Search-Based Fuzzing of Web APIs with EvoMaster

Prof. Andrea Arcuri

Kristiania University College, Oslo, Norway

In this Talk

- Introduction: Prof. Arcuri
- Search-Based Software Testing
- System Testing of Web APIs with **EvoMaster**
- Advance Topics
- Concluding Remarks

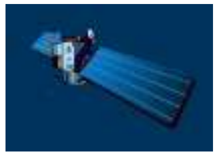
About Myself

- Prof. Andrea Arcuri
- Italian
- Work in Norway, Oslo
- Kristiania University College
- PhD in 2009 on AI applied to Software Testing, UK
- Main research interest: Search-Based Software Testing (**SBST**)
- Lead of Artificial Intelligence in Software Engineering (AISE) Lab

AISE Lab

- Currently 6 people (including PhD students and postdocs)
- Hiring another 5 by the end of the year (2022)
- Focusing on SBST topics

Search-Based Software Testing



Are software applications doing what are they supposed to do?

Software often riddled with bugs…

What to do? **Test** the software

But how to test "*properly*"?

Manual testing is expensive, tedious and of limited effect

Automated Test Case Generation

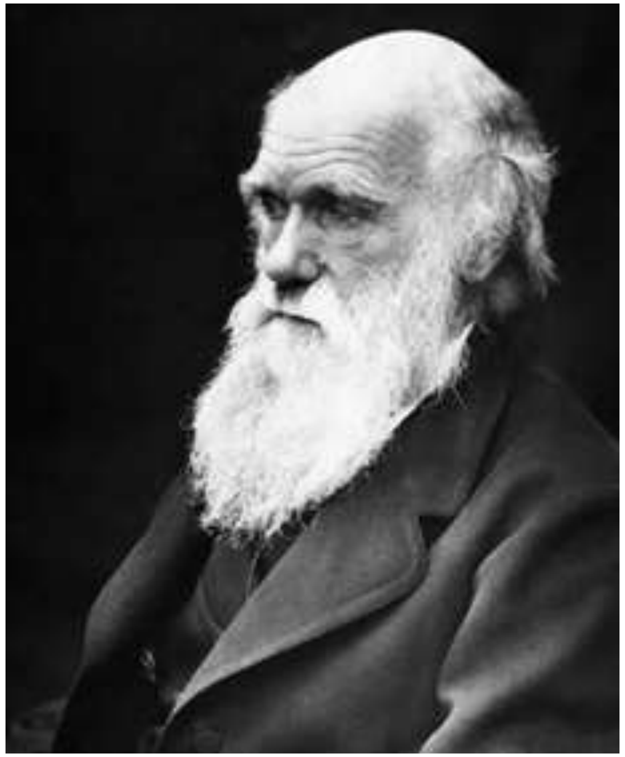
- **Automatically generate test cases**
- Model software testing as an **optimization problem**
    - Maximize code coverage
    - Find bugs
    - Etc.
- Use optimization algorithms
- Benefits: *cheaper and more effective than manual testing*
- *Hard problem to automate*
    - given a non-linear constraint, there is no guaranteed algorithm that can solve it in polynomial time

2 Uses of Generated Tests

- If automated oracles: **automatically detect faults**
- No oracles / faults: **regressing testing**
  - Tests can be added to Git, to capture current behavior of system
  - If in future introduce new bug that breaks functionality, regression tests will start to fail

Search-Based Software Testing (SBST)

