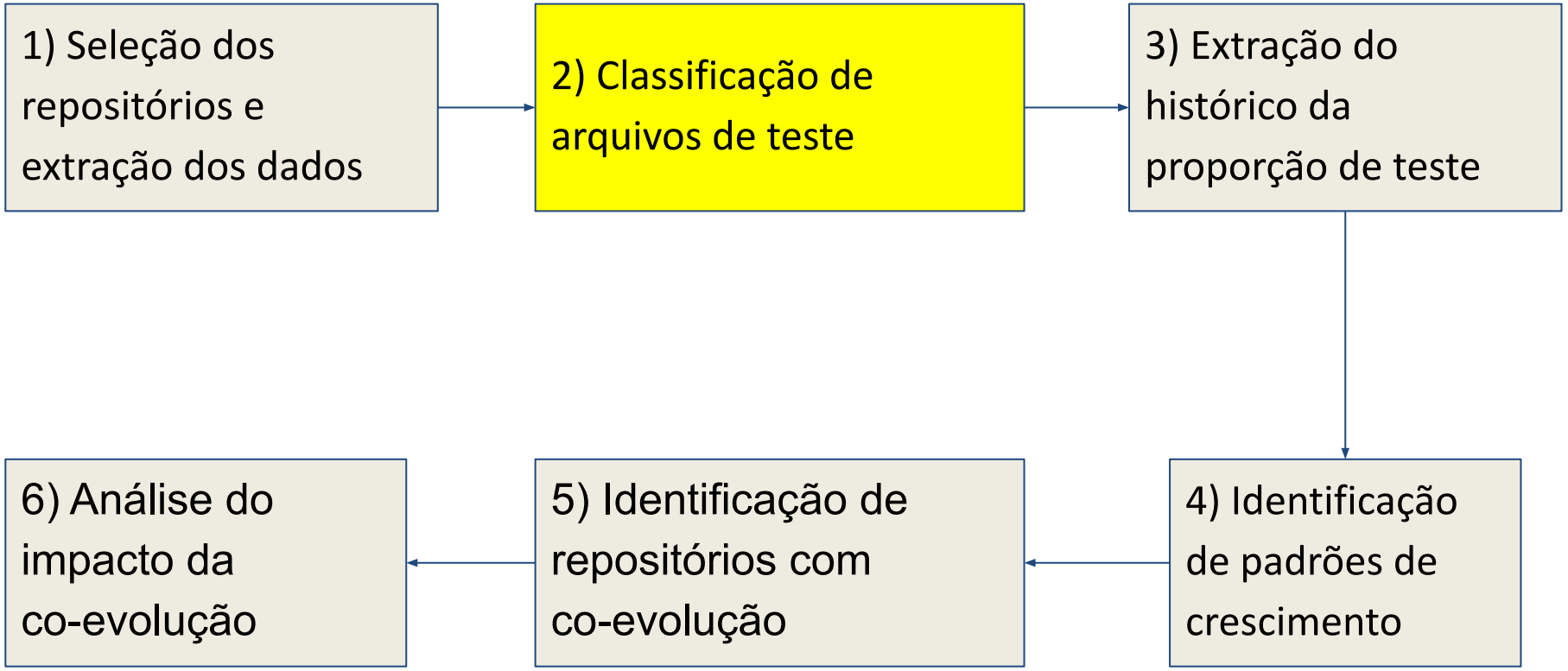
Linguagem	Commits	Devs	Forks	Issues	Size	Stars
Javascript	1.74M	52.62K	1.67M	141.69K	28.35GB	10.79M
Java	2.00M	37.95K	1.35M	101.22K	51.89GB	4.77M
Python	1.96M	61.67K	1.20M	135.67K	27.78GB	5.51M
PHP	2.11M	47.37K	461.74K	63.76K	17.81GB	2.46M
Ruby	2.11M	64.34K	392.38K	42.56K	14.22GB	2.12M
Todos	**9.95M**	**263.97K**	**5.08M**	**484.92K**	**140.08GB**	**25.68M**

Seleção e extração

Amostra de dados extraídos

Hash	68c3d2***c12fe5	4a19e4c***99bdb
Autor	Mon***9	Ky***ch
E-mail	Mon***9@users.noreply.github.com	kr***rk@gmail.com
Data	Thu Feb 23 23:00:50 2017 -0800	Mon Oct 14 00:45:03 2019 -0400
Situacao	ADDED	MODIFIED
Caminho	src/avatar/Avatar.js	src/avatar/Avatar.js

Abordagem

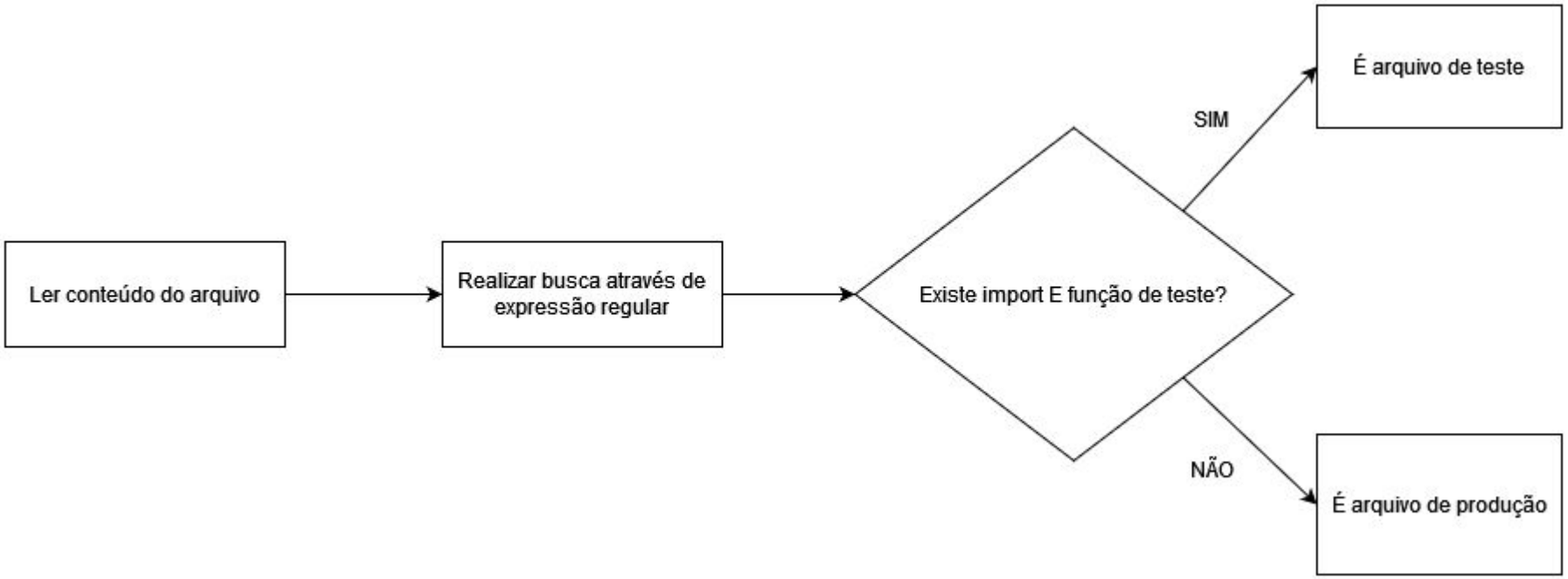

1) Seleção dos repositórios e extração dos dados
2) Classificação de arquivos de teste
3) Extração do histórico da proporção de teste
4) Identificação de padrões de crescimento
5) Identificação de repositórios com co-evolução
6) Análise do impacto da co-evolução

Classificação de teste

Expressões regulares do Catálogo de Framework (Gonzales et al., 2017)

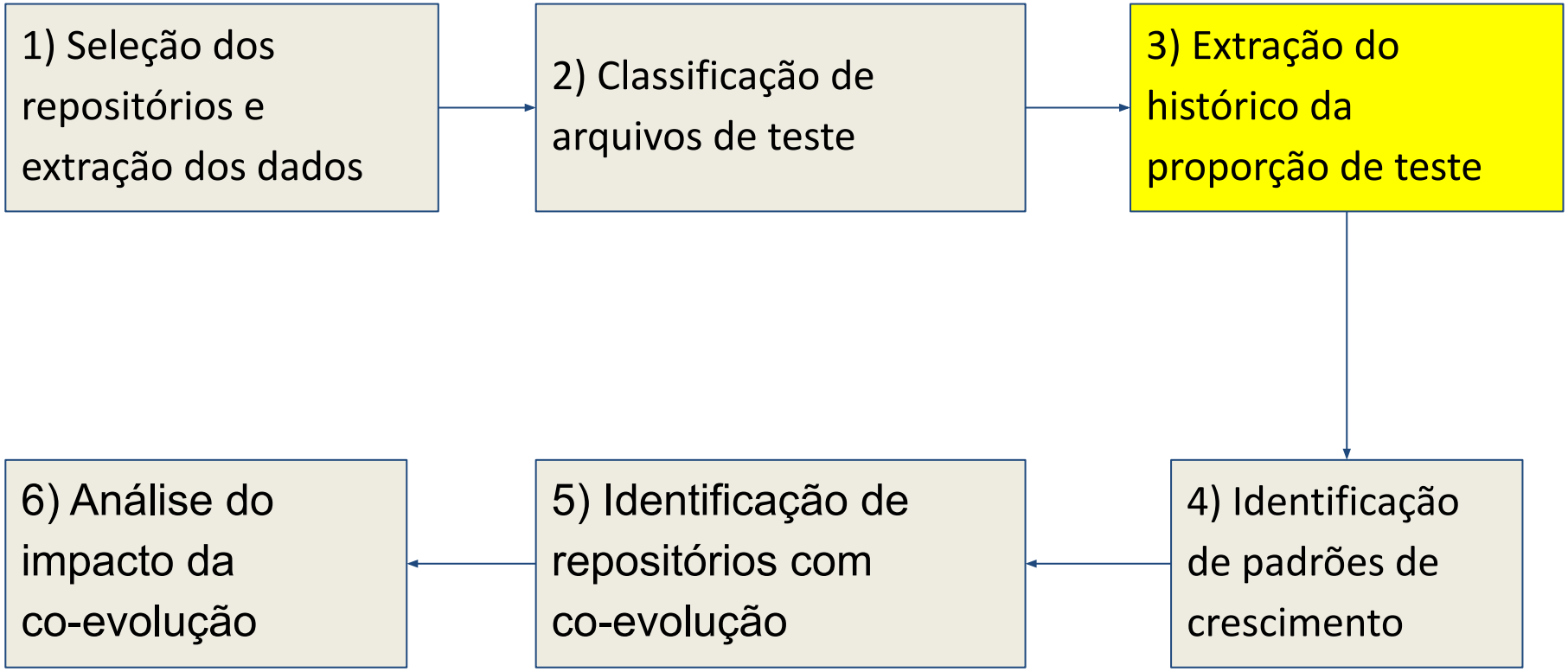
Tipo	Expressão regular
Import Javascript	JS.Test [\.[\w]*]*, jsUnity [\.[\w]*]*, ...
Função Javascript	test\(, describe\(
Import Java	import +[\w\.]*JUnit, import +[\w\.]*jmock, ...
Função Java	@Test, @Before, @After, @RunWidth, assert.*, ...
Import Python	((from)|(import)) +unittest, ...
Função Python	def test_, testmod\(
Import PHP	extends PHPUnit_Framework_TestCase, ...
Função PHP	test, assert, @test, CLEAN, EXPECT ...
Import Ruby	(“|’)[\w]*test[\w]*(“|’), (“|’)[\w]*spec[\w]*(“|’) ...
Função Ruby	assert\(, assert_, describe

