Università della Svizzera italiana

Faculty of Informatics

AutoBench

Finding Workloads That You Need Using Pluggable Hybrid Analysis

Yudi Zheng, Andrea Rosà, Luca Salucci, Yao Li, Haiyang Sun, Omar Javed, Lubomír Bulej, Lydia Y. Chen, Zhengwei Qi and Walter Binder

PhD candidate
Dynamic Analysis Group
Università della Svizzera Italiana (USI)
Lugano, Switzerland

IEEE SANER 2016
March 17th, 2016
Osaka, Japan

Research question

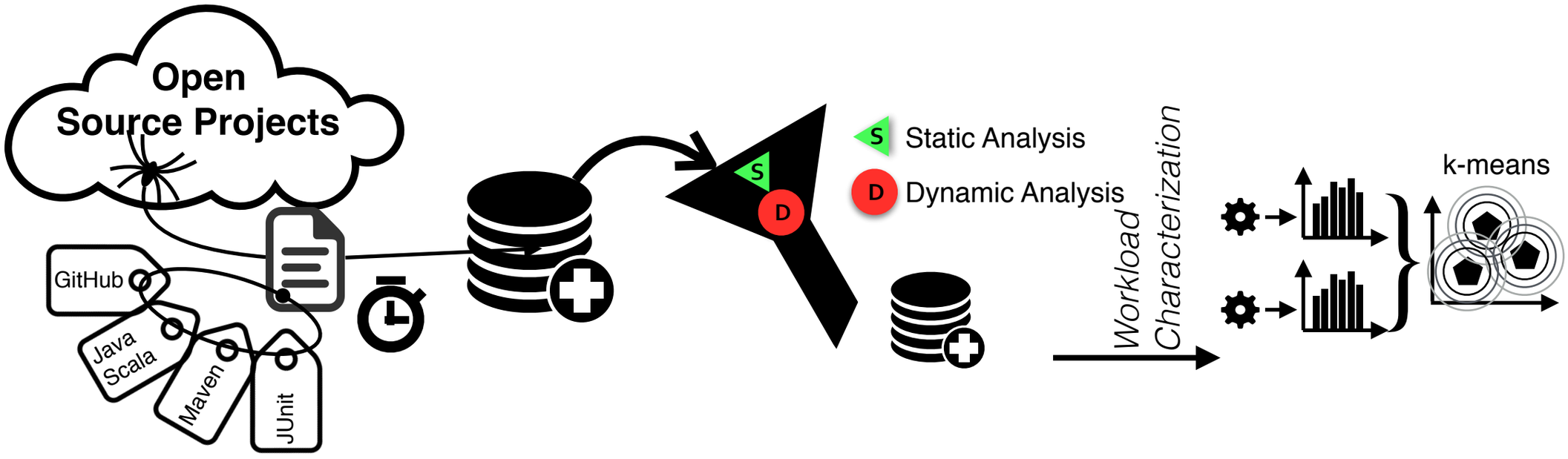
Can a **fully automated** process find **open-source workloads** that are suitable as **benchmarks** for **specific** evaluation needs?

- Definitions:
  - **Workload**: the execution profile of a running application
  - **Benchmark**: workload considered as reference for a given domain
- Motivation:
  - Lack of benchmark suites targeting **specific needs**
  - First steps towards **automatic benchmark synthesis**

