Are the Scala Checkers Effective? Evaluating Checkers with Real-world Projects

Xin Zhang, Jiwei Yan, Baoquan Cui, Jun Yan, Jian Zhang
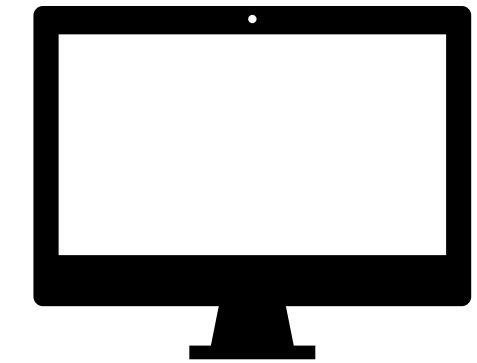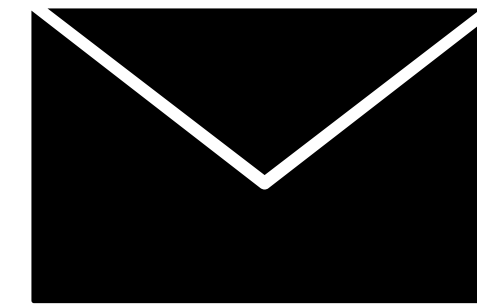
Outline

- Introduction

- Motivation

- Approach

- Experiments

- Conclusion

191176
Warnings

Scala checkers are widely used in practice.
115 Checkers throw 191K warnings on 6 Scala projects.
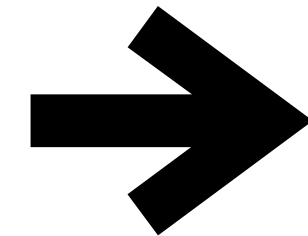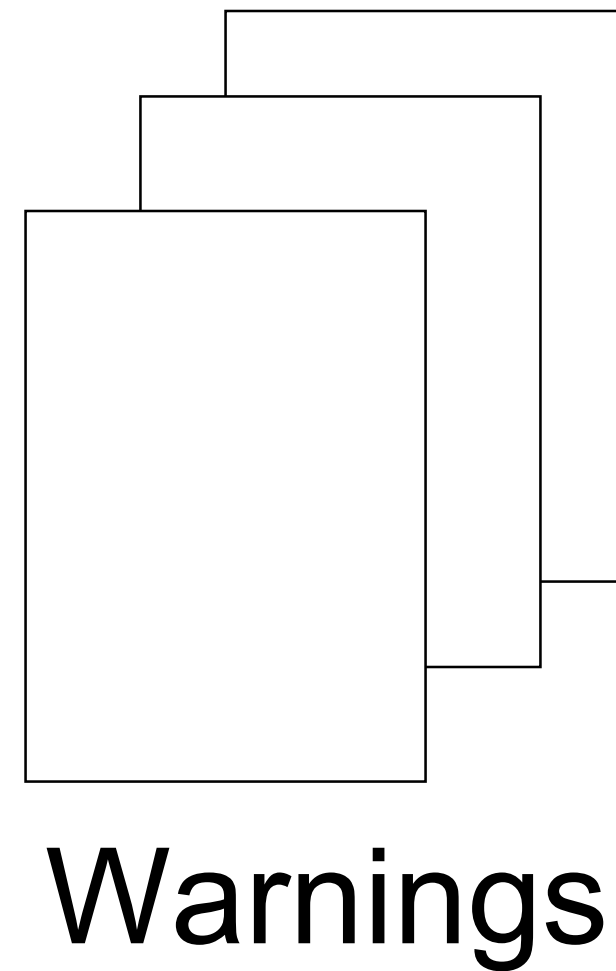
Existing Evaluation Approaches
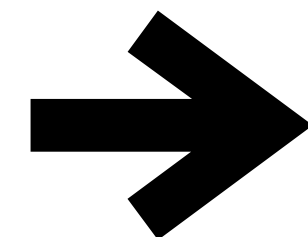
- Interviews