Classificação de teste



Fluxo da classificação de teste (Gonzales et al., 2017)

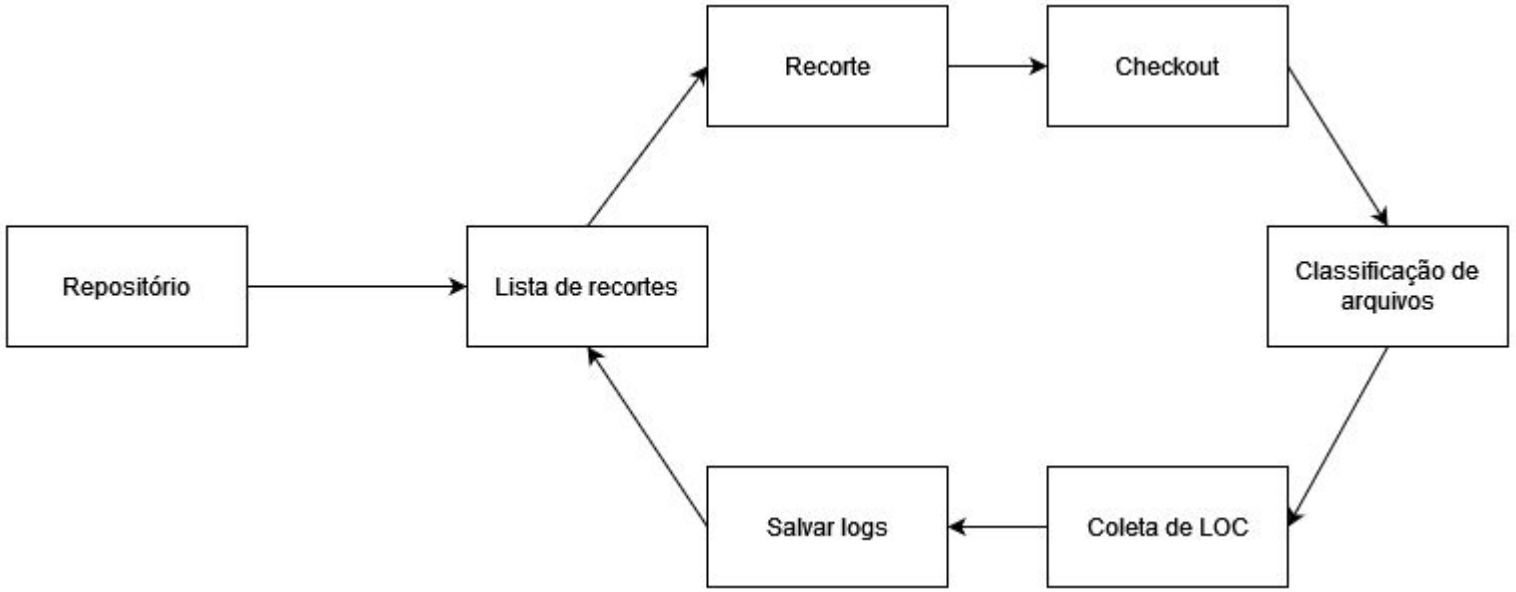
Abordagem


1) Seleção dos repositórios e extração dos dados
2) Classificação de arquivos de teste
3) Extração do histórico da proporção de teste
4) Identificação de padrões de crescimento
5) Identificação de repositórios com co-evolução
6) Análise do impacto da co-evolução

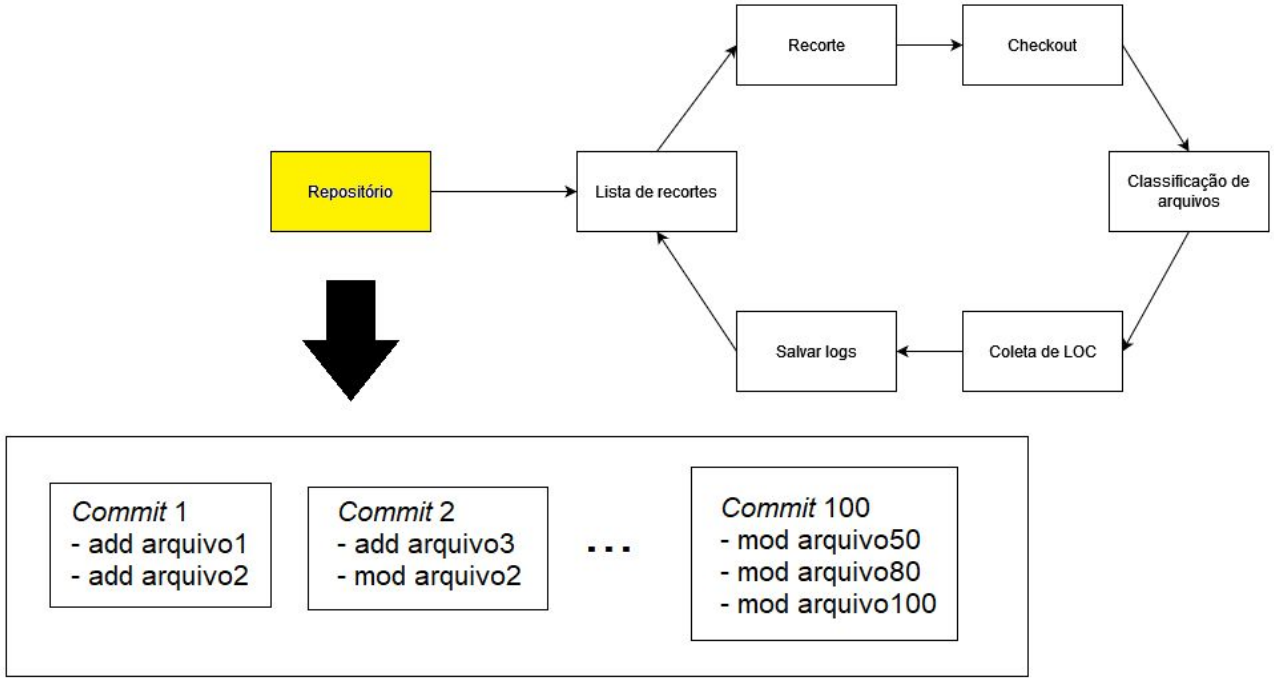
Proporção de Teste

$$\text{proporcaoTeste} = \frac{totalLinhasCodigoTeste}{totalLinhasCodigoTeste + totalLinhasCodigoProducao}$$