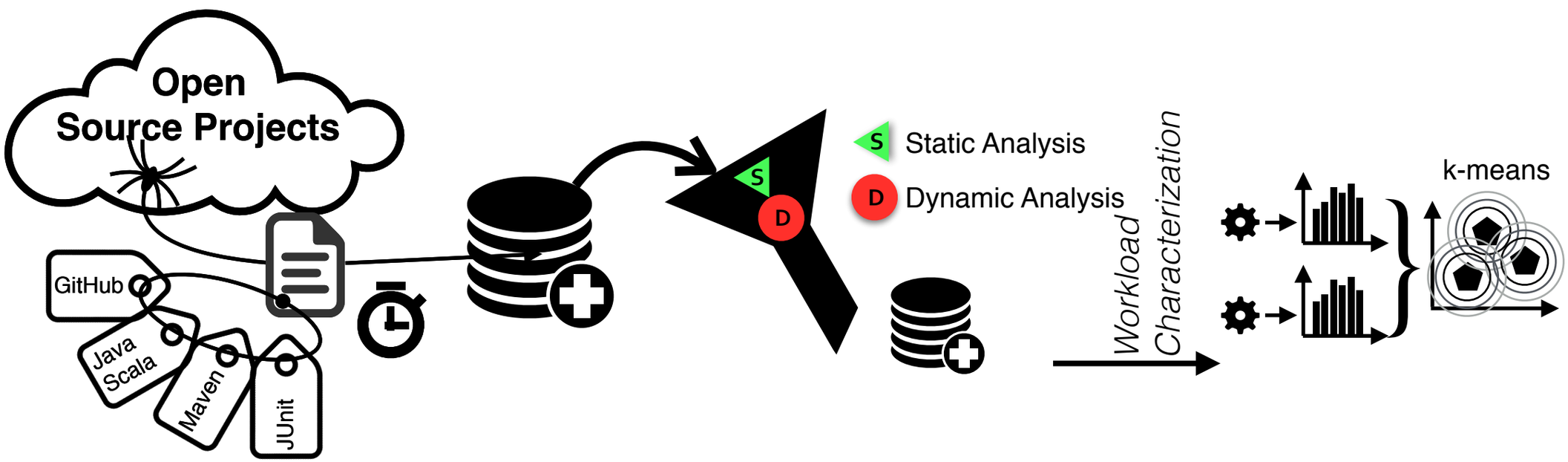
AutoBench

- A methodology and toolchain to **automatically find workloads** and synthesize **benchmark suites** from open-source projects

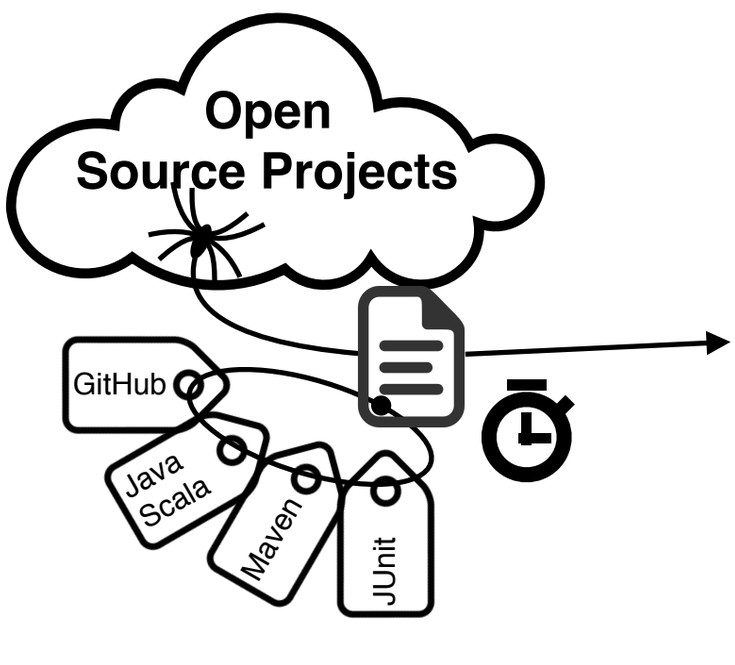
  - Workloads = **Unit tests**



AutoBench


Open
Source Projects
GitHub
Java
Scala
Maven
JUnit
S
D
Static Analysis
Dynamic Analysis
Workload
Characterization
k-means

