Test Case Prioritization Using Online Fault Detection Information

Mohsen Laali

Huai Liu

Margaret Hamilton

Maria Spichkova

Heinz Schmidt

Outline

- Introduction
  - Regression Testing
  - Test Case Prioritization (TCP)
- Research Questions
- Experimental Methodology & Results
- Conclusions

Introduction

- Regression Testing
- Different Types
  - Test Case Minimization
  - Test Case Selection
  - Test Case Prioritization

```
1: function PRIORITIZATION (ST, NST, P, Q)
2:     if | NST| == 1 then
3:         Q.Add(NST)
4:         return Q
5:     end if
6:     t = Selection(ST, NST, P)
7:     NST = NST / t
8:     ST = ST ∪ t
9:     Q.Add(t)
10:     PRIORITIZATION (ST, NST, P, Q)
11: end function
```

ST: Selected Test case, NST: Not Selected Test case, P: Program,

Q: Queue for ordered test cases, and t: test case.

TCP Techniques

- Coverage Granularity
  - Statement $SC(t, P)$
  - Branch $BC(t, P)$

- Existing Techniques
  - Total → TS
  - Additional → TB → AS, AB

$$t \in NST \text{ and } |SC(t,P)| \text{ is maximum} \Rightarrow \text{return } t$$

$$t \in NST \text{ and } |SC(t,P) - CS(ST,P)| \text{ is maximum} \Rightarrow \text{return } t$$

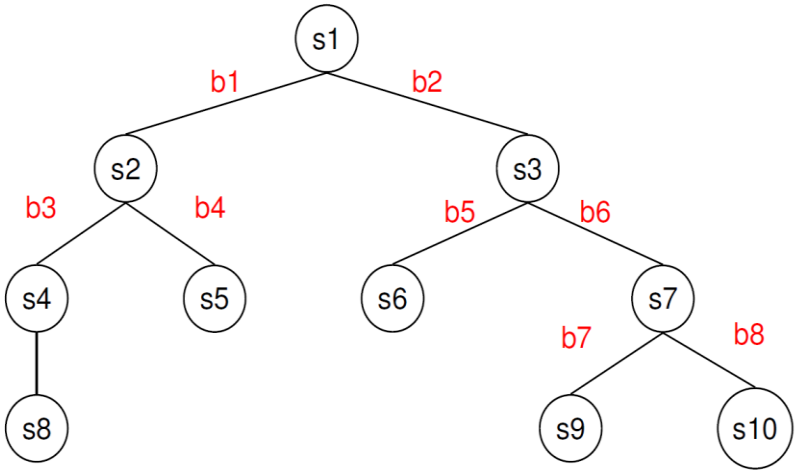
```
1: function CS(ST, P)
2:     TS = ∪_{t∈ST} ∪_{S∈SC(t,P)} S
3:     return TS
4: end function
```

ST: Selected Test case, P: Program, TS: Total set of covered Statements, t: test case,

SC: Statement Coverage function for t over program P returning covered statement S

TCP Techniques Illustration

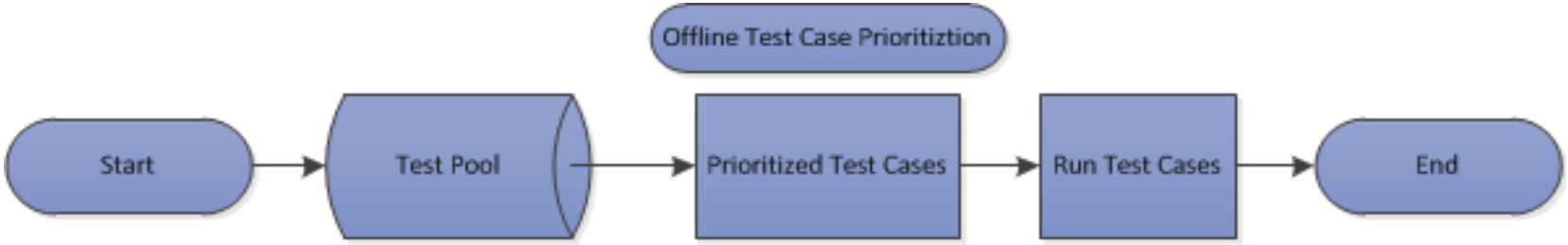
si	t1	t2	t3	t4	t5	bi	t1	t2	t3	t4	t5
s1	✓	✓	✓	✓	✓	b1	✓	✓			
s2	✓	✓				b2			✓	✓	✓
s3			✓	✓	✓	b3	✓				
s4	✓					b4		✓			
s5		✓				b5			✓		
s6			✓			b6				✓	✓
s7				✓	✓	b7				✓	
s8	✓					b8					✓
s9				✓							
s10					✓						



