Ranking source code static analysis warnings for continuous monitoring of FLOSS repositories

Athos Ribeiro Paulo Meirelles Nelson Lago
Fabio Kon

Institute of Mathematics and Statistics
University of São Paulo

Federal University of São Paulo

June 2018

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

Who am I?

- Former guest researcher at the US National Institute of Standards and Technology
- Masters degree student at University of São Paulo
- Fedora Project contributor

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

What is Static Analysis?

- Extract facts from the **source code** to
- find **flaws**
- by reporting **warnings**.
- Warnings are aggregated in **reports**
- Reports are generated by **static analyzers**

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

Why use Static Analysis?

- Find (and act on) **flaws**
- by exploring abstractions of **many** program behaviors
  - **Hard** to do with automated testing
- to increase software **quality**

Motivation

Undecidability

- Fundamental static analysis problems are **undecidable**
- Heuristics $\rightarrow$ **false positives** and **false negatives**

- **Reports** contain too much (misleading) information
- More misleading information $\rightarrow$ Less valuable tool
- Too much misleading information $\rightarrow$ **Worthless** tool

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

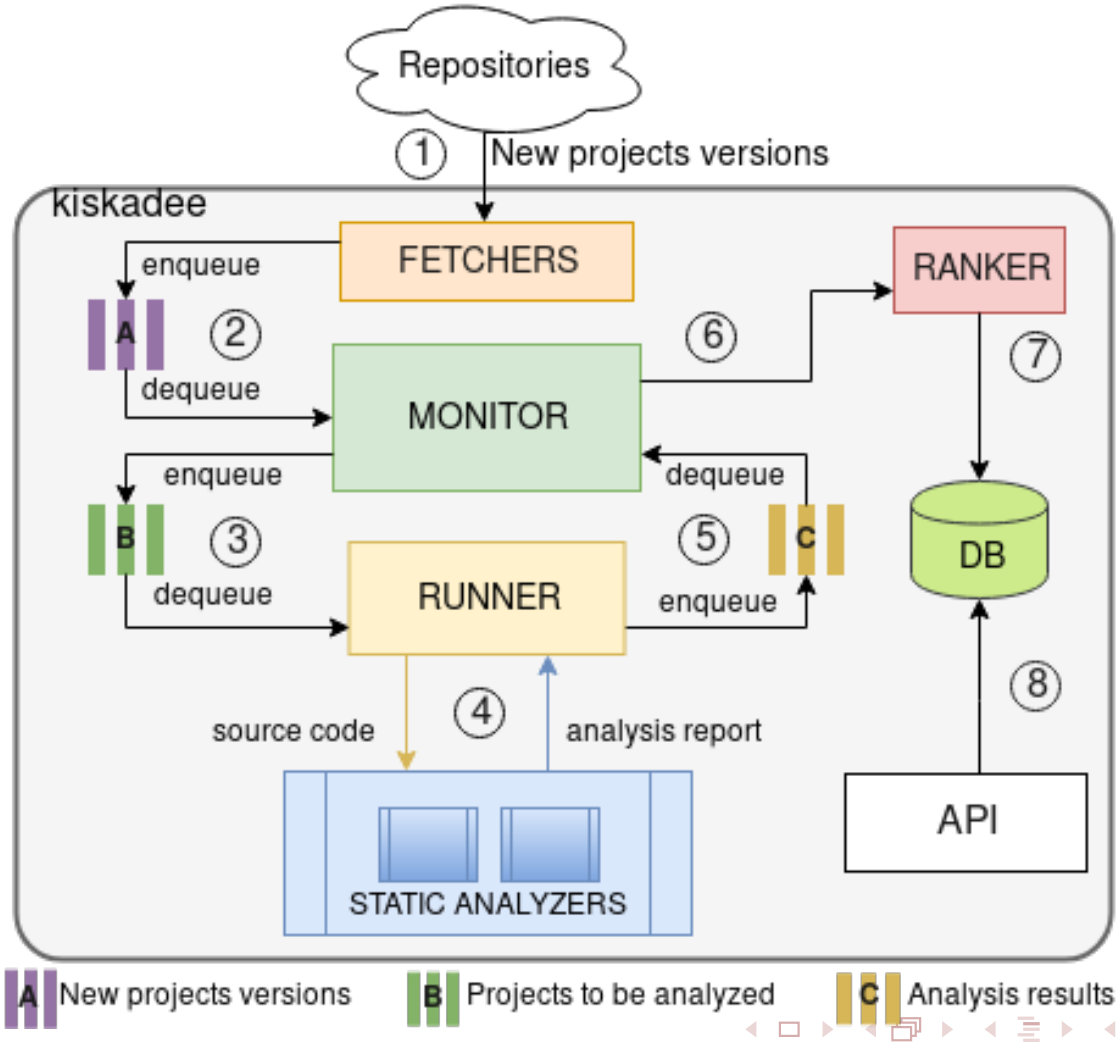
Possible solution...

- Use **multiple** static analyzers
  - Increase coverage $\rightarrow$ reduce false negatives
- Apply a classification approach on the output (warnings)
  - Discard (**possible**) false positives

kiskadee: a continuous static analysis tool

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

kiskadee design overview



CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE FLOSS Competence Center

We start with a classification problem...