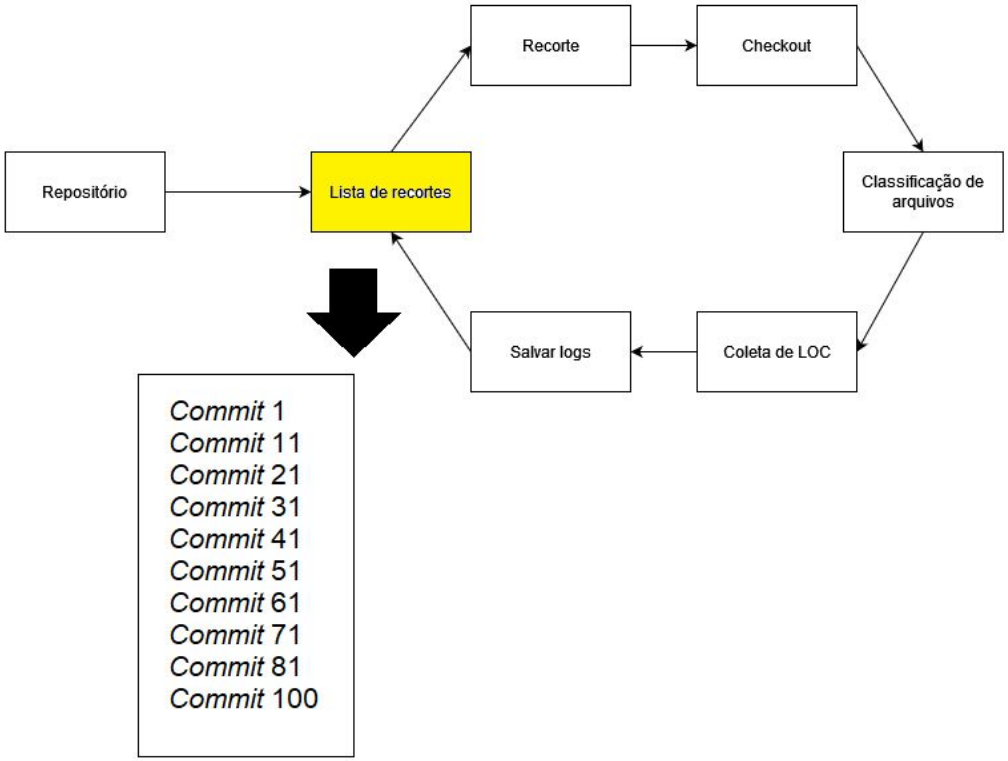
Proporção de teste



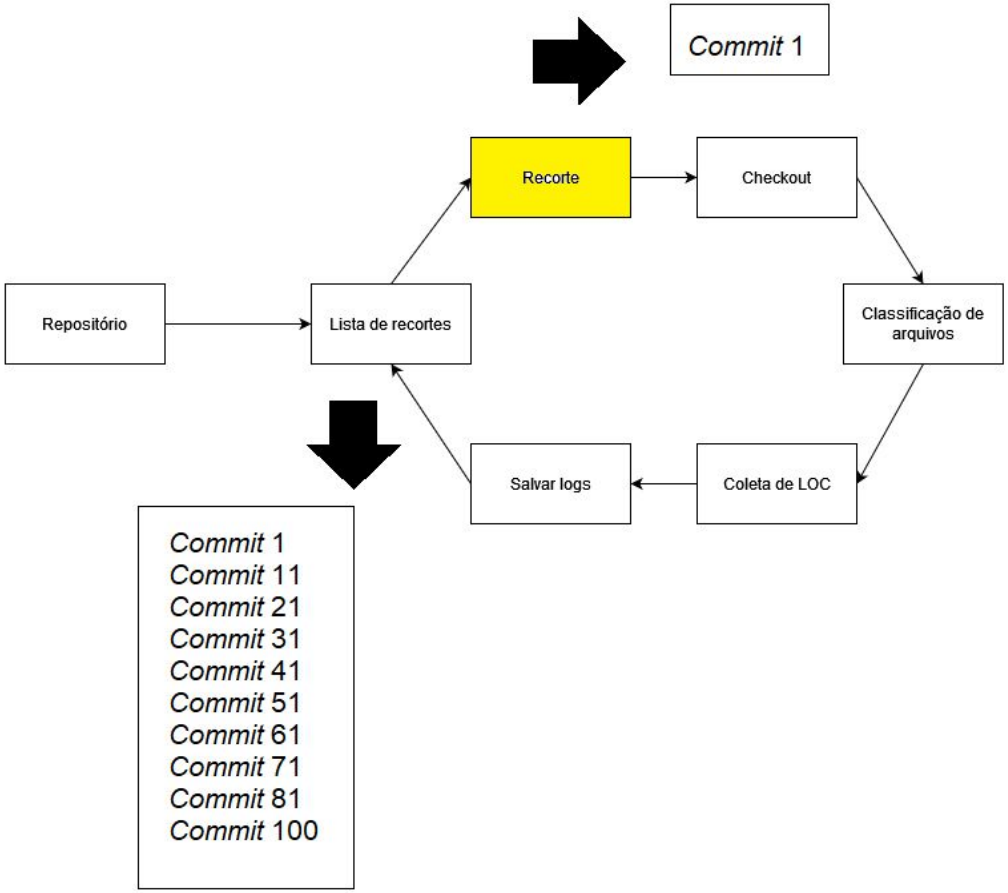
Proporção de teste Repositório



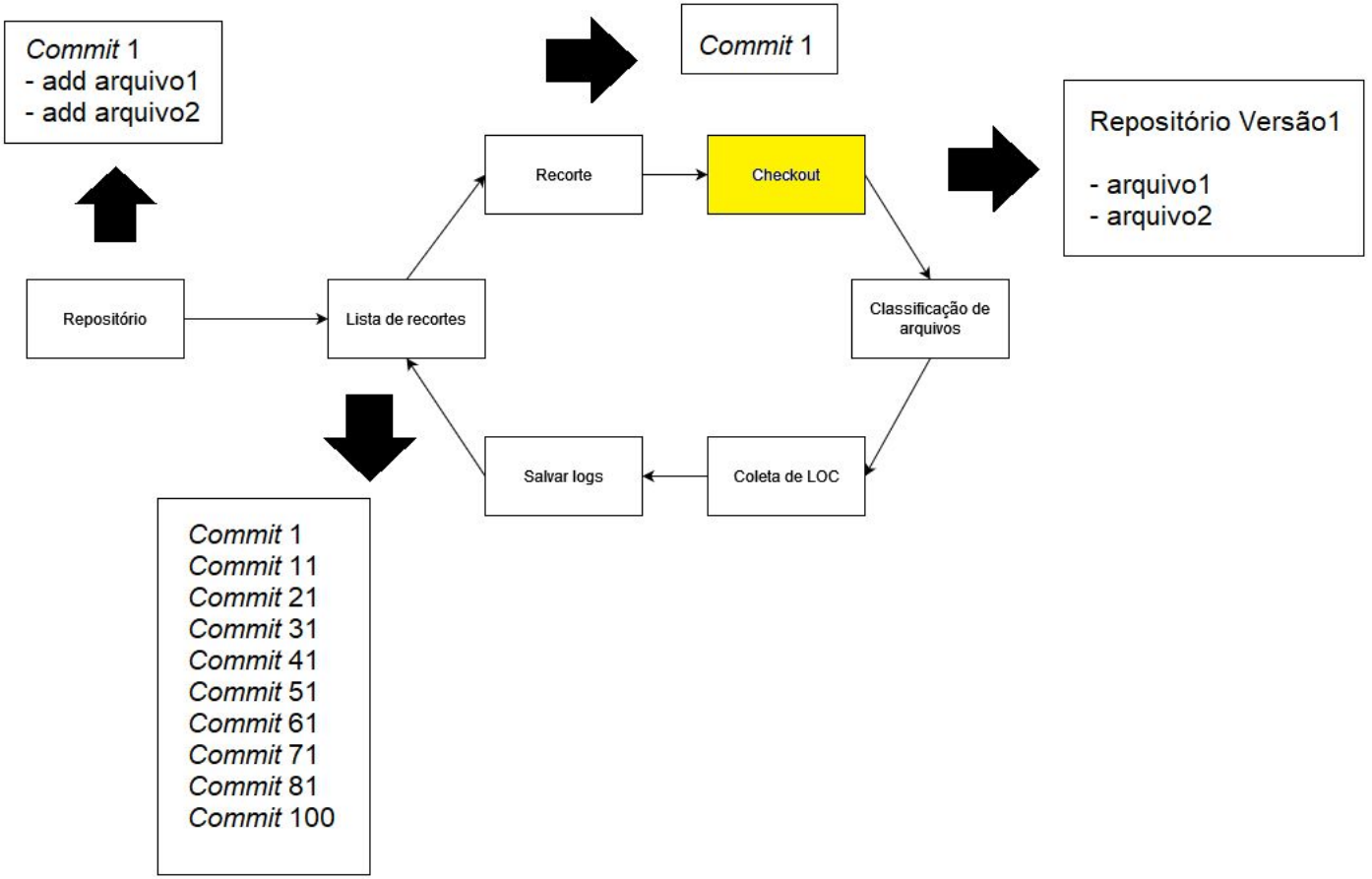
Proporção de teste
Lista de recortes