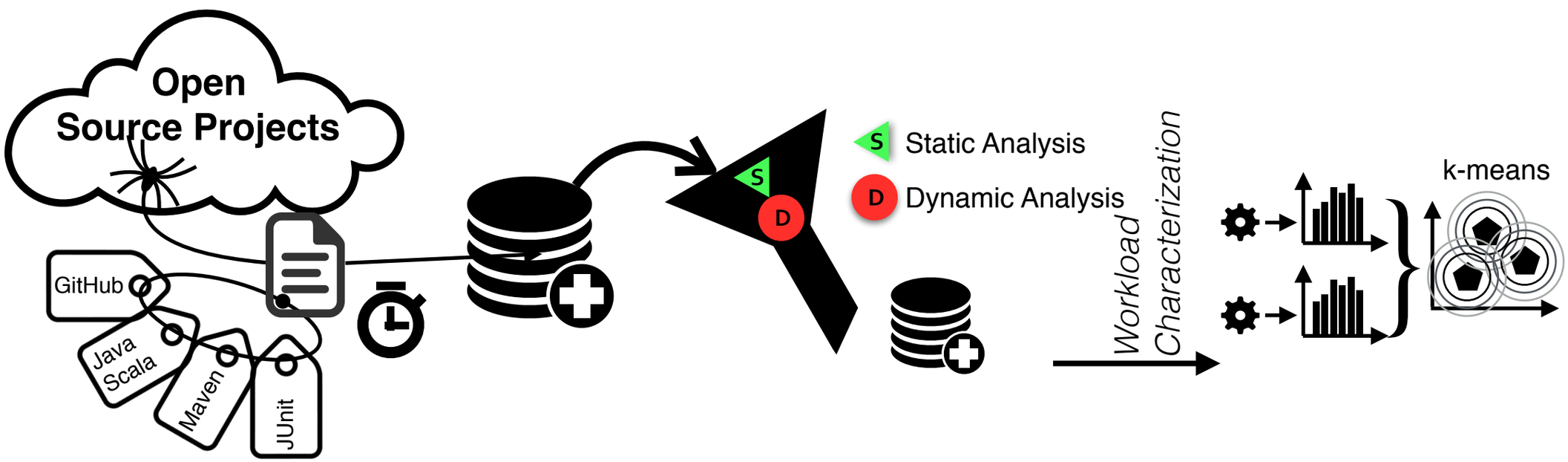
AutoBench



Finding

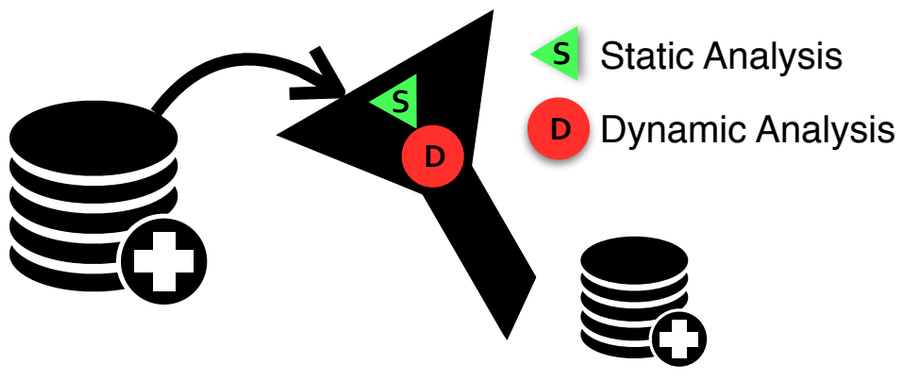
- Find a base of **executable** and **significative** workloads from **open-source** projects
  - **Significative:** workloads with execution time >= **T**