- Questionnaires

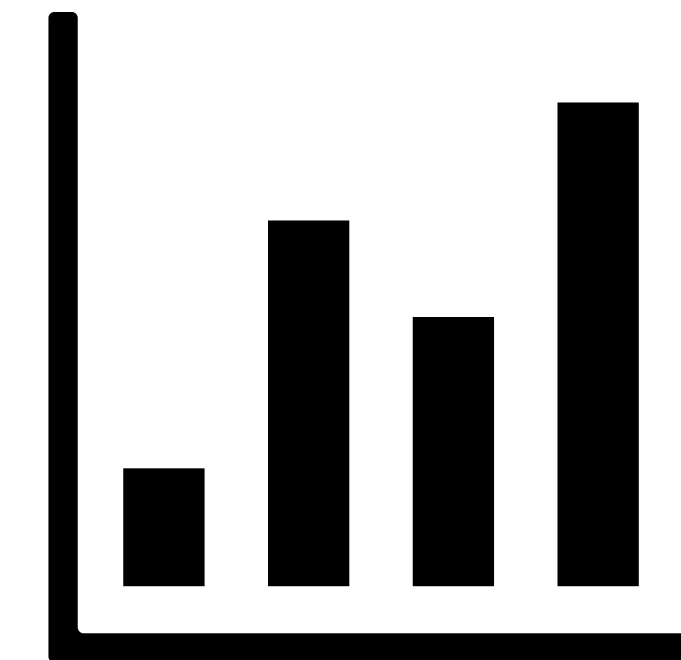- Manual Inspection

191K Warnings

Our Target
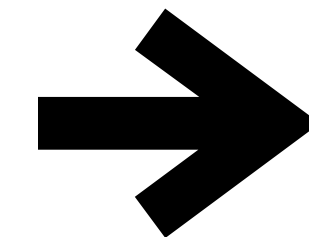
Evaluating Scala checkers automatically
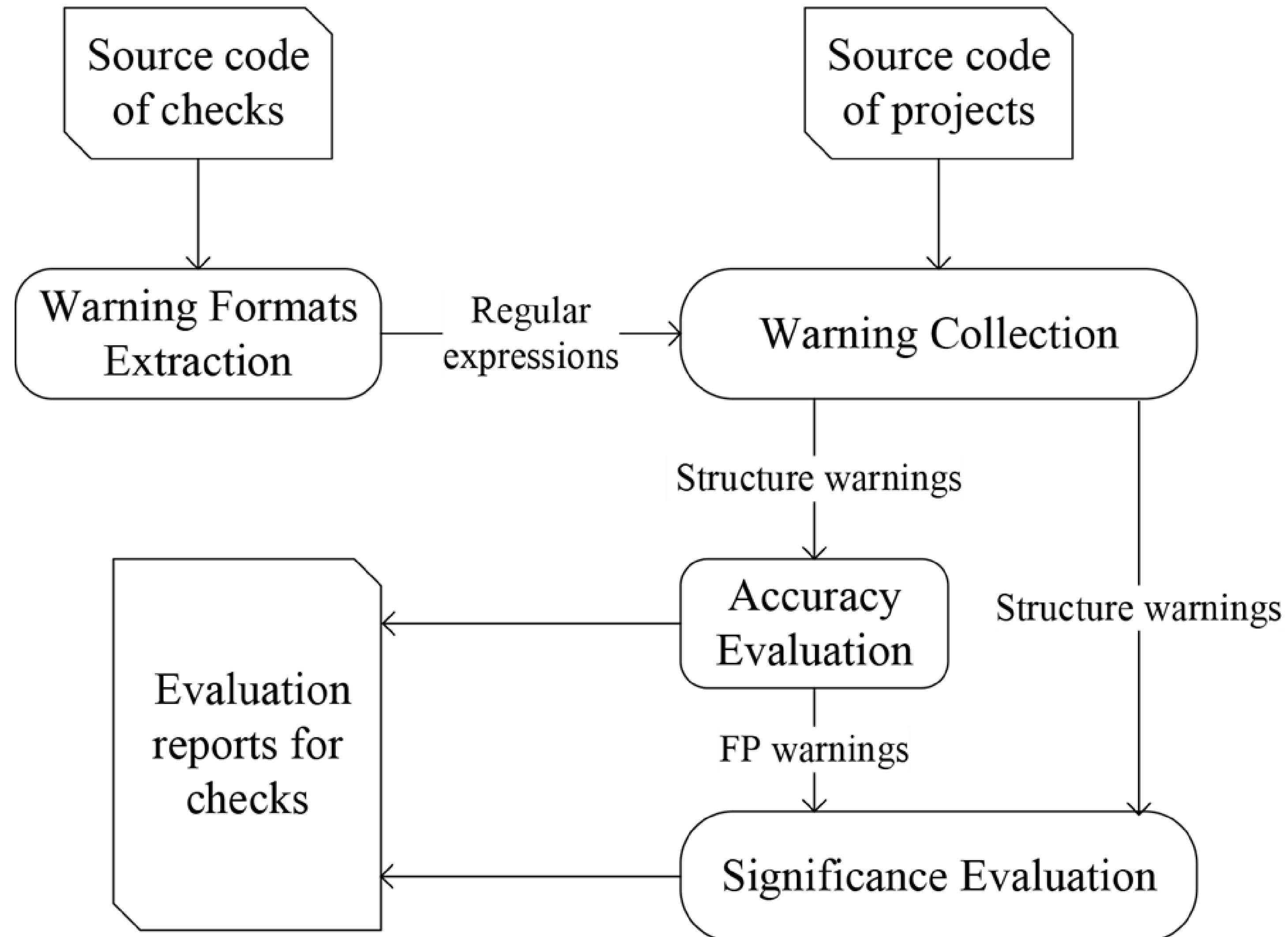