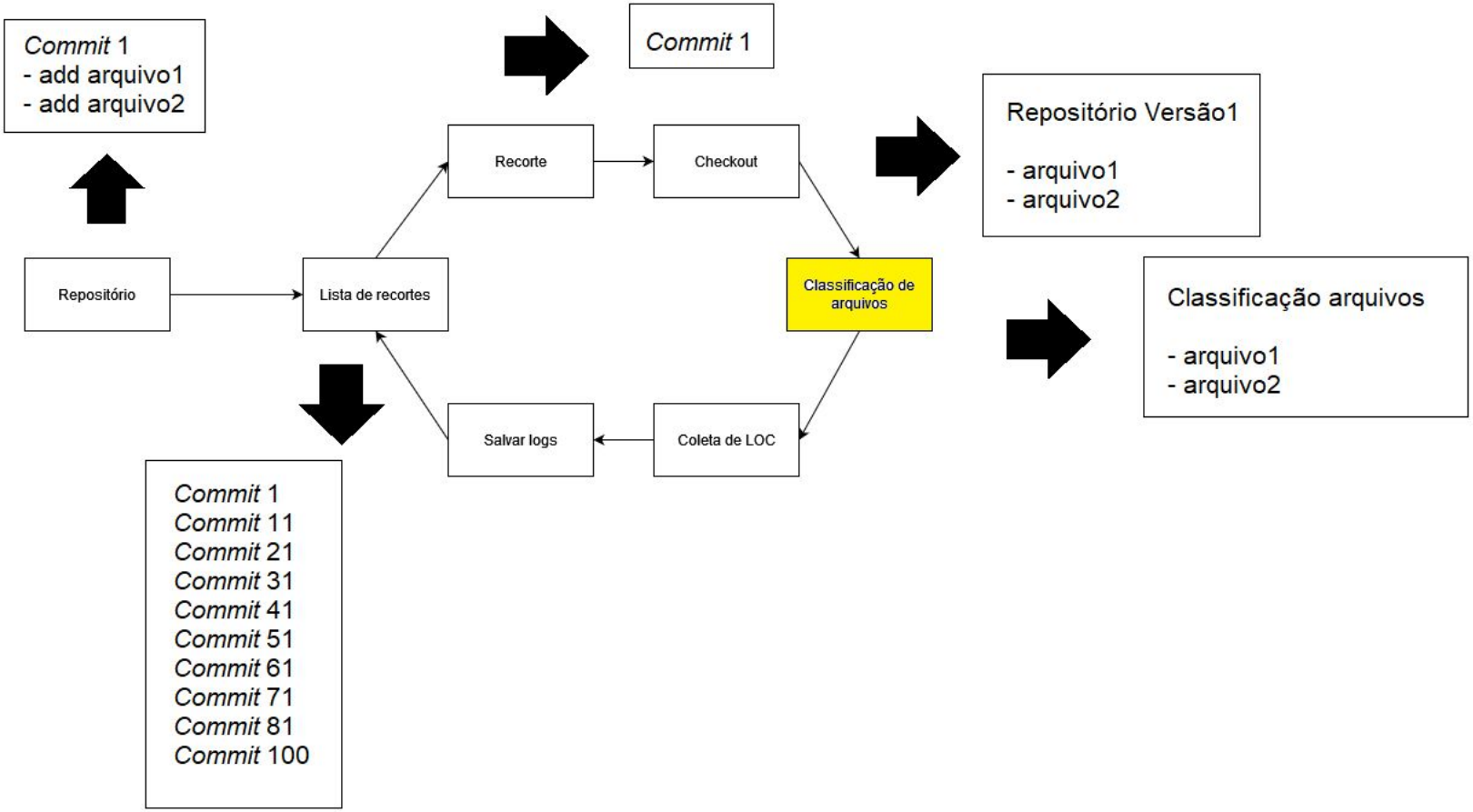
Proporção de teste Recorte



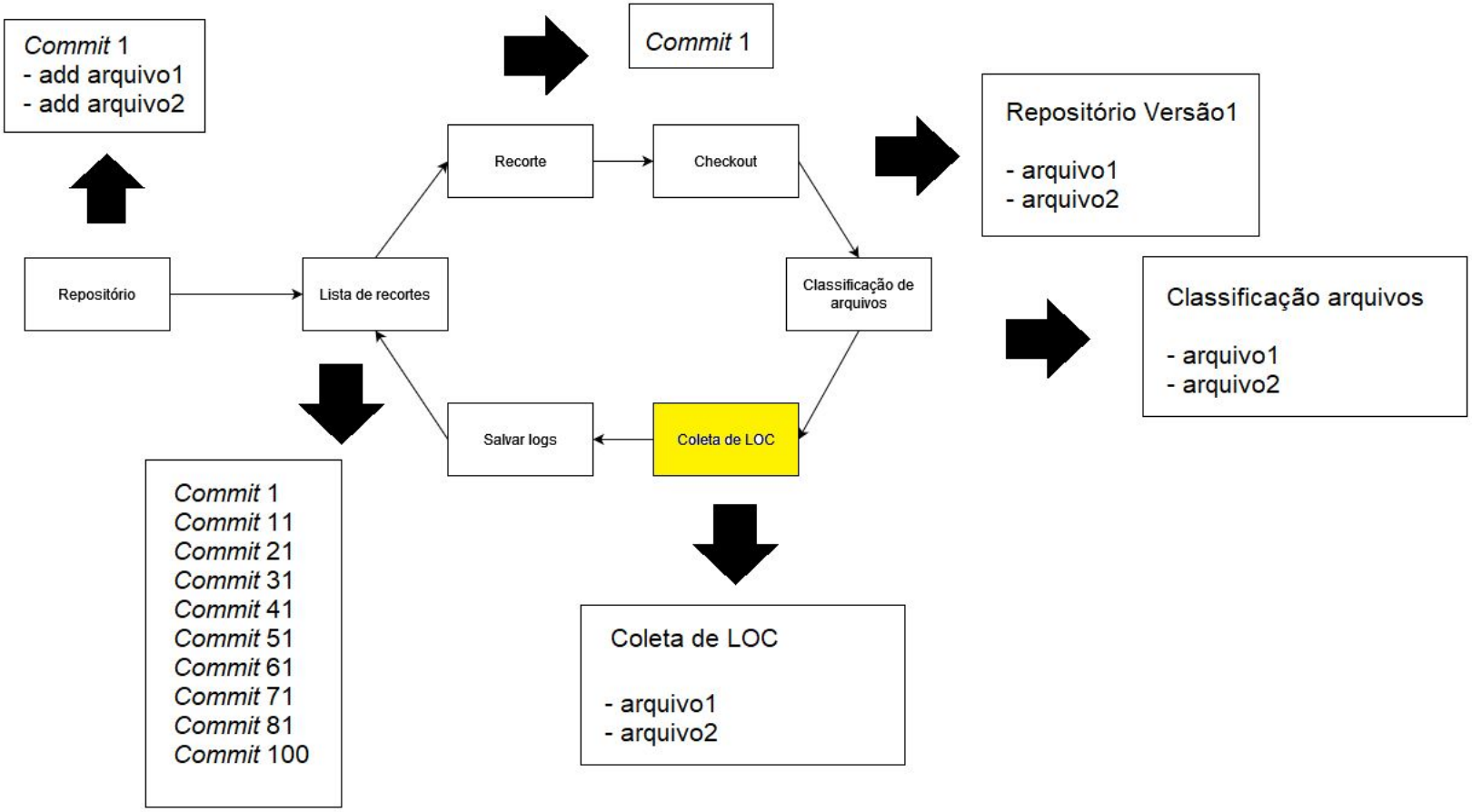
Proporção de teste Checkout



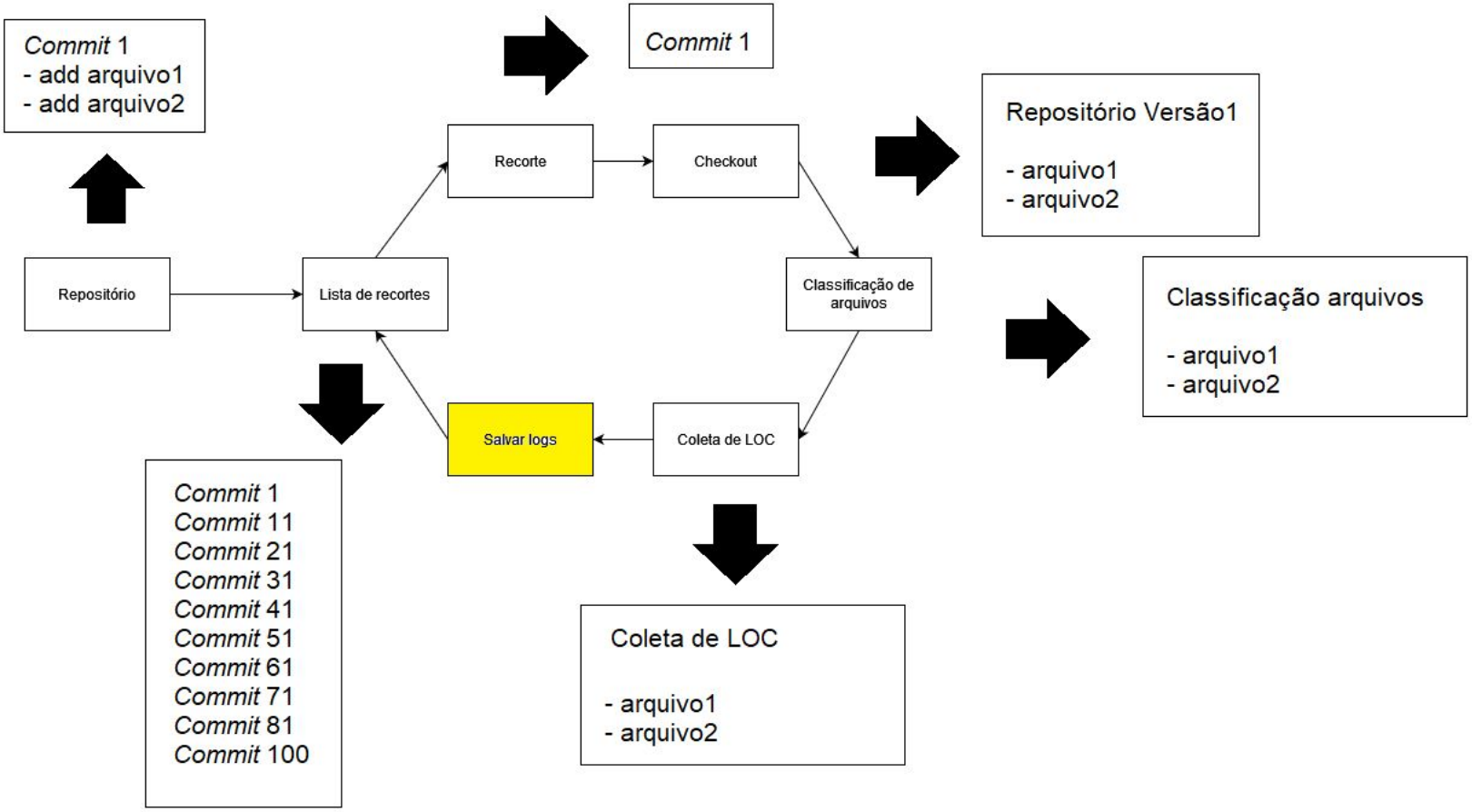
Proporção de teste
Classificação de teste