AutoBench



Finding

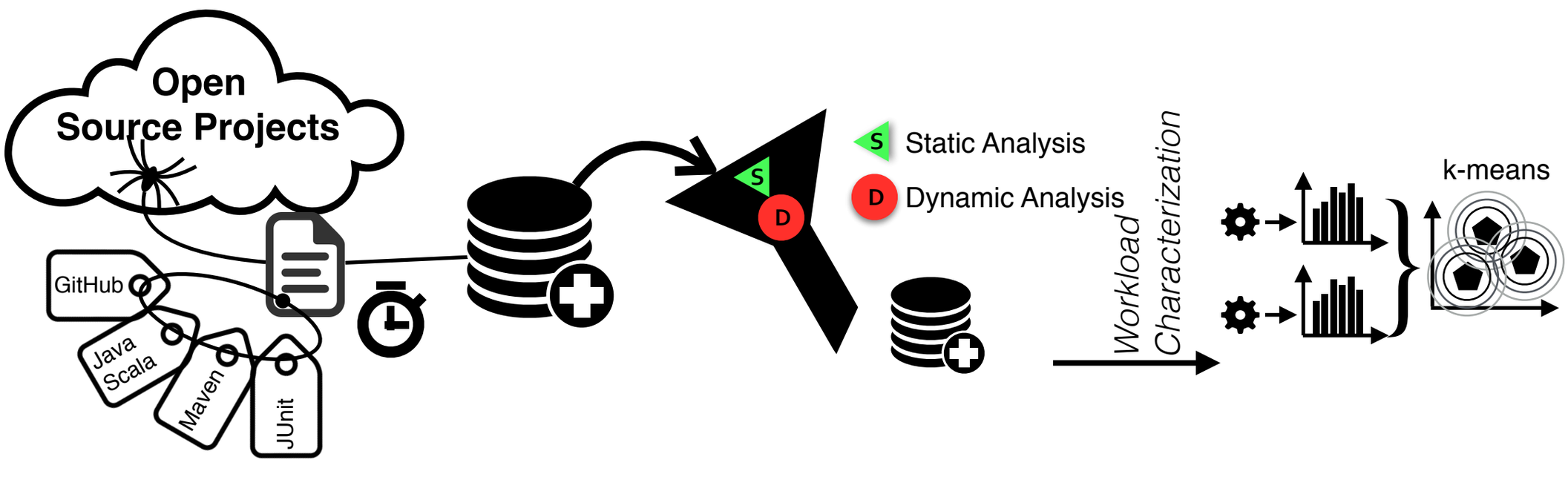
AutoBench



Finding **Filtering**

- Preserve only workloads matching **specific needs**
  - Filter pipeline
    - Static/dynamic analyses written by users
      - **Dynamic analysis** refines results of static analysis
    - Analyses reflect specific user requirements
    - Workloads must pass all filters to be considered **suitable**