Warnings

Significance

Accuracy

Evaluation results for checkers

Our Approach

Warning Collection

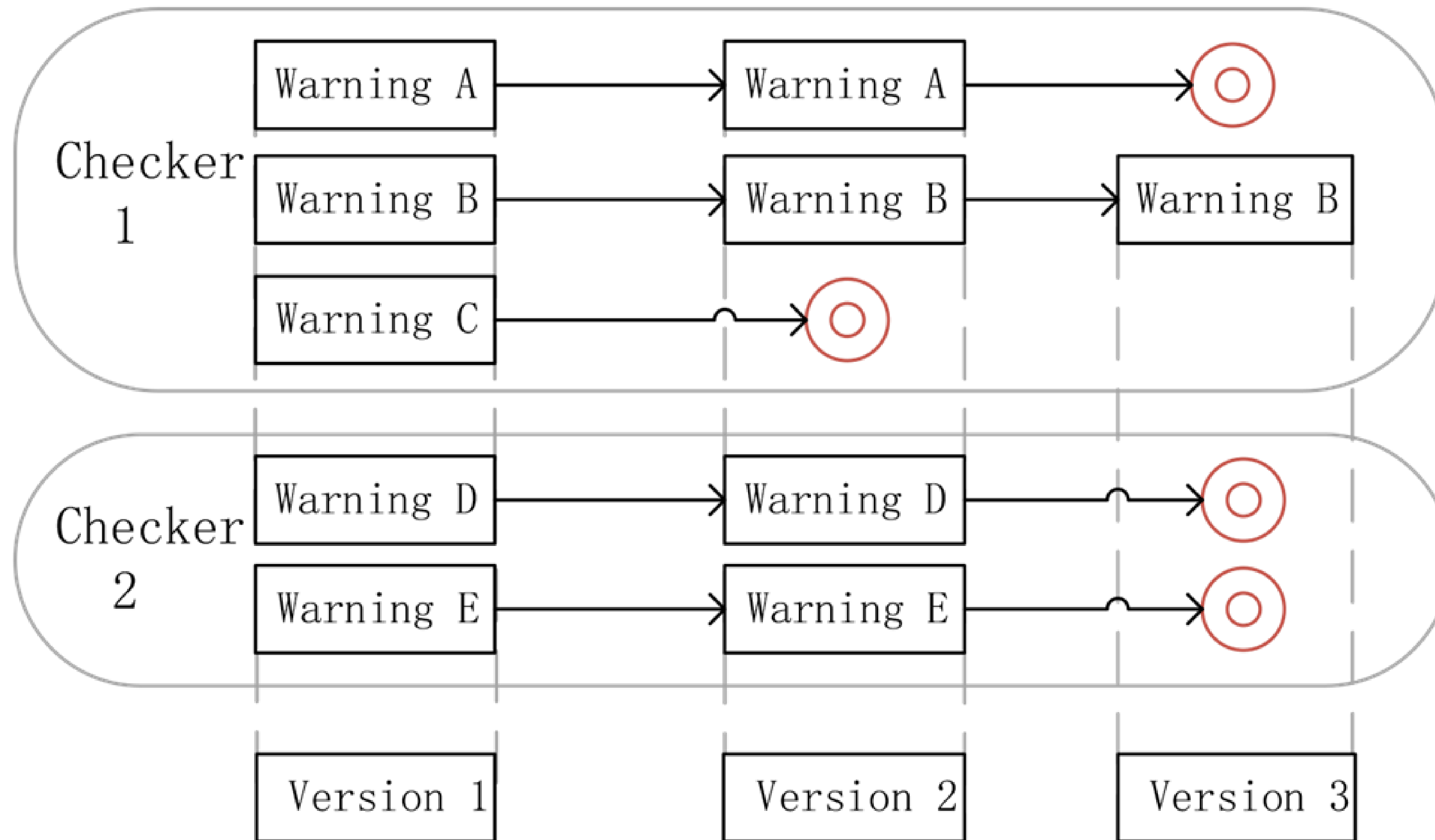- Scanning the configuration files and add checkers in.

```
53:  val bootstrapScalaVersion = "2.11.5"

...

71:  scalaVersion := bootstrapScalaVersion,

// specify the Scala version

...

147: scalacOptions += "-Xplugin:scapegoat_2.11.jar",

// inserted by the module
```