- Total Statement TCP: t1, t4, t5, t2, t3
- Additional Branch TCP: t5, t2, t1, t3, t4

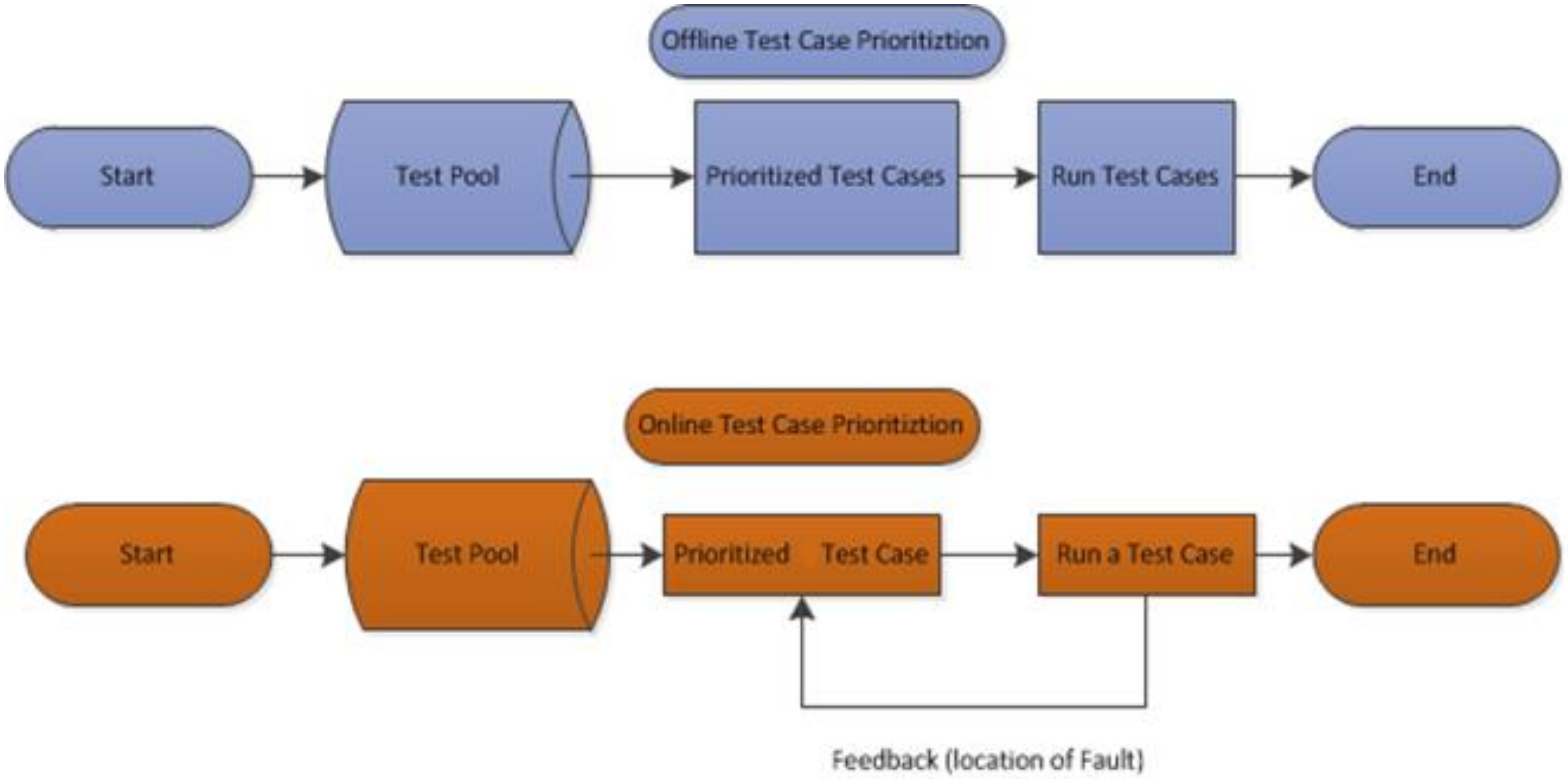
Code-level TCP

- Motivation
  - Even Distribution of Faults
    - 80-20 rule
  - Offline Method
    - Location of fault