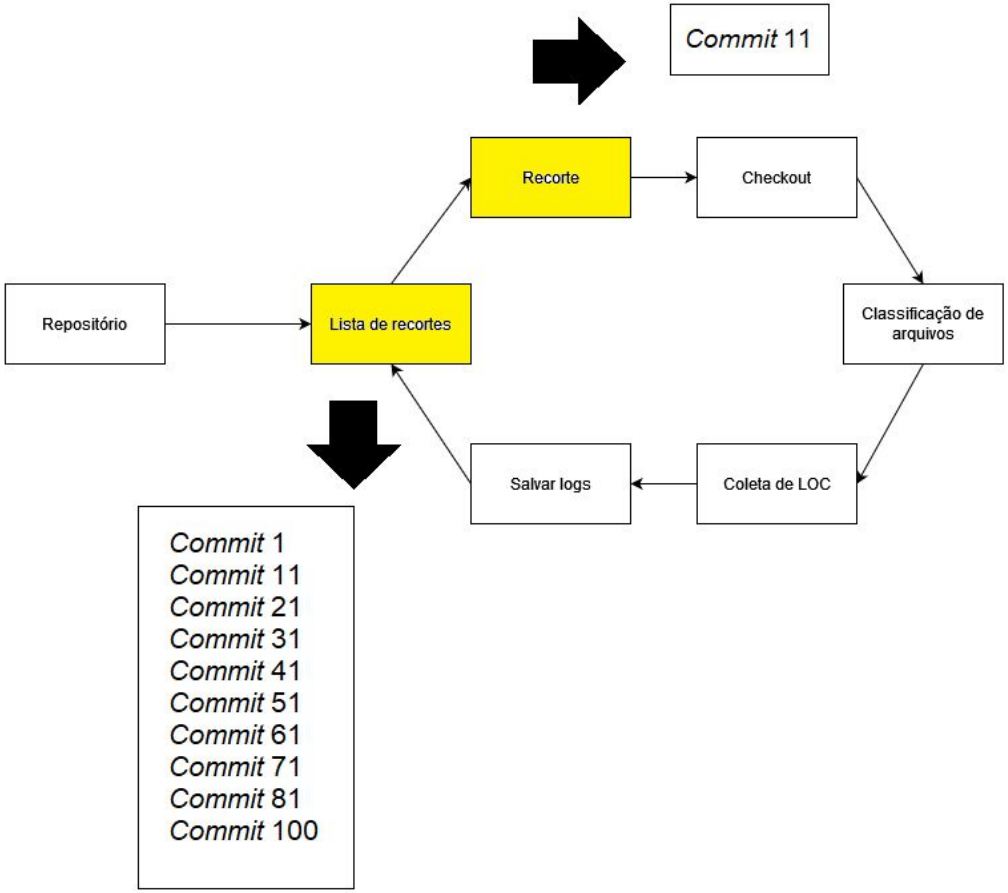
Proporção de teste
Coleta de LOC



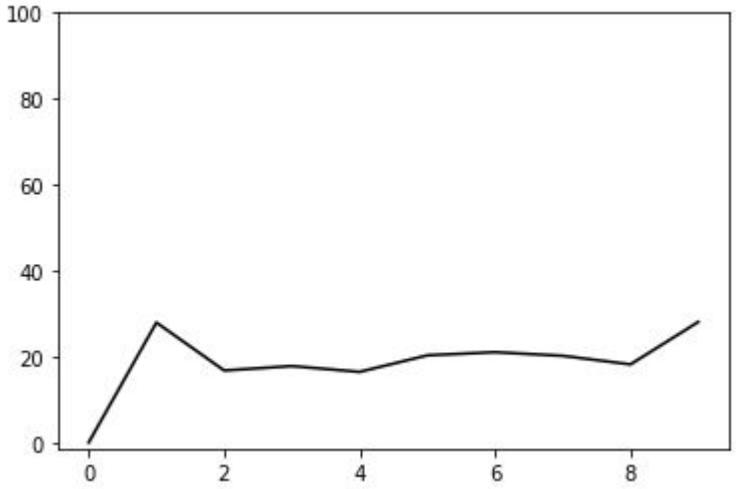
Proporção de teste
Salvar logs



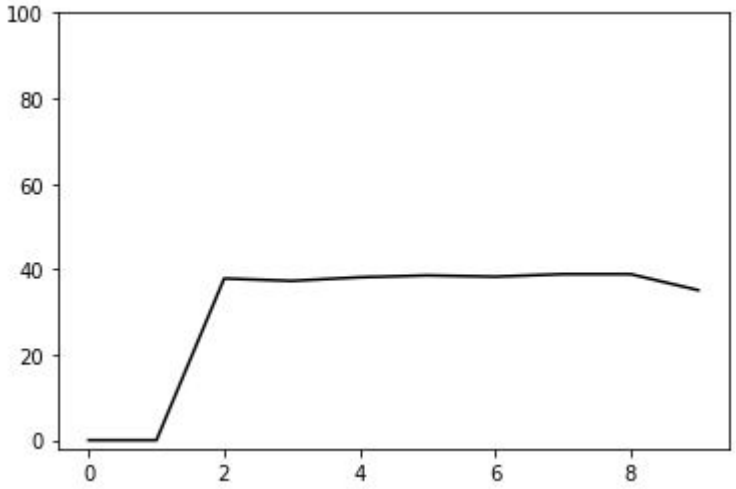
Proporção de teste
Nova iteração



Proporção de teste Resultado

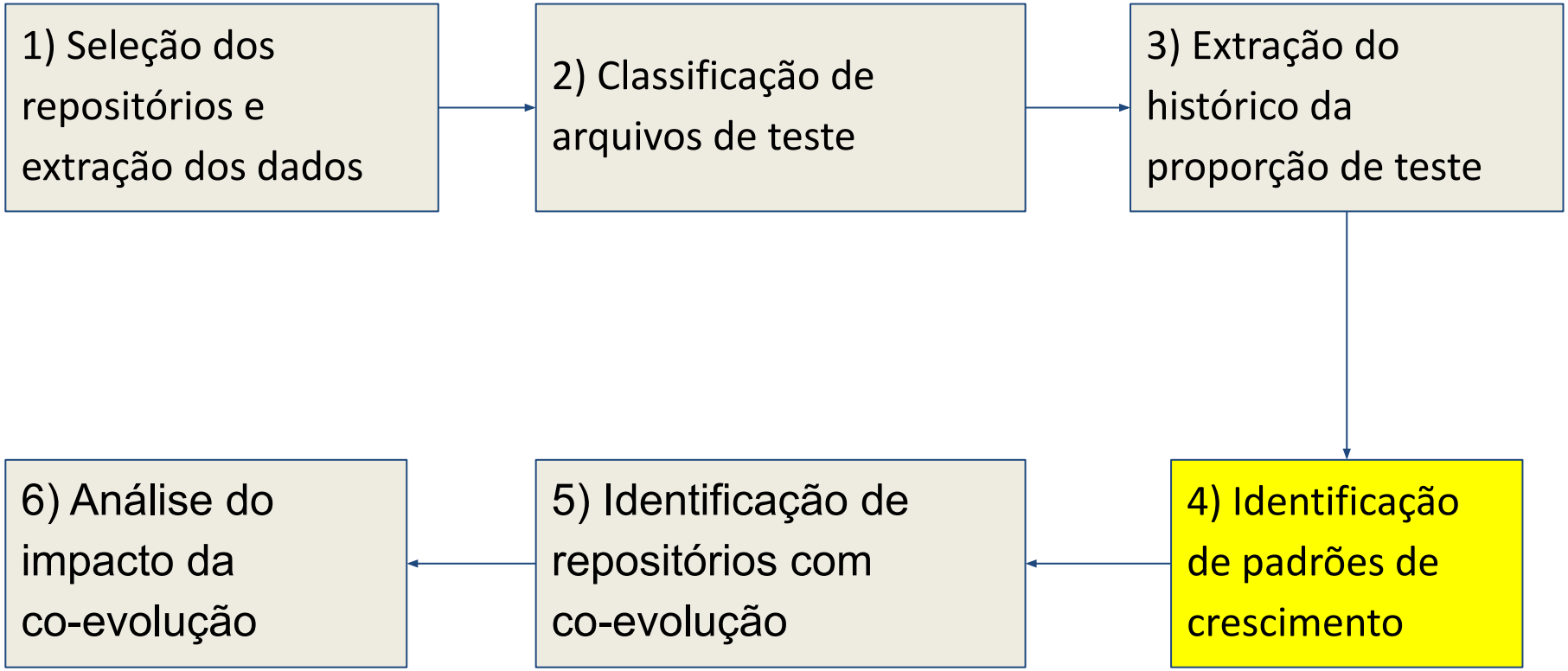


a) Proporção de teste do hashie/hashie

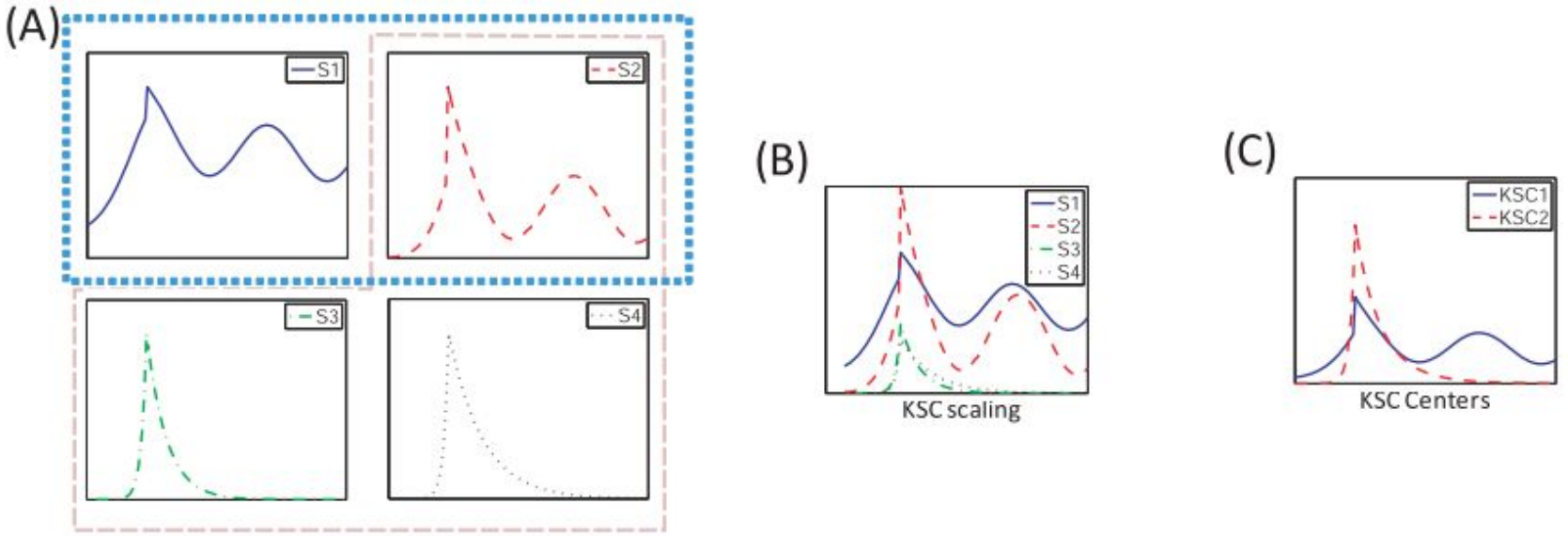


b) Proporção de teste do ircmaxell/password

Abordagem

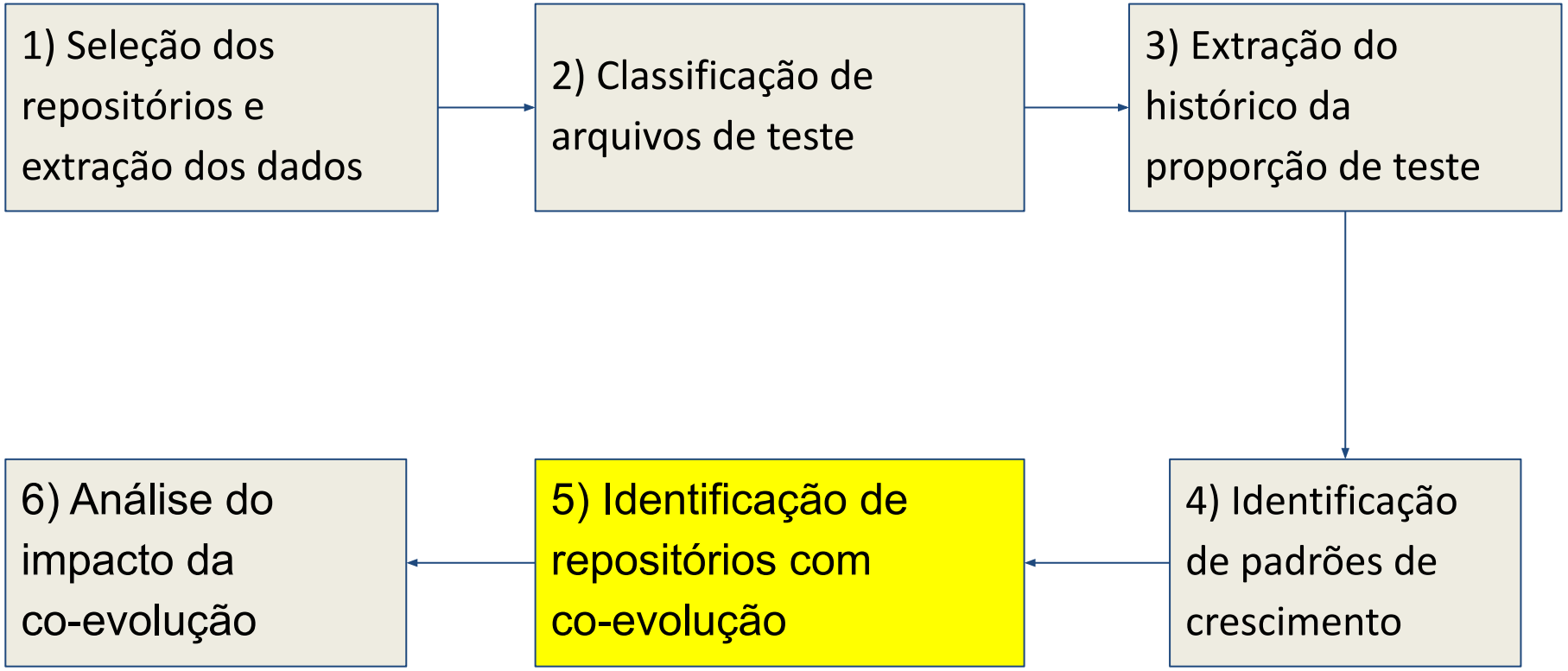

1) Seleção dos repositórios e extração dos dados
2) Classificação de arquivos de teste
3) Extração do histórico da proporção de teste
4) Identificação de padrões de crescimento
5) Identificação de repositórios com co-evolução
6) Análise do impacto da co-evolução