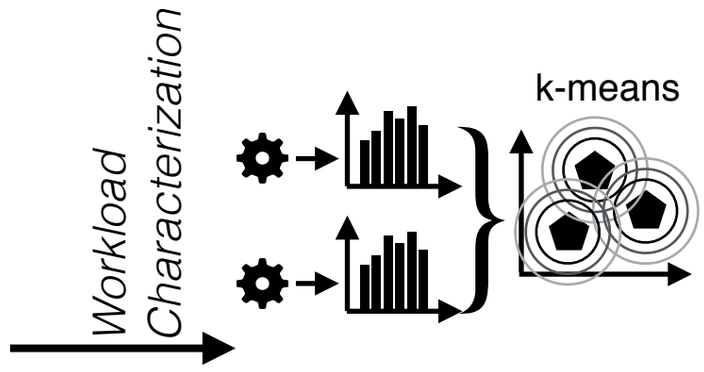
AutoBench



Finding

Filtering

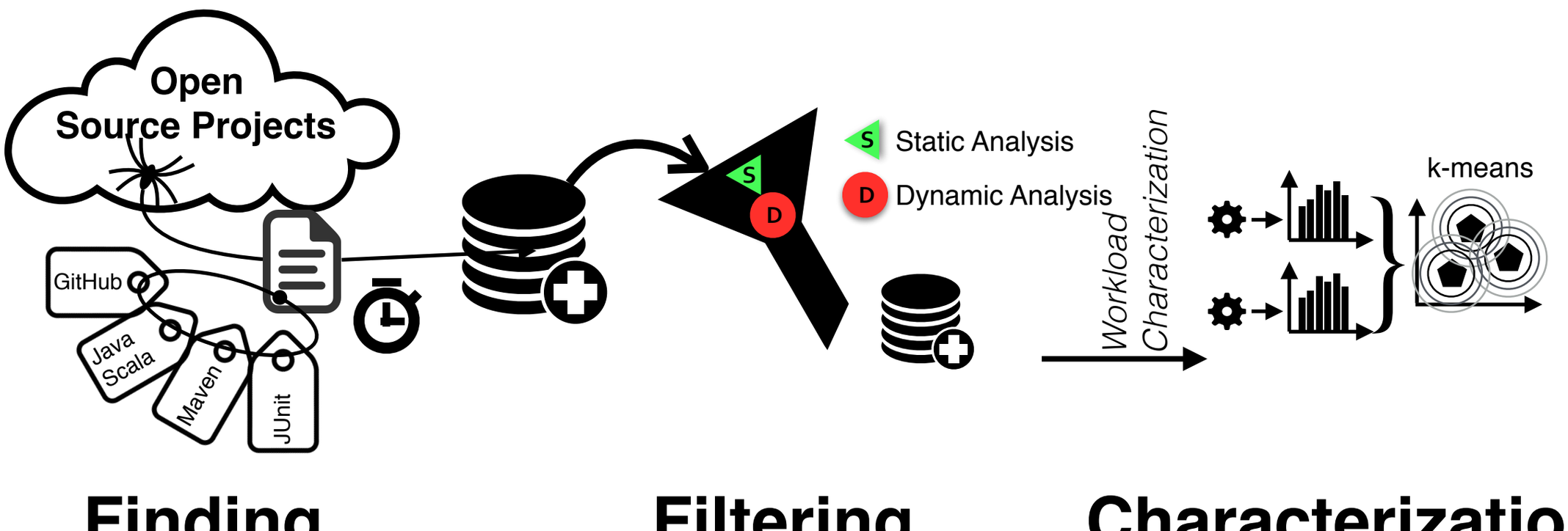
AutoBench



Finding **Filtering** **Characterization**

- Workload characterization identifies a **diverse** set of **K** benchmarks
- Diversity metrics:
  - Execution time
  - N. of object allocations
  - N. of method invocations
  - …

AutoBench



Finding

Filtering

Characterization

Evaluation

Can a fully automated process find open-source workloads that are suitable as benchmarks for specific evaluation needs?

1. **Significance**: Can we expect to find tests that are long enough to serve as benchmark workload?
2. **Suitability**: Can dynamic analysis improve identification of workloads matching a specific evaluation context?
3. **Diversity**: Can we find diversity in the workloads that would allow synthesizing a benchmark suite?

T = 1s **K = 12**

Evaluation

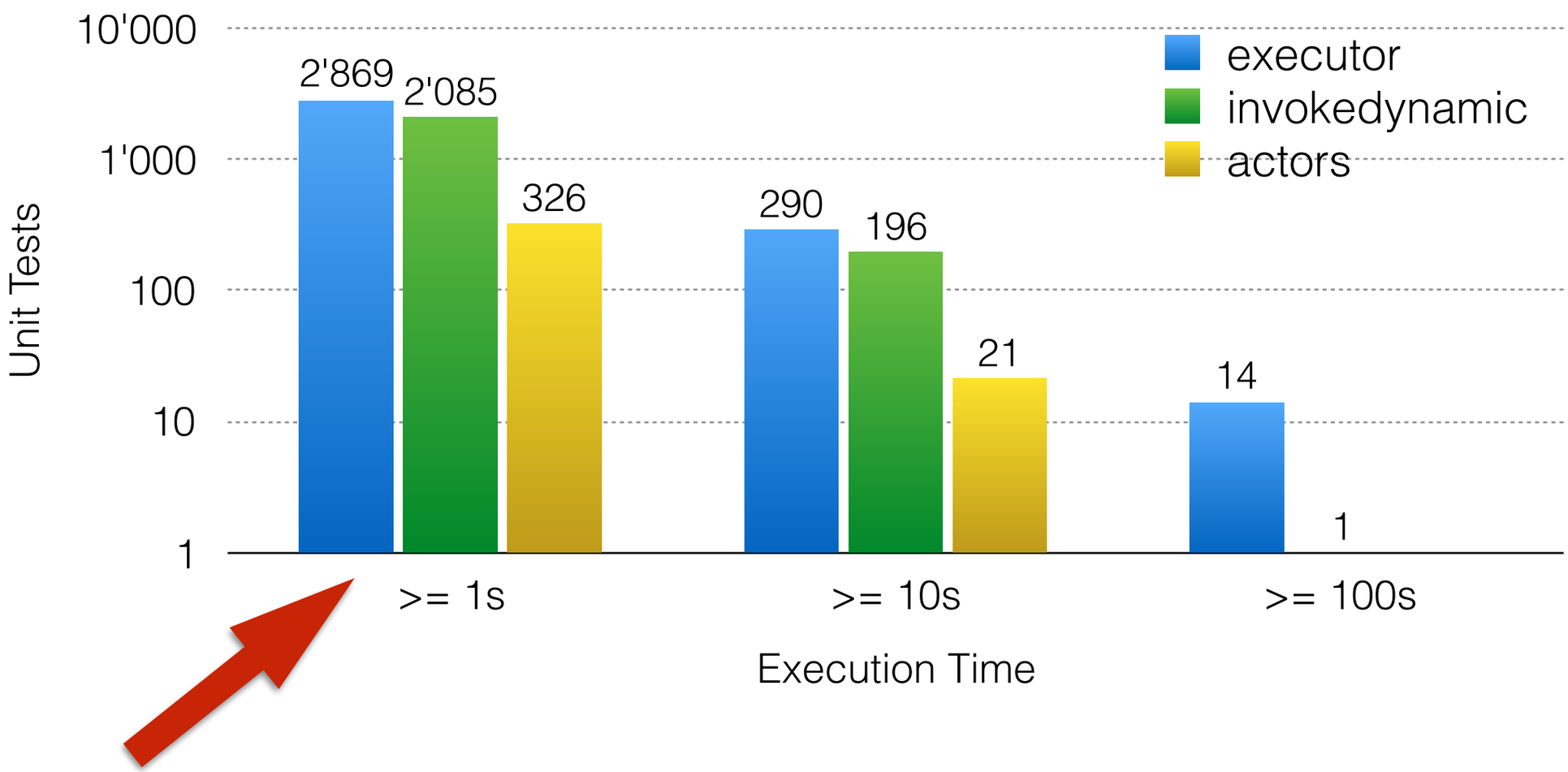
Can a fully automated process find open-source workloads that are suitable as benchmarks for specific evaluation needs?