- True Positive / False Positive
- No information on the analyzed project
- Weak classifiers
  - AdaBoost
    - Boost decision trees
    - Can use both categorical and non-categorical features
    - DT are easy to train
- Supervised learning
  - We need a set of known flaws to train our model

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

Juliet: Static analysis test suite

- NIST/SAMATE $\rightarrow$ **Juliet**
- Test suite with **61,387 test cases**
- **118** different **CWEs**
- C programming language
- Specific flaws in **known locations**

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

Preparing a dataset from Juliet

	Before Pruning	After Pruning
C	36,078	22,459
C++	25,309	16,641
Total	61,387	**39,100**

Table: Number of Juliet test cases

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE FLOSS Competence Center

Running static analyzers on our Juliet subset

Tool	Warnings
Clang Static Analyzer	37,229
Cppcheck	124,025
Frama-C	120,573
Total	**281,827**

Table: Total number of warnings generated per tool

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

Labeling warnings

Tool	Warnings	TP	FP	FP Rate	Precision
Clang Analyzer	6207	984	5223	0.84	0.16
Cppcheck	4035	314	3721	0.92	0.08
Frama-C	15717	8892	6825	0.43	0.57
Aggregated tools	25959	10190	15769	**0.61**	**0.39**

Table: Labeled warnings per tool

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

Features and final dataset

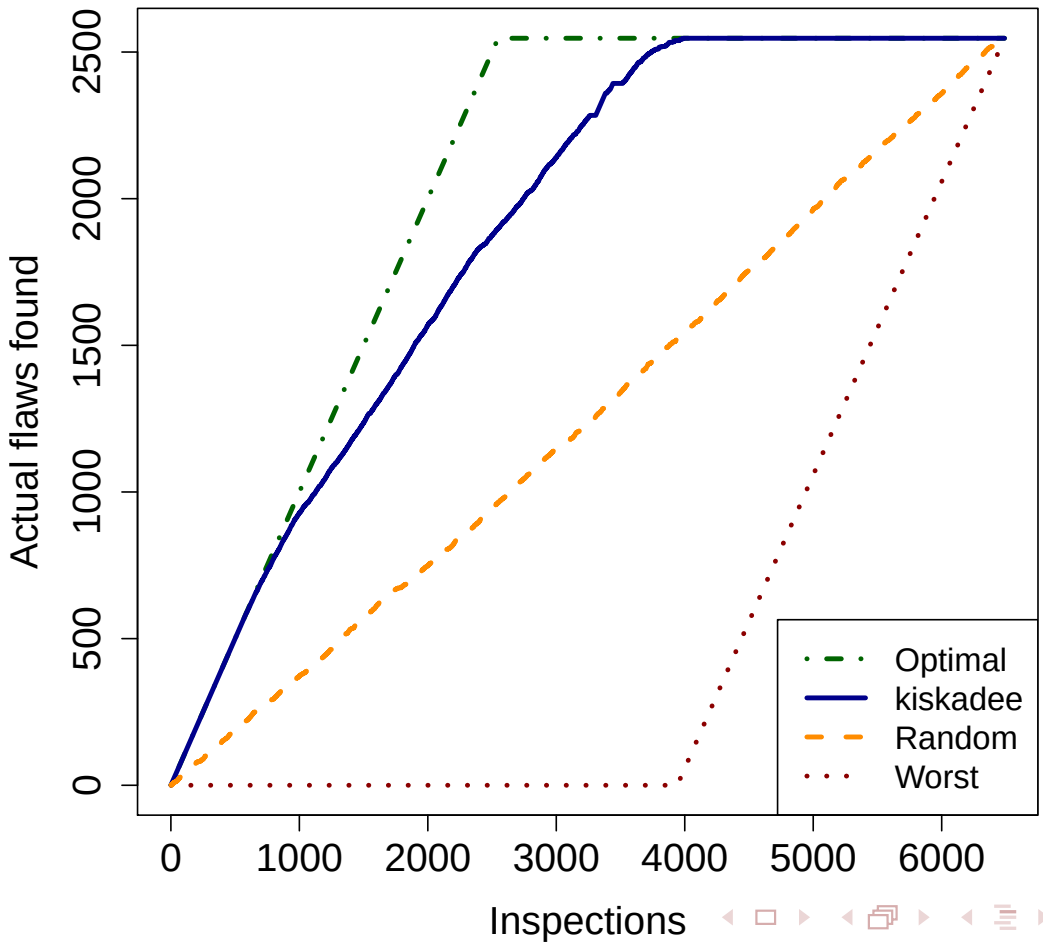
- Dataset with features to train AdaBoost classifier extracted from labeled warnings
- Features
  - Tool name
  - Programming language
  - Severity
  - Redundancy (with tool names)
  - Number of neighbors
  - Category
  - Number of warnings in the same file
  - **True**/**False** positive

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

Ranking warnings

- Use AdaBoost classifier probabilities
- Warnings with higher probabilities of being True Positives to the top
- Warnings with lower probabilities of being True Positives to the bottom

Inspections vs flaws found



CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

Conclusions

- FLOSS static analyzers show poor FP rates
- Ranking model does not use features based on the analyzed project
- We trade accuracy for generalization

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

Thanks to

- University of Brasília
  - All *Software Maintenance and Evolution* course students who worked on kiskadee
- David Silva
- Google Summer of Code
- Fedora Project

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center

Questions?

athoscr@ime.usp.br

pagure.io/kiskadee

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center