- Can a TCP Technique be improved using the location of previously identified faults?
- How different coverage criteria could affect the effectiveness of TCP techniques based on the location of pervious detected faults?

Proposing Online TCP Techniques



Online TCP Techniques

Online Test Case Prioritization Algorithm

1: **function** SELECTION(ST, NST, P)
2: Choose t ∈ NST and $\sum_{s \in SC(t,P)}$ W(s, ST, NST, P) is maximum
3: **return** t
4: **end function**

ST: Selected Test cases, NST: not selected test cases, P: Program, s: statement, t: test case, W: calculates a weight for statement s with help of selected (ST), Not Selected Test cases (NST) and program (P)

Proposing Online TCP Techniques (Cont.)

Weight for each statements

```
1: function W(s,ST, NST,P)
2:     CoveredStatements = CS(ST ,P)
3:     if s ∈ CoveredStatements then
4:         return WeightCoveredStatments(s,ST)
5:     else
6:         return UCW
7:     end if
8: end function
```

UCW = 0 → OS 0, OB 0

UCW = 1 → OS 1, OB 1

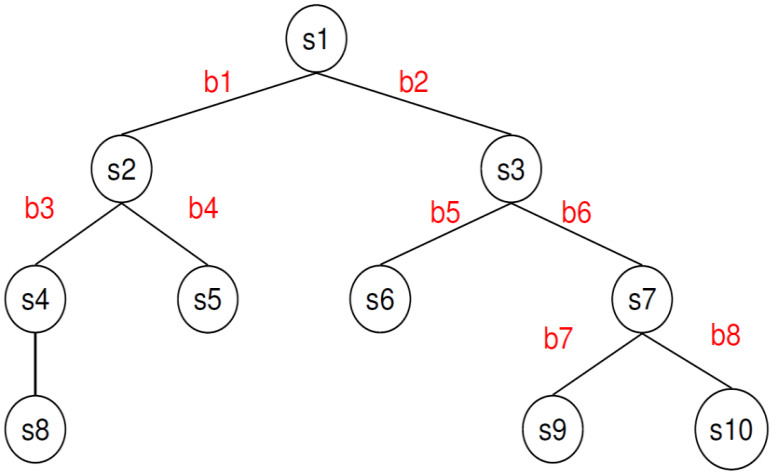
ST: Selected Test cases, NST: not selected test cases, P: Program, s: statement, t: test case, UCW: UnCovered unit weight, W: calculate a weight for statement s with help of selected (ST), Not Selected Test cases (NST) and program (P)

Weight for covered statements

```
1: function WeightCoveredStatments(s,ST,P)
2:     AF = ∪_{t∈NST} DF(t)
3:     CW = Σ_{(t∈NST)&(s∈SC(t,P))} |DF(t,p)|/AF
4:     return CW
5: end function
```

ST: Selected Test cases, NST: not selected test cases, P: Program, s: statement, t: test case, AF: a set of All detected Faults, CW: Covered unit Weight

Online TCP – Statement-Based Example



si	t1	t2	t3	t4	t5
s1	3/3	2/8+3/8+3/8	2/8+3/8+3/8	2/8+3/8+3/8	2/5+3/5
s2	3/3	3/8+3/8			
s3			2/8	2/8	2/5
s4	3/3				
s5		3/8			
s6			1		
s7				2/8	2/5
s8	3/3				
s9				1	
s10					2/5
Total	4	2.12	2.25	2.5	2.2