Use cases:

- **executor**: benchmarks utilizing task execution frameworks in Java applications
- **invokedynamic**: benchmarks executing a specific Java bytecode instruction
- **actors**: benchmarks utilizing actor libraries in Java applications

Evaluation

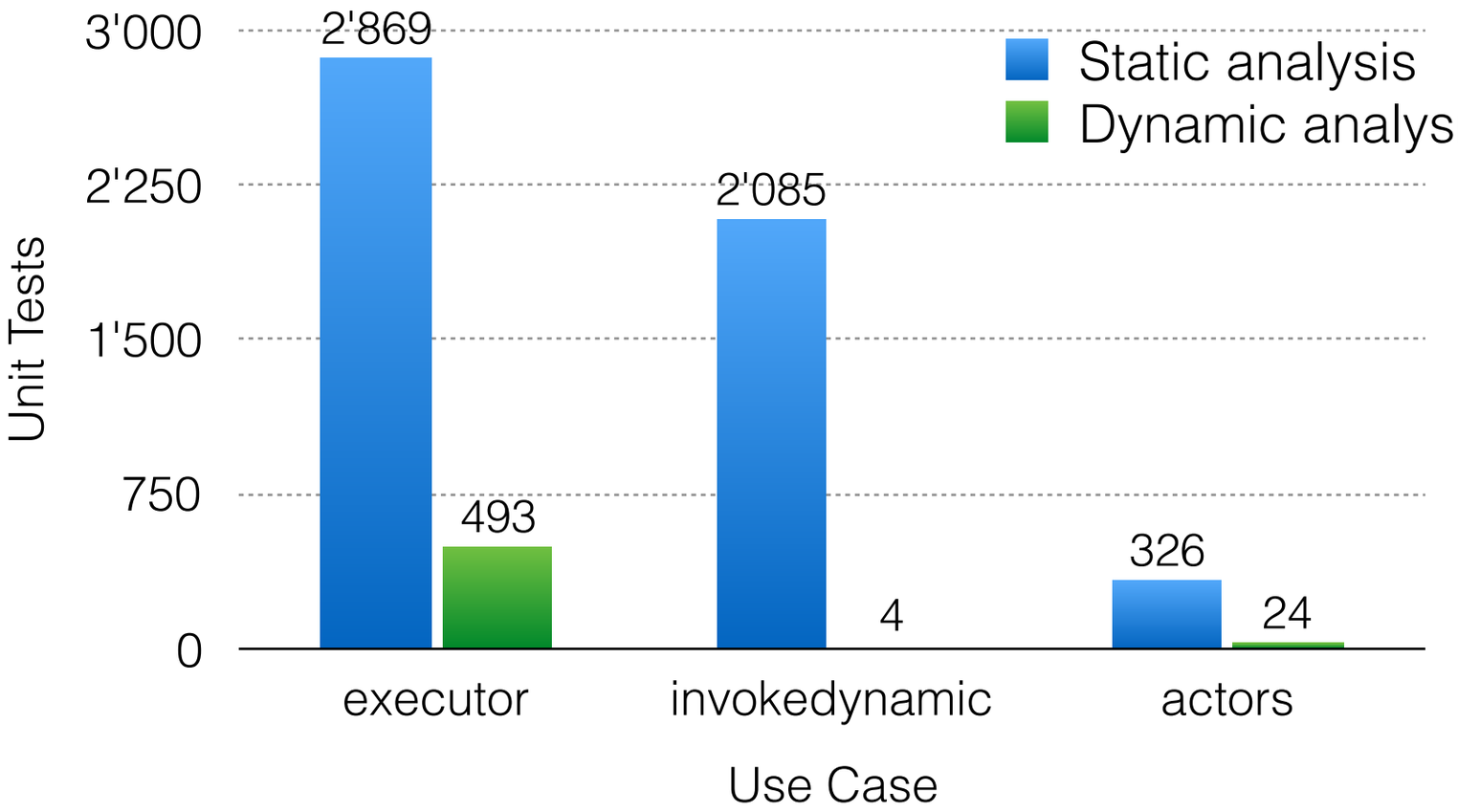
1. **Significance:** Can we expect to find tests that are long enough to serve as benchmark workload?