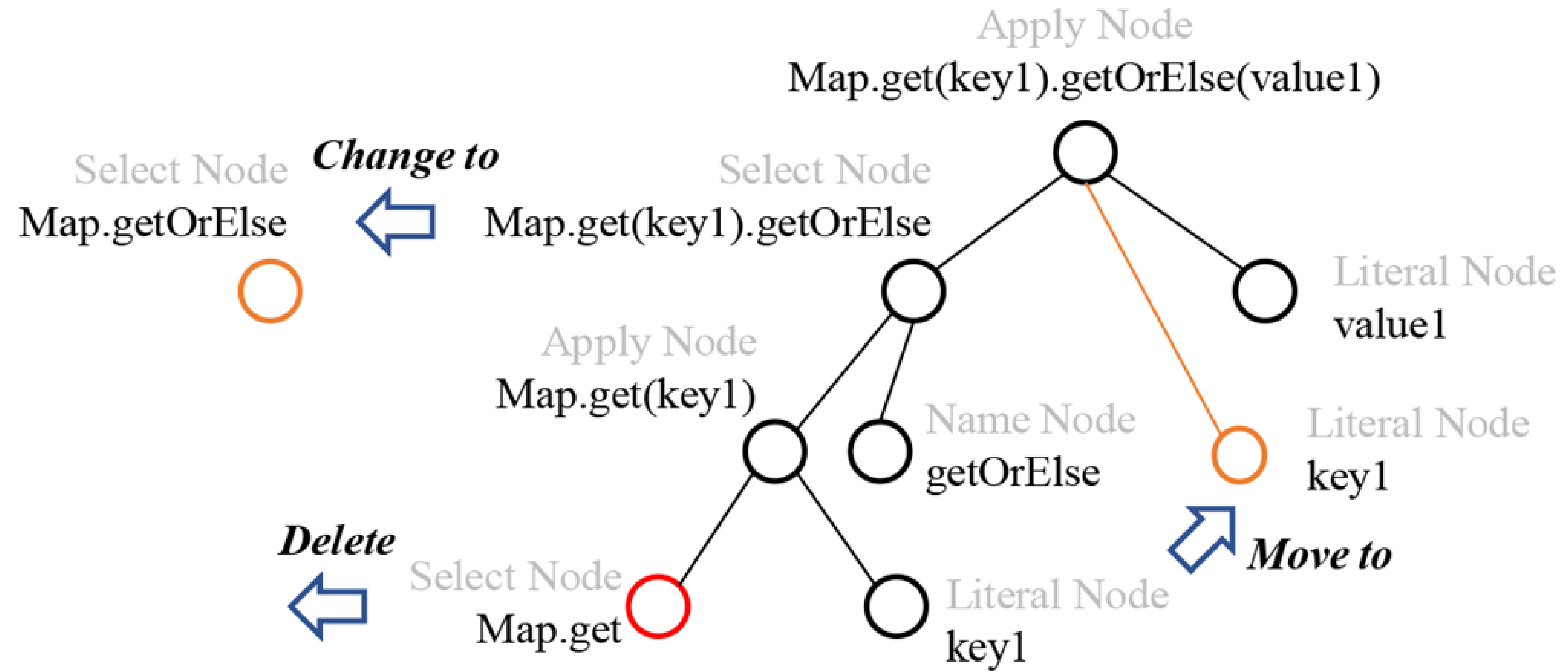# Significance Evaluation

Evaluation based on warning lasting versions

Accuracy Evaluation

Evaluation based on fixing patches

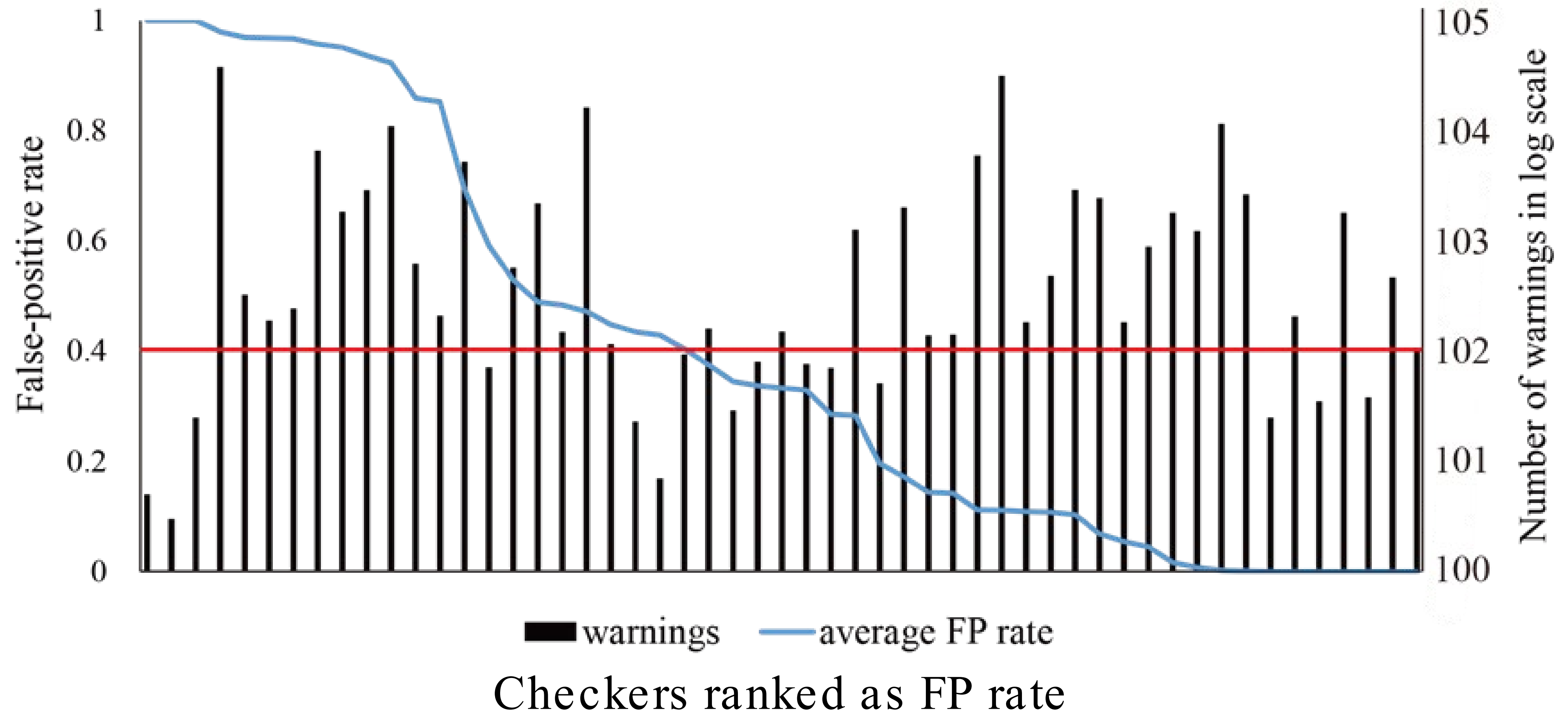# Experiment Dataset

Project	LoC	Stars	#Ver.	#W. 10k.	#W. V.
Scala	12,950	435K	51	2,237	3
Akka	11,147	24K	139	3,335	38
Gitbucket	8,171	15K	213	331	25
Scalaz	4,392	58K	10	55	67
Slick	2,396	32K	113	368	20
Elastic4s	1,516	25K	103	259	150