- Step 3
  - Selecting t2 with maximum weight
  - Given 3 detected faults by t2
  - Updating weights by 3/8
- Step 4
  - Selecting t4 with maximum weight
  - Given 4 detected faults by 4
  - Updating weights by 4/12
- Final order:
  - t1, t5, t2, t4, t3

fi	t1	t2	t3	t4	t5
f1	✓	✓			
f2			✓	✓	
f3					✓
f4			✓		
f5		✓			
f6	✓	✓			
f7			✓	✓	
f8				✓	
f9			✓	✓	✓
f10	✓				
Total	3	3	4	4	2

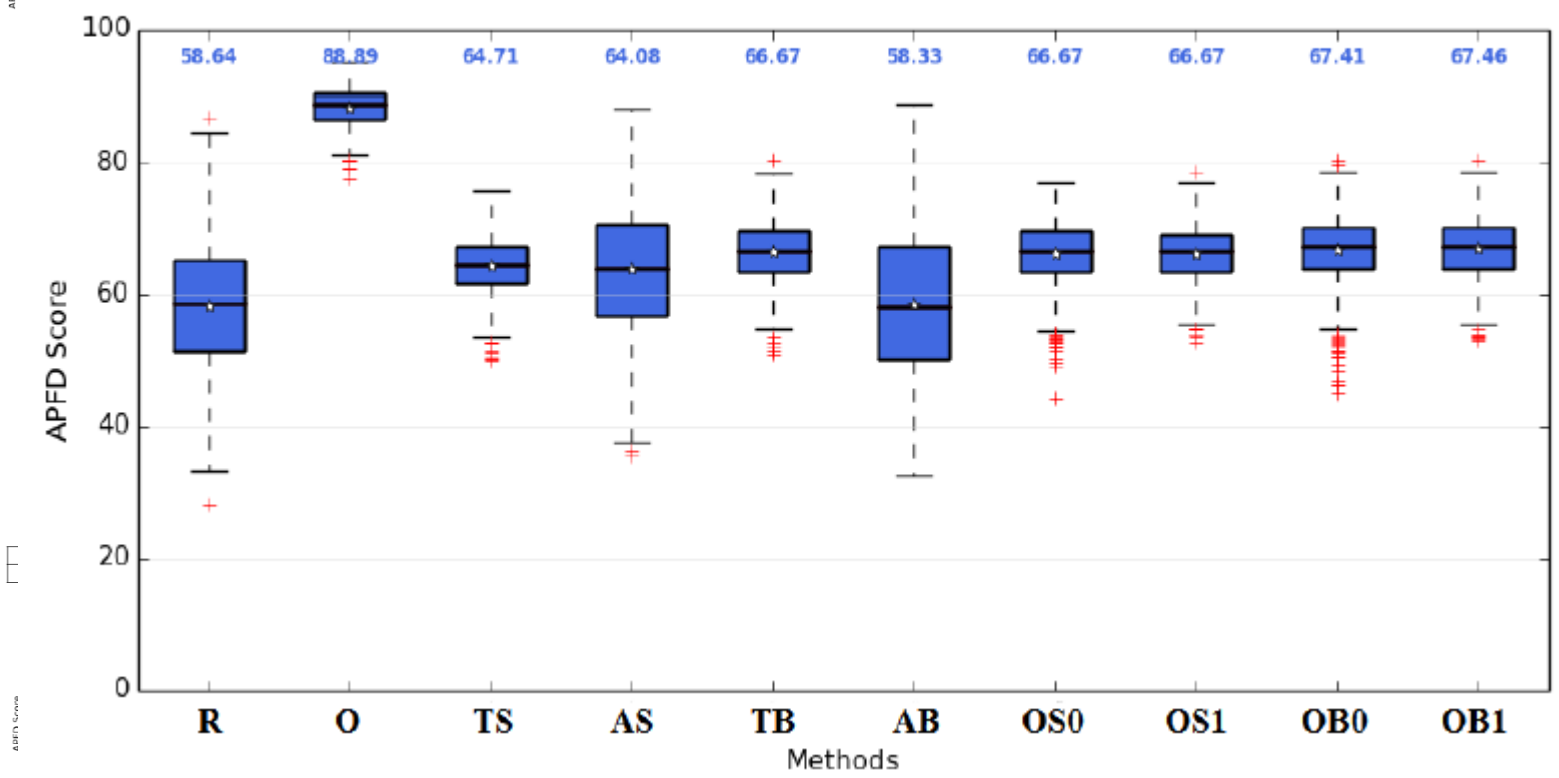
Experimental Objects

- Siemens Programs
  - Consists of 7 programs
  - Test case, faulty versions, and oracle version