☑ In all use cases, a significant number of tests run for more than the time threshold

Evaluation

2. Suitability: Can dynamic analysis improve identification of workloads matching a specific evaluation context?



Stage	%
executor	17.18%
invokedynamic	0.19%
actors	7.36%

☑ Dynamic analysis is a fundamental tool for selecting only suitable workloads

Evaluation

3. Diversity: Can we find diversity in the workloads that would allow synthesizing a benchmark suite?

Executor: 493 unit tests after dynamic filtering

#	Project	Unit Test	Time [s]	Allocations	Invocations
1	prova	test/ws/prova/test2/ProvaWorkflowsTest.predicate_join	2.523	17938	469320
2	datacube	com/urbanairship/datacube/HBaseBackfillerTest.testMutationsWhileBackfilling	96.806	127961	7628592
3	jdeferred	org/jdeferred/impl/FilteredPromiseTest.testNoOpFilter	2.003	72	755
4	datacube	com/urbanairship/datacube/BackfillExampleTest.test	62.371	78728	4367224
5	cmb	com/comcast/cmb/test/unit/CassandraTest.testCassandraCounters	4.007	54	636
6	svarut	no/kommune/bergen/soa/svarut/ServiceContextTest.init	1.983	12510	205187
7	antlr4	org/antlr/v4/test/runtime/java/TestPerformance.testExpressionGrammar_1	1.212	109350	2595015
8	prova	test/ws/prova/test2/ProvaMessagingTest.ring_parallel	17.961	251119179	3144752537
9	pangool	com/datasalt/pangool/tuplemr/mapred/lib/output/TestMultipleOutputs.test	2.088	241759	12943082
10	prova	test/ws/prova/test2/ProvaFunctionalProgrammingTest.func_reactive_unfoldr_iteration_perf_large	2.690	78180455	928571592
11	graphdb	org/neo4j/backup/TestBackup.fullThenIncremental	2.159	11568	88334
12	prova	test/ws/prova/test2/ProvaMetadataTest.cep005	13.267	30393192	443299688

Evaluation

3. **Diversity:** Can we find diversity in the workloads that would allow synthesizing a benchmark suite?

☑ From 493 workload candidates to 12 representative and diverse ones

#	Project	Unit Test	Time [s]	Allocations	Invocations
1	prova	test/ws/prova/test2/ProvaWorkflowsTest.predicate_join	2.523	17938	469320
2	datacube	com/urbanairship/datacube/HBaseBackfillerTest.testMutationsWhileBackfilling	96.806	127961	7628592
3	jdeferred	org/jdeferred/impl/FilteredPromiseTest.testNoOpFilter	2.003	72	755
4	datacube	com/urbanairship/datacube/BackfillExampleTest.test	62.371	78728	4367224
5	cmb	com/comcast/cmb/test/unit/CassandraTest.testCassandraCounters	4.007	54	636
6	svarut	no/kommune/bergen/soa/svarut/ServiceContextTest.init	1.983	12510	205187
7	antlr4	org/antlr/v4/test/runtime/java/TestPerformance.testExpressionGrammar_1	1.212	109350	2595015
8	prova	test/ws/prova/test2/ProvaMessagingTest.ring_parallel	17.961	251119179	3144752537
9	pangool	com/datasalt/pangool/tuplemr/mapred/lib/output/TestMultipleOutputs.test	2.088	241759	12943082
10	prova	test/ws/prova/test2/ProvaFunctionalProgrammingTest.func_reactive_unfoldr_iteration_perf_large	2.690	78180455	928571592
11	graphdb	org/neo4j/backup/TestBackup.fullThenIncremental	2.159	11568	88334
12	prova	test/ws/prova/test2/ProvaMetadataTest.cep005	13.267	30393192	443299688