Identificando Padrões

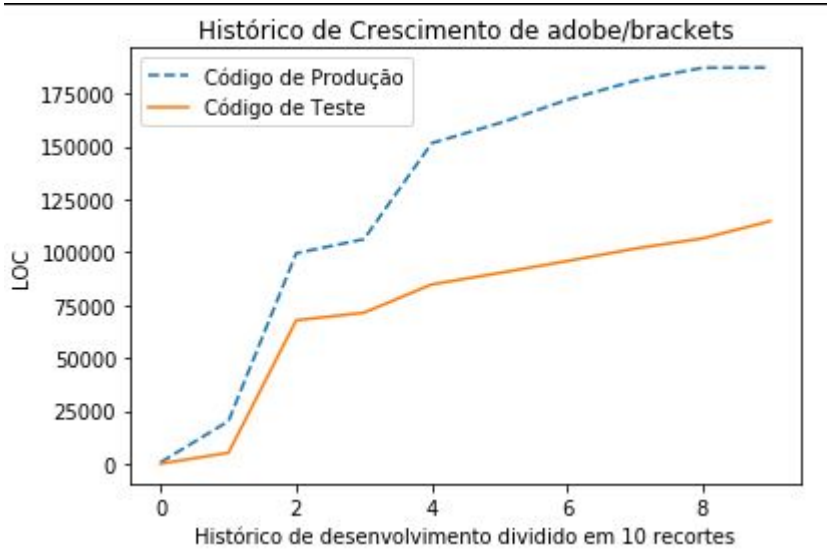


Funcionamento do Algoritmo KSC

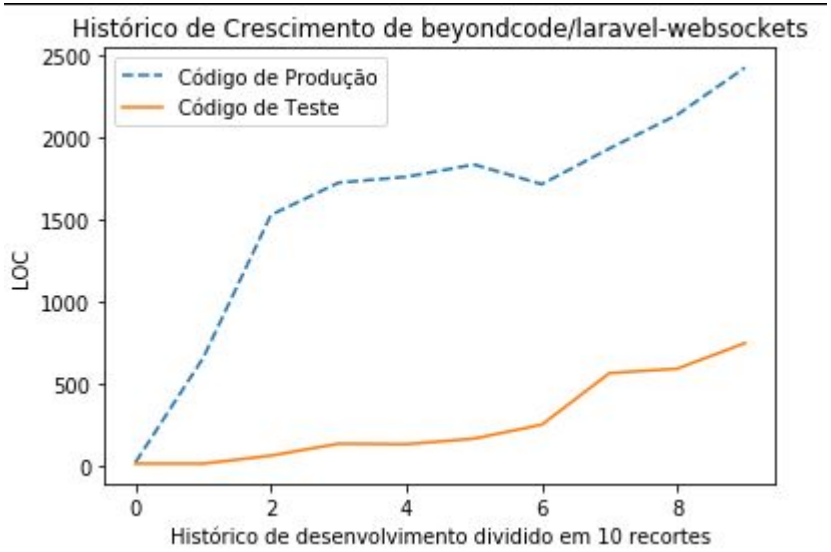
Abordagem


1) Seleção dos repositórios e extração dos dados
2) Classificação de arquivos de teste
3) Extração do histórico da proporção de teste
4) Identificação de padrões de crescimento
5) Identificação de repositórios com co-evolução
6) Análise do impacto da co-evolução

Identificando co-evolução

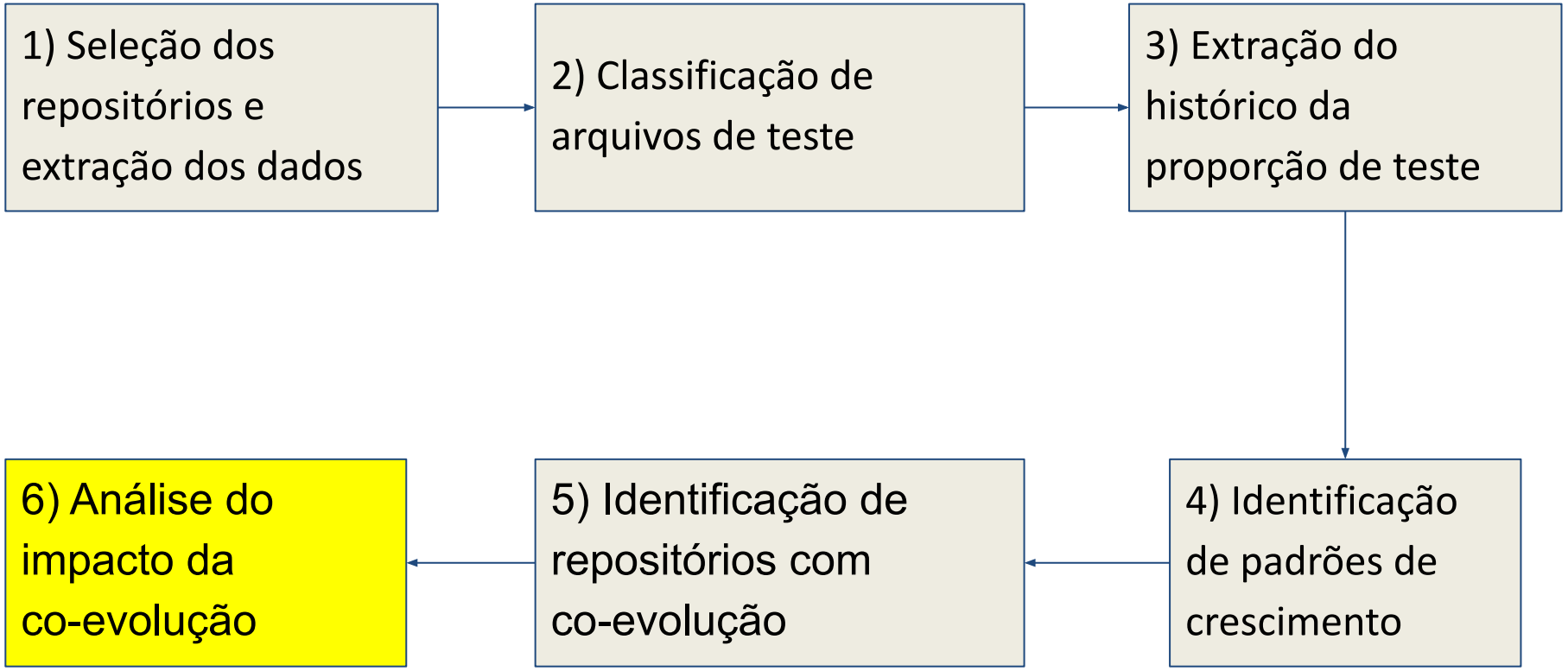


a) 4ºquartil (Forte co-evolução)
p de person é 0.98

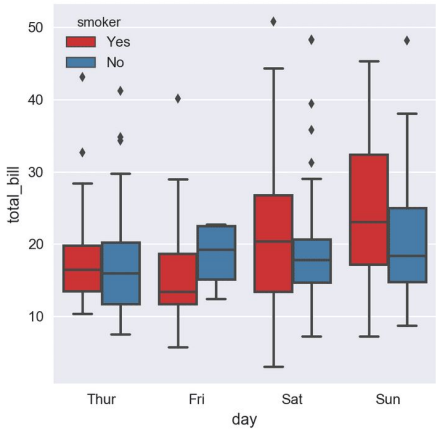


b) 1ºquartil (Fraca co-evolução)
p de pearson é 0.73

Abordagem


1) Seleção dos repositórios e extração dos dados
2) Classificação de arquivos de teste
3) Extração do histórico da proporção de teste
4) Identificação de padrões de crescimento
5) Identificação de repositórios com co-evolução
6) Análise do impacto da co-evolução

Impactos da co-evolução



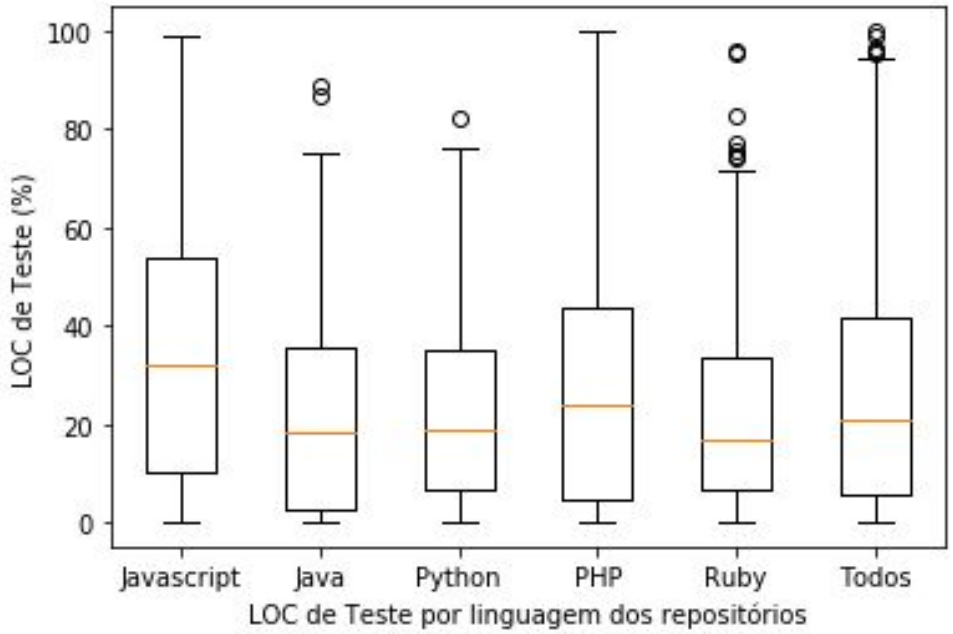
Comparando os repositórios com forte e fraca co-evolução através de quatro características, são elas:

- Número de Commits;
- Número de Colaboradores;
- Número de Forks;
- Número de Issues.

Resultados

QP1) Como testes evoluem em projetos de software?

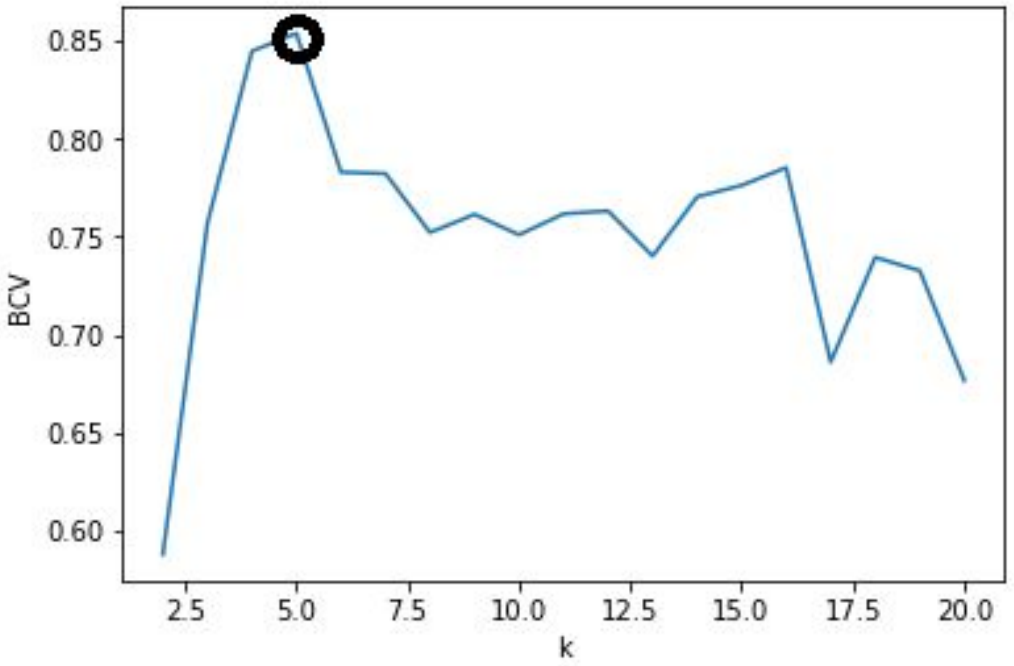
1.914 (63,8%) possuem teste.