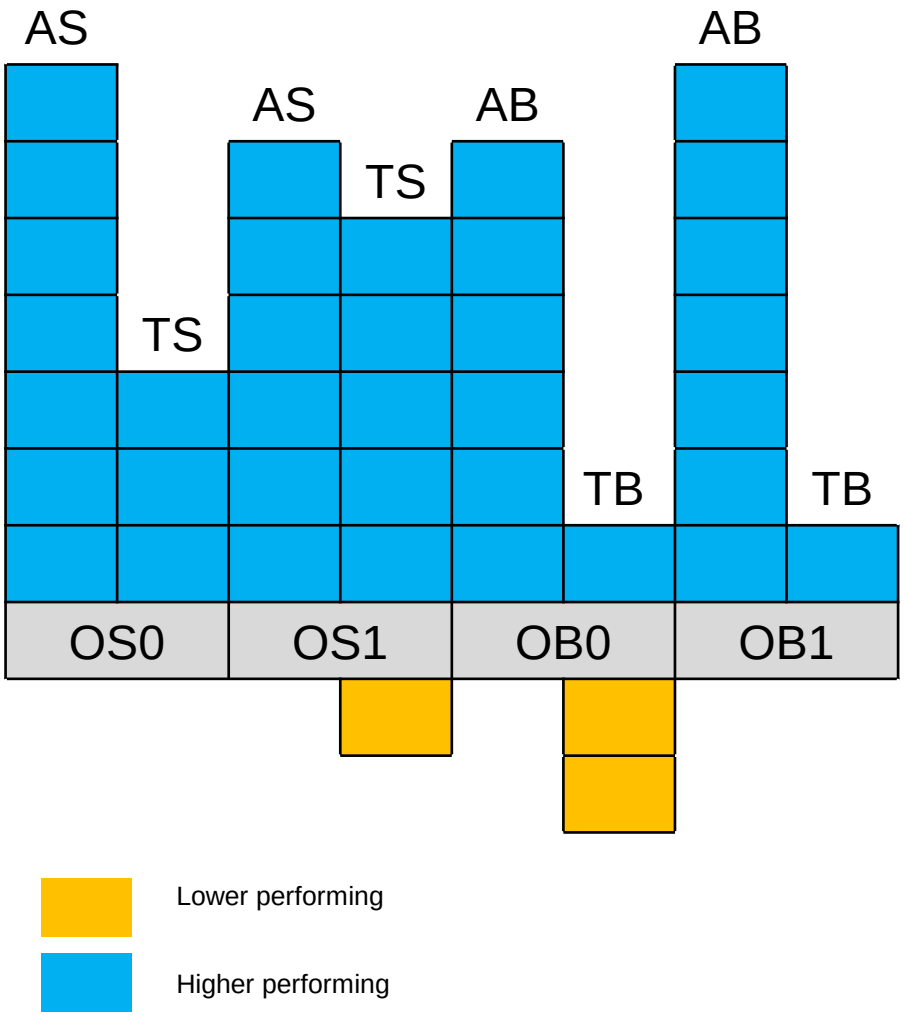
Program	LoC	Number of Faulty version	Number of test cases
print-tokens	726	7	4130
print-tokens2	570	10	4115
replace	564	32	5542
schedule	412	9	2650
schedule2	374	10	2710
tcas	173	41	1608
tot-info	565	23	1052

Experimental Study

Method	O	OB1	OB0	TB	OS1	OS0	TS	AS	AB	R
Grouping	a	b	bc	bc	c	c	d	d	e	e



Summary of Statistical Analyses



- Example 1:

OS0 outperforms baseline AS in 7 object programs.

- Example 2:

OB0 outperforms baseline TB in one object program, while twice has been outperformed by the TB.

Summary

- Can a TCP Technique be improved using the location of previously identified faults?
  - Our proposed techniques outperform baselines for Siemens experimental objects using APFD score.