RQ1: The Accuracy of Checkers

The accuracy of checkers on all the dataset

Checkers ranked as FP rate

RQ1: The Accuracy of Checkers

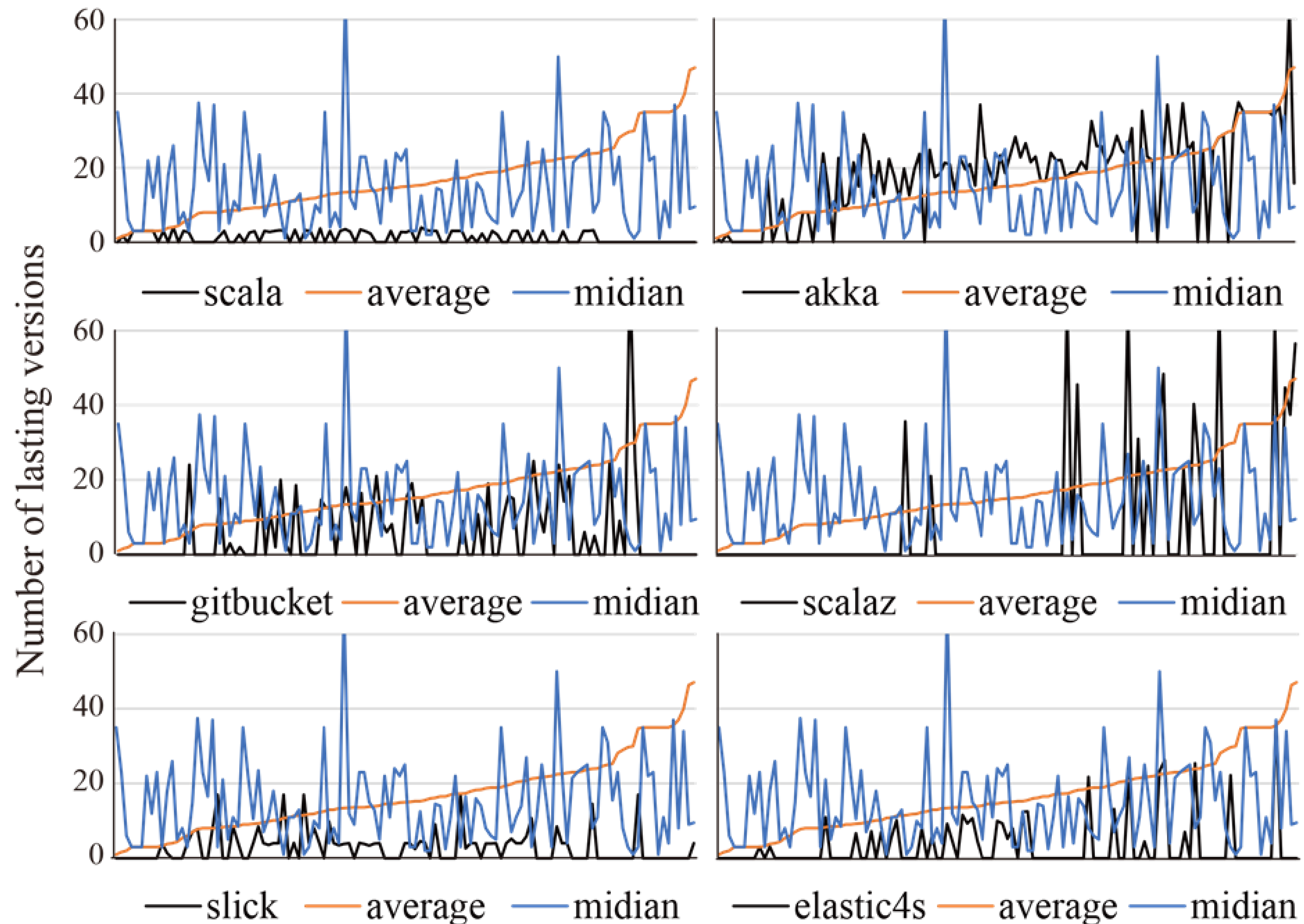**The accuracy of checkers on different projects**
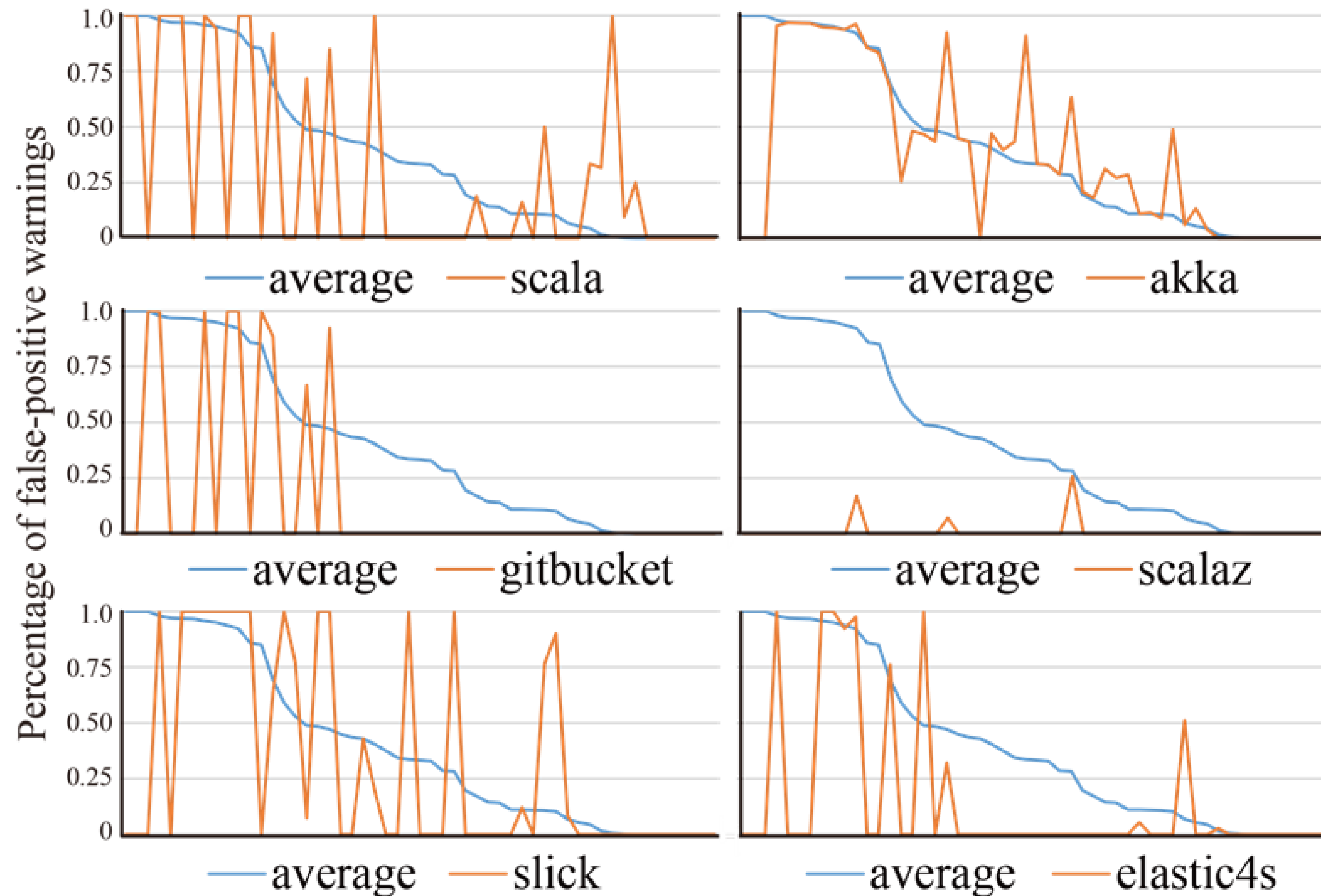
RQ1: The Accuracy of Checkers

- Checkers have a considerable false-positive rate, particularly those with a high number of warnings.

- The false-positive rates of a single checker may vary greatly between projects.

RQ2: The Significance of Checkers

The accuracy of checkers on different projects

RQ2: The Significance of Checkers

- The significance of checkers varies between projects and has a poor relationship with the number of warnings.

RQ3: The Efficiency of Warning Tracking

- Owing to code movement, half of the warnings would be missed in the precise match in most cases.
- Cross-file code movement does not affect the warning track in our approach.

Conclusion

- We describe an automated approach for analyzing the significance and accuracy of checkers on open-source project evolution.

- 1/5 of checkers had both high accuracy and significance, implying that they can deliver more significant alarms

- Our approach may help users choose checkers that conform to the coding styles of their projects, as well as developers identify the cause of false positives and improve code.

Thanks for watching

Q & A