QP1) Como testes evoluem em projetos de software?

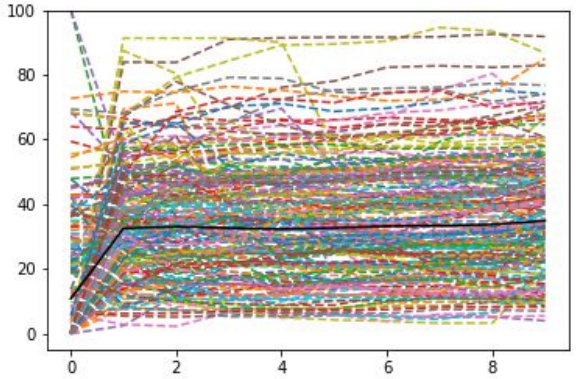
- Identificados os projetos que possuem teste
- Realizamos filtros para remover os repositórios menos significantes e que poderiam representar ruídos na clusterização

QP1) Como testes evoluem em projetos de software?

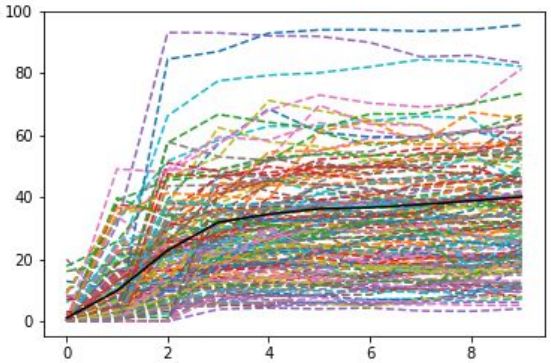


Melhor *k* igual a 5

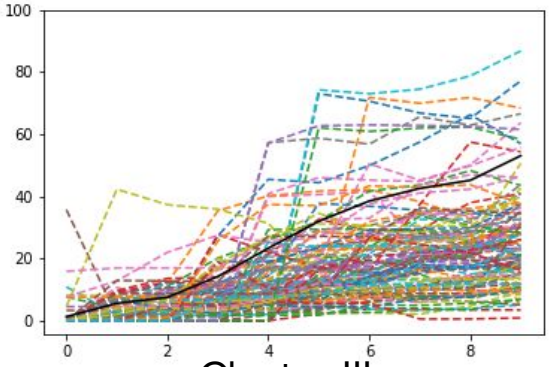
QP1) Como testes evoluem em projetos de software?



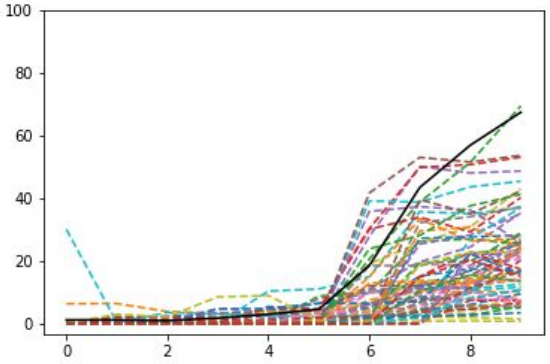
Cluster I



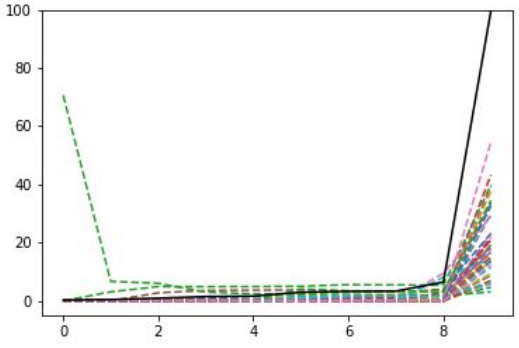
Cluster II



Cluster III



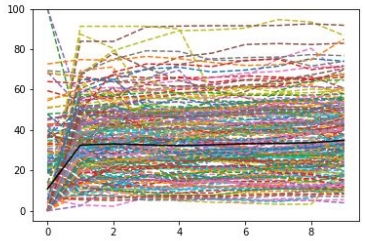
Cluster IV



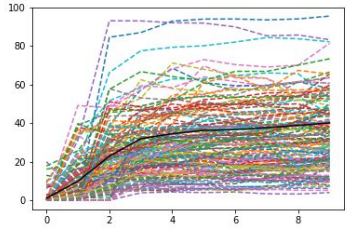
Cluster V

QP1) Como testes evoluem em projetos de software?

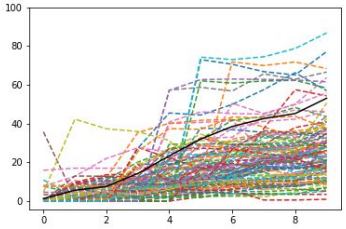
Cluster	Característica	Total(%)	Javascript(%)	Java(%)	Python(%)	PHP(%)	Ruby(%)
I	Testes estáveis desde o início	35	**38**	**52**	24	**30**	**29**
II	Testes estáveis	26	23	30	**31**	25	20
III	Crescimento contínuo dos testes	20	23	11	26	18	25
IV	Testes tardios e em crescimento	12	10	5	15	12	18
V	Testes tardios e em crescimento intensivo	7	6	3	4	15	8



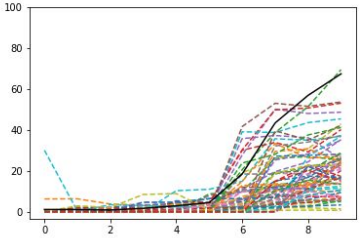
Cluster I



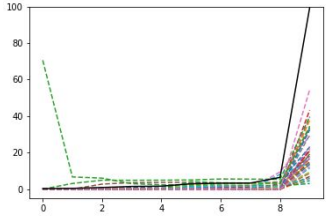
Cluster II



Cluster III



Cluster IV

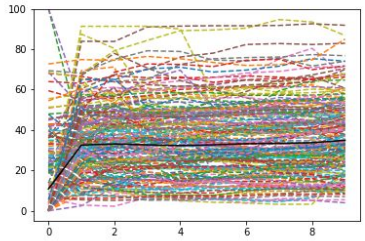


Cluster V

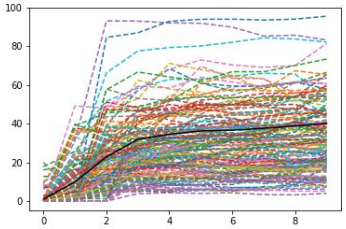
QP1) Como testes evoluem em projetos de software?

Resumo da QP1:

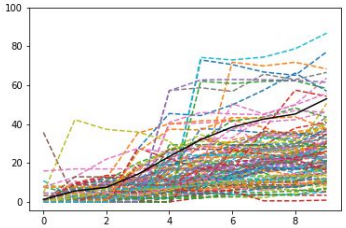
- Foram identificados **5 padrões de evolução de teste**.
- Os dois padrões **mais comuns (*clusters I e II*)**.
- Os demais **(*clusters IV e V*)** representam uma adoção **tardia de testes**.



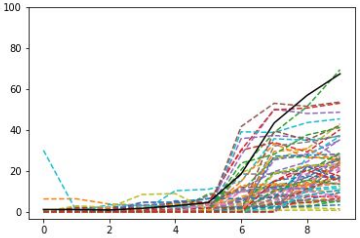
Cluster I



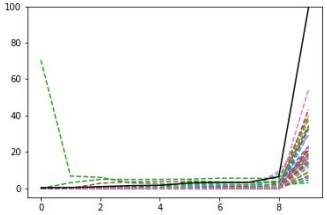
Cluster II



Cluster III

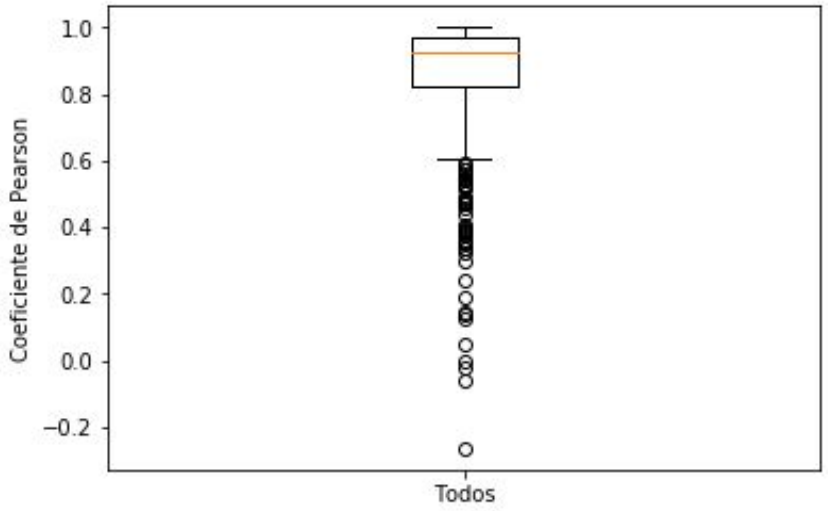


Cluster IV



Cluster V

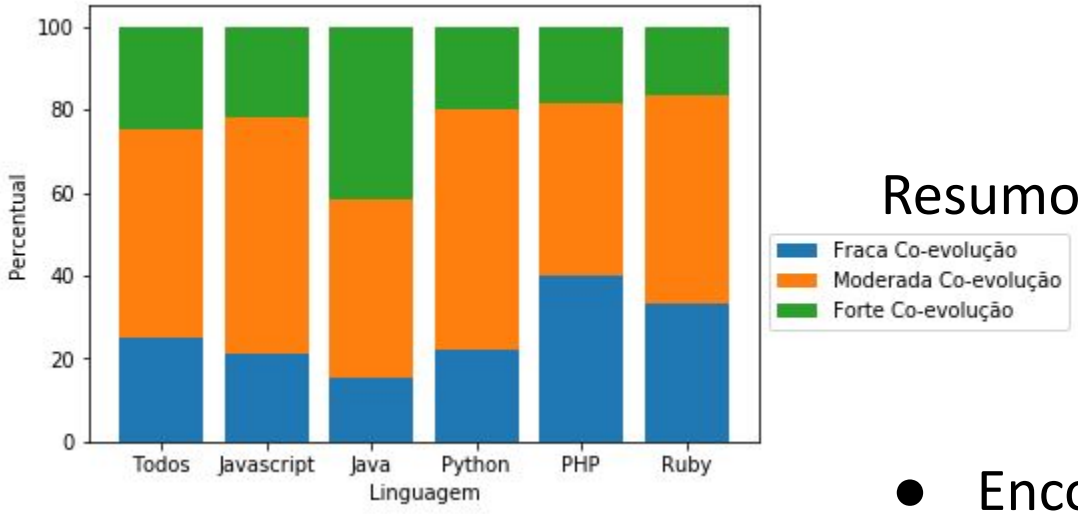
QP2) Com que frequência o código-fonte e testes co-evoluem em projetos de software?