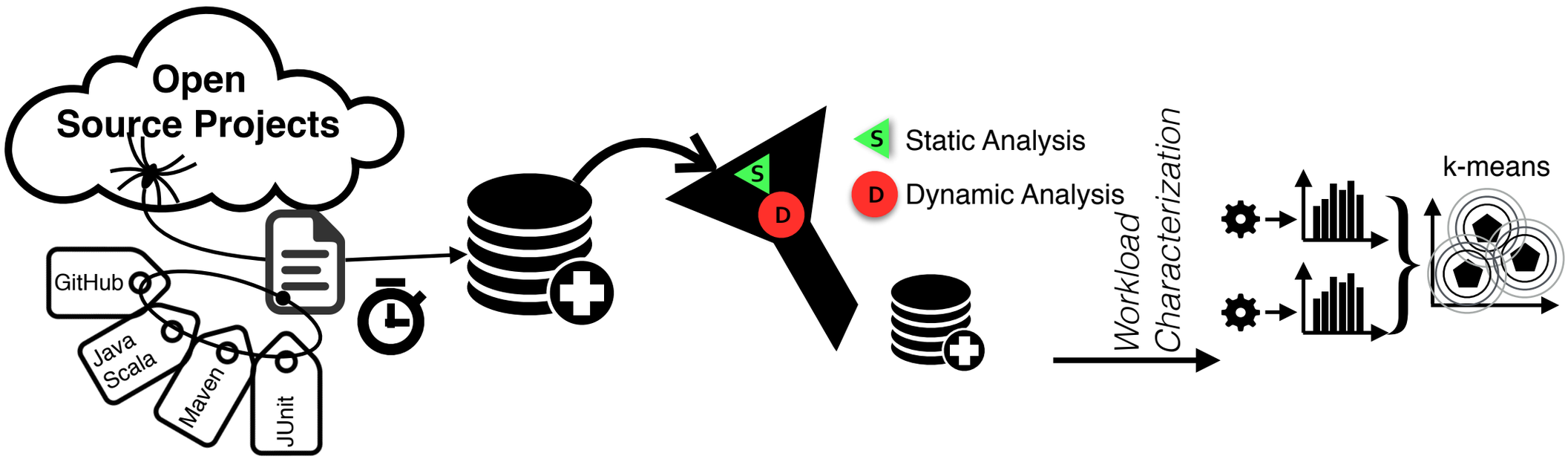
Discussion

- Investigate representativeness of unit tests wrt. real-world workloads
- Recognize and filter out "garbage" projects
- Recognize and filter out workloads with non-deterministic execution profile
- Replace execution time with other metrics to measure significance

Conclusions

- We investigated the feasibility of using **unit tests as workloads** in custom benchmarks
- **AutoBench**: a methodology and toolchain to find, filter, and classify workloads from open-source projects
- Encouraging results towards **automatic generation of benchmark suites** for specific needs.
- Future work:
  - DSL for expressing workload properties and optimization goals
  - Ranking of similar workloads instead of filtering

AutoBench



- More information in:
  - Yudi Zheng, Andrea Rosà, Luca Salucci, Yao Li, Haiyang Sun, Omar Javed, Lubomír Bulej, Lydia Y. Chen, Zhengwei Qi and Walter Binder. “AutoBench: Finding Workloads That You Need Using Pluggable Hybrid Analyses”. In IEEE SANER 2016 (ERA paper).
  - Slides: `http://inf.usi.ch/phd/rosaa/autobench_saner16.pdf`
- Contact detail:
  - Andrea Rosà
    andrea.rosa@usi.ch
    http://www.inf.usi.ch/phd/rosaa