- How different coverage criteria could affect the effectiveness of TCP techniques based on the location of previously detected faults?
  - Coverage criterion affects the performance
  - Comparing our methods with baselines
    - Both statement & branch-based outperform baselines
  - Comparing our proposed methods
    - Branch-based methods more effective than Statement-based


THANK
YOU

Mohsen Laali

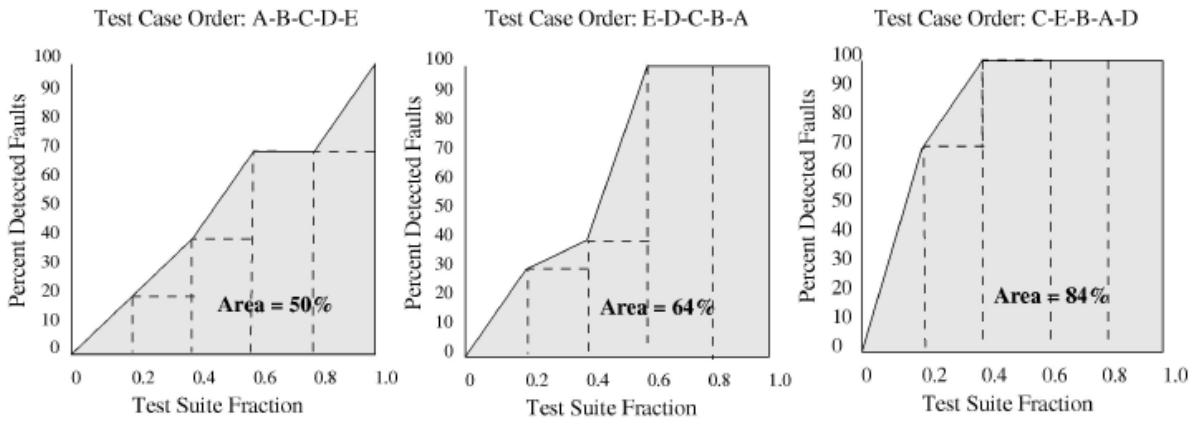
Mohsen.Laali@rmit.edu.au

Backup Slides

APFD

test	fault									
	1	2	3	4	5	6	7	8	9	10
A	x				x					
B	x				x	x	x			
C	x	x	x	x	x	x	x			
D					x					
E								x	x	x

(a)



(b) (c) (d)

Fig. 4. Example illustrating the APFD measure. (a) Test suite and faults exposed. (b) APFD for Prioitized Suite T1. (c) APFD for Prioritized Suite T2. (d) APFD for Prioritized Suite T3.

RQ 2) Human-level TCP

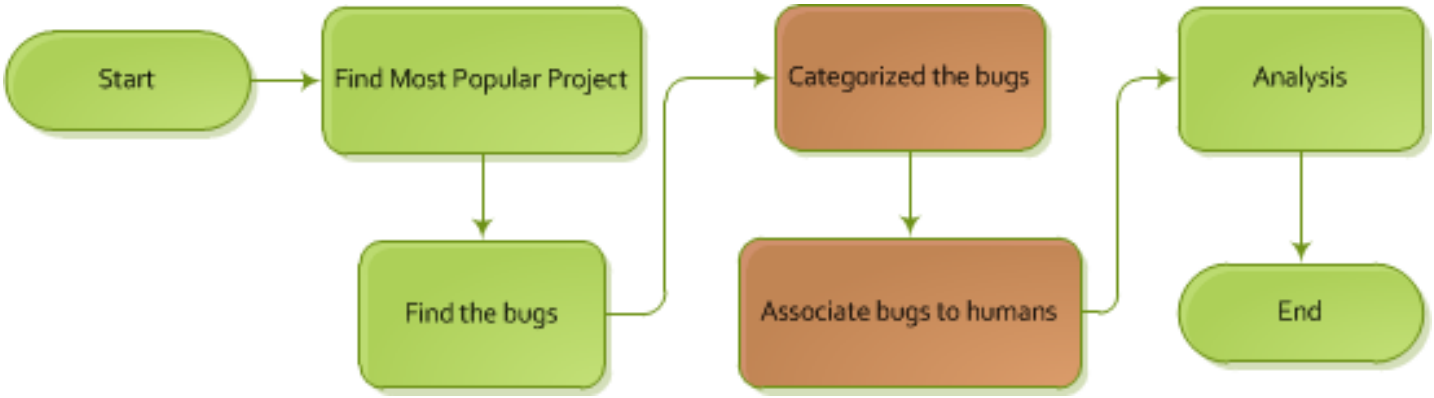
- RQ 2-1) How often do users repeat the same type of faults in software developments?
- RQ 2-2) What do affect the frequency of users' faults in software developments?
  - Volume of contributions to the code
  - Time of contributions to the code
  - Change rate of the code
  - User community of the code language

RQ 2) Human-level TCP

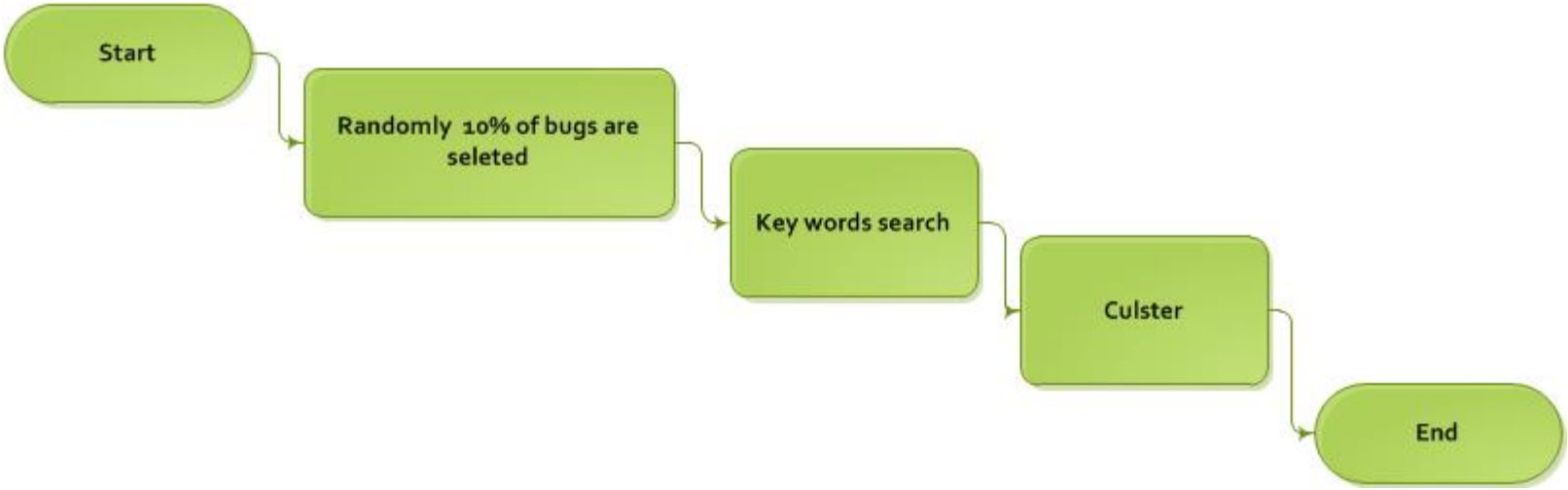
- Data
  - Git hub archive
- Methods
  - Pre-requisites
    - Human Error Classification
    - Error Associations
  - RQ 2-1
  - RQ 2-2

RQ 2) Human Classification Error

- Method

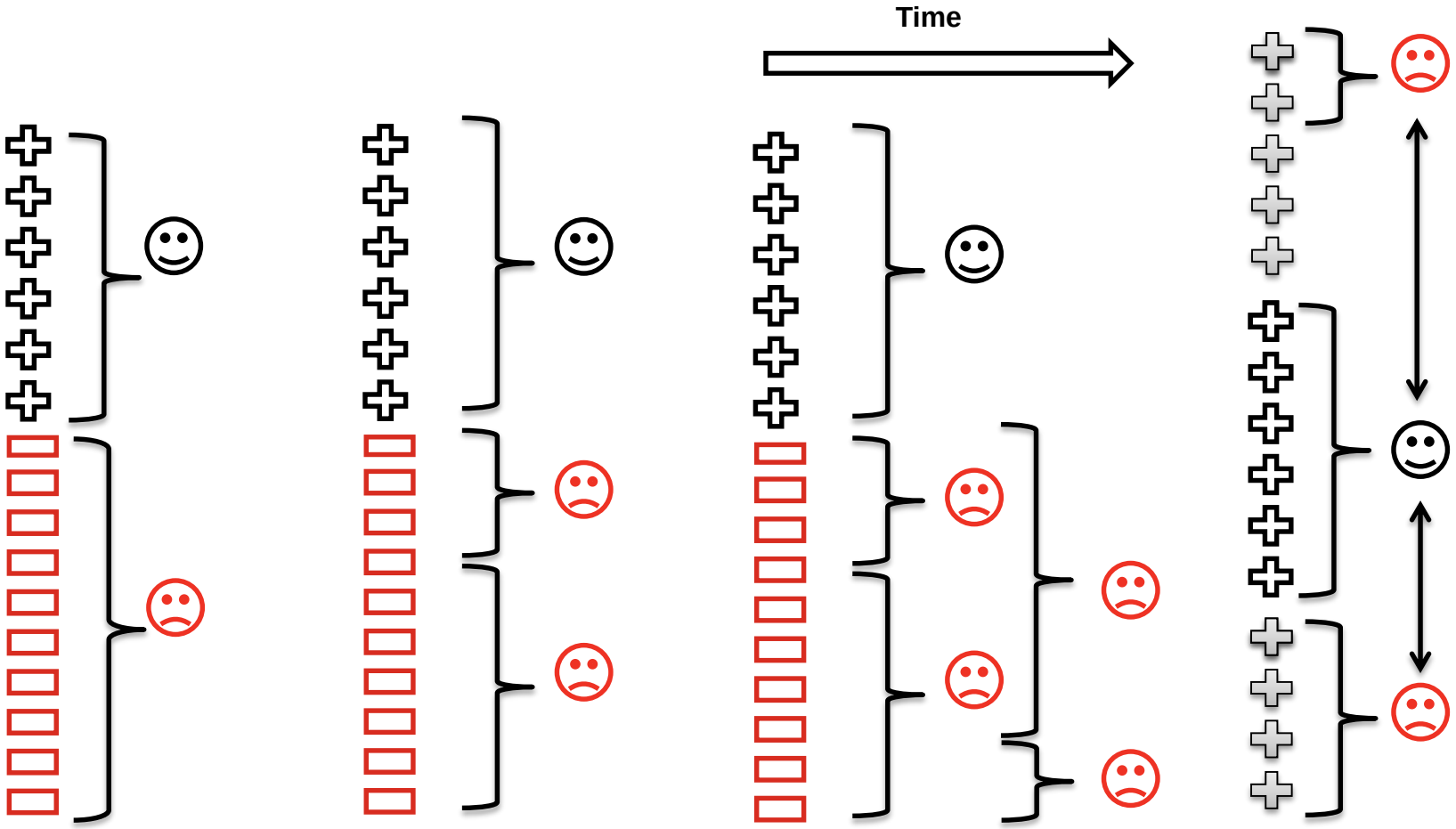


RQ 2) Bugs Classification



	Bug Type	Bug Description	Search keywords/phrases
Cause	**Algorithm** (Algo)	algorithmic or logical errors	algorithm
	Concurrancy (Conc)	multi-threading or multi-processing related issues	deadlock, race condition, synchronization error.
	Memory (Mem)	incorrect memory handling	memory leak, null pointer, buffer overflow, heap overflow, null pointer, dangling pointer, double free, segmentation fault.
	Programming (Prog)	generic programming errors	exception handling, error handling, type error, typo, compilation error, copy-paste error, refactoring, missing switch case, faulty initialization, default value.
Impact	**Security** (Sec)	correctly runs but can be exploited by attackers	buffer overflow, security, password, oauth, ssl
	Performance (Perf)	correctly runs with delayed response	optimization problem, performance
	Failure (Fail)	crash or hang	reboot, crash, hang, restart
	Unknown (Unkn)	not part of the above seven categories	

RQ 2) Error Associations



Future Work

- Online TCP Policy
  - 80-20 rule
  - Captured 20% by the online execution of test cases

- Possible Improvements
  - Who makes the faults?
  - Human habits in making faults
  - Capturing 20% incorporating human factors