- p > 0.96 = **forte co-evolução**;
- 0.82 < p <= 0.96 = **moderada co-evolução**;
- p <= 0.82 = **fraca co-evolução**;

Foram identificados **289 repositórios com fortes indícios de co-evolução**, 602 com moderada e 312 com fraca.

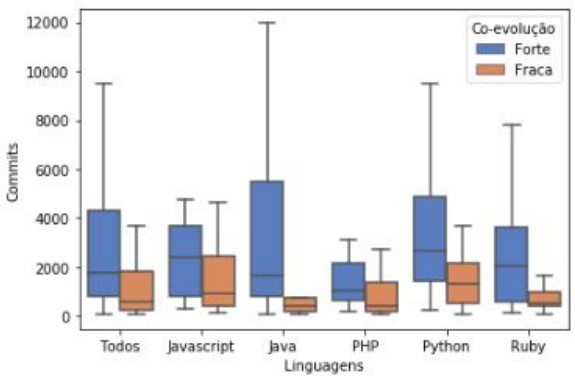
QP2) Com que frequência o código-fonte e testes co-evoluem em projetos de software?



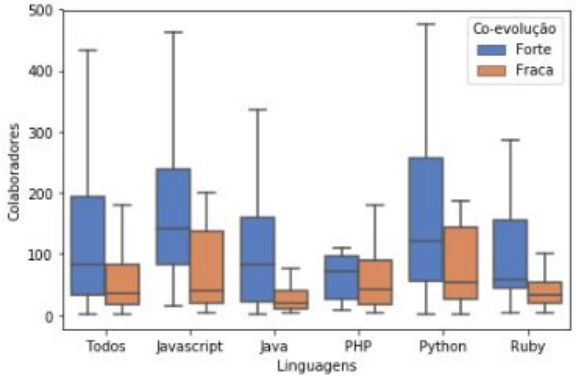
Resumo da QP2:

- Encontramos **repositórios com fortes indícios de co-evolução** em todas as linguagens, com destaque para a linguagem **Java**;

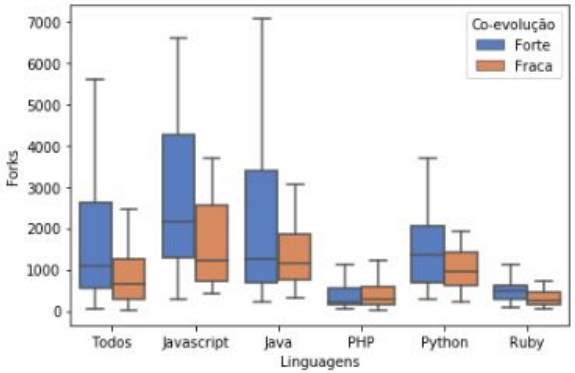
QP3) Que características distinguem projetos onde há co-evolução de código de teste de projetos onde essa prática não é comum?



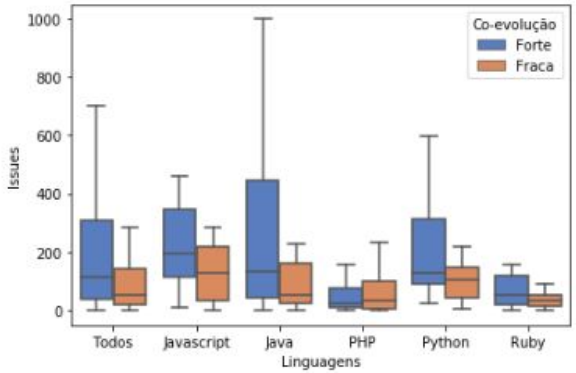
(a) *Commits*



(b) Colaboradores

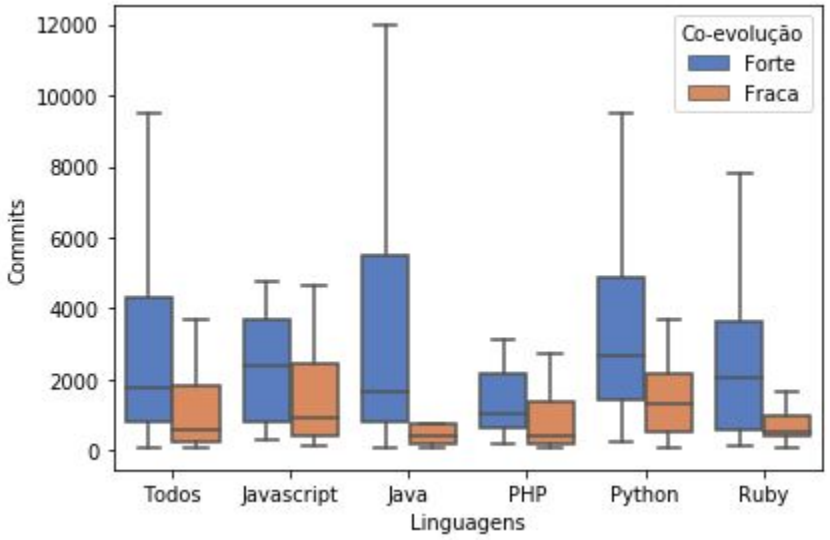


(c) *Forks*



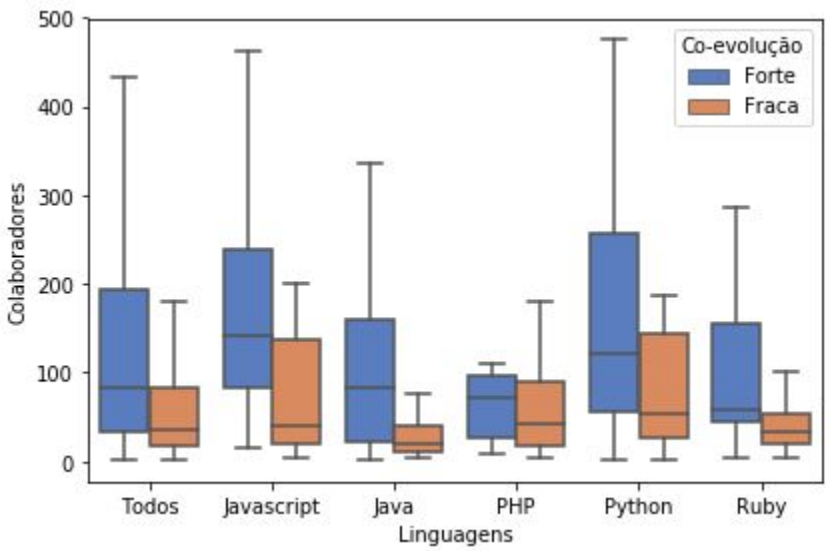
(d) *Issues*

Commits



Pode-se afirmar que para **todas as linguagens**, os repositórios com **forte co-evolução** possuem **mais *commits***.

Colaboradores



Diferença média

- Javascript, Java e Ruby

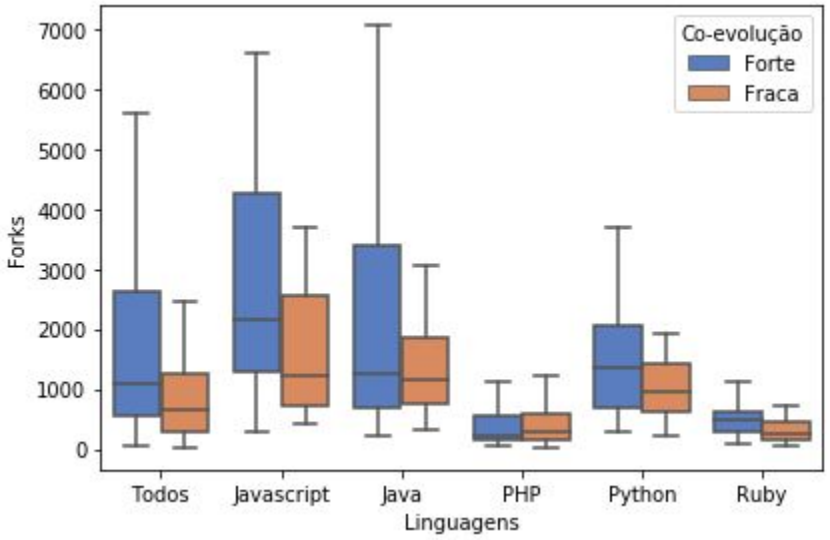
Diferença pequena

- Python

Diferença não é estatisticamente diferente

- PHP

Forks



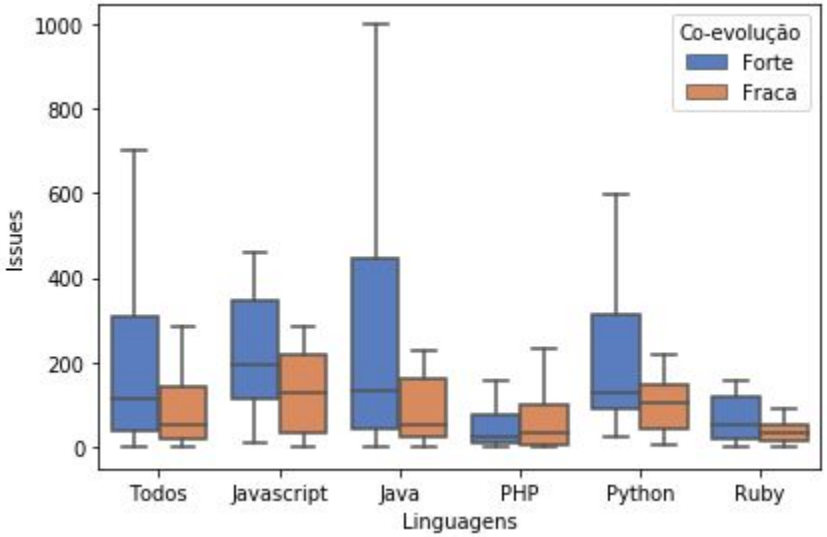
Diferença média

- Javascript

Diferença não é estatisticamente diferente

- Java, Python, PHP e Ruby

Issues



Diferença média

- Java

Diferença pequena

- Javascript e Python

Diferença não é estatisticamente diferente

- PHP e Ruby

Todos

Resumo da QP3:

De modo geral, os repositórios com forte co-evolução de teste **possuem mais commits, mais colaboradores, mais forks e mais issues**.

Conclusão e trabalhos futuros

4. Conclusões

As principais contribuições deste trabalho são:

- A construção e disponibilização pública de **um grande dataset**;
- A **identificação de padrões comportamentais comuns** no desenvolvimento de testes;

4. Conclusões

As principais contribuições deste trabalho são:

- Definição de **uma metodologia para identificação de co-evolução de testes** em projetos de desenvolvimento de software;

- Provendo **indícios da influência da co-evolução** em aspectos diversos do projeto.

4. Conclusões

Trabalhos futuros:

- **Comparar ou realizar outras abordagens** para identificação da co-evolução de teste;
- **Realizar um survey** com os desenvolvedores;
- **Verificar os defeitos pós-release** dos projetos;

Obrigado!

Aluno: Charles José Lima de Miranda;
Orientador: Guilherme Amaral Avelino
Co-orientador: Pedro de Alcântara dos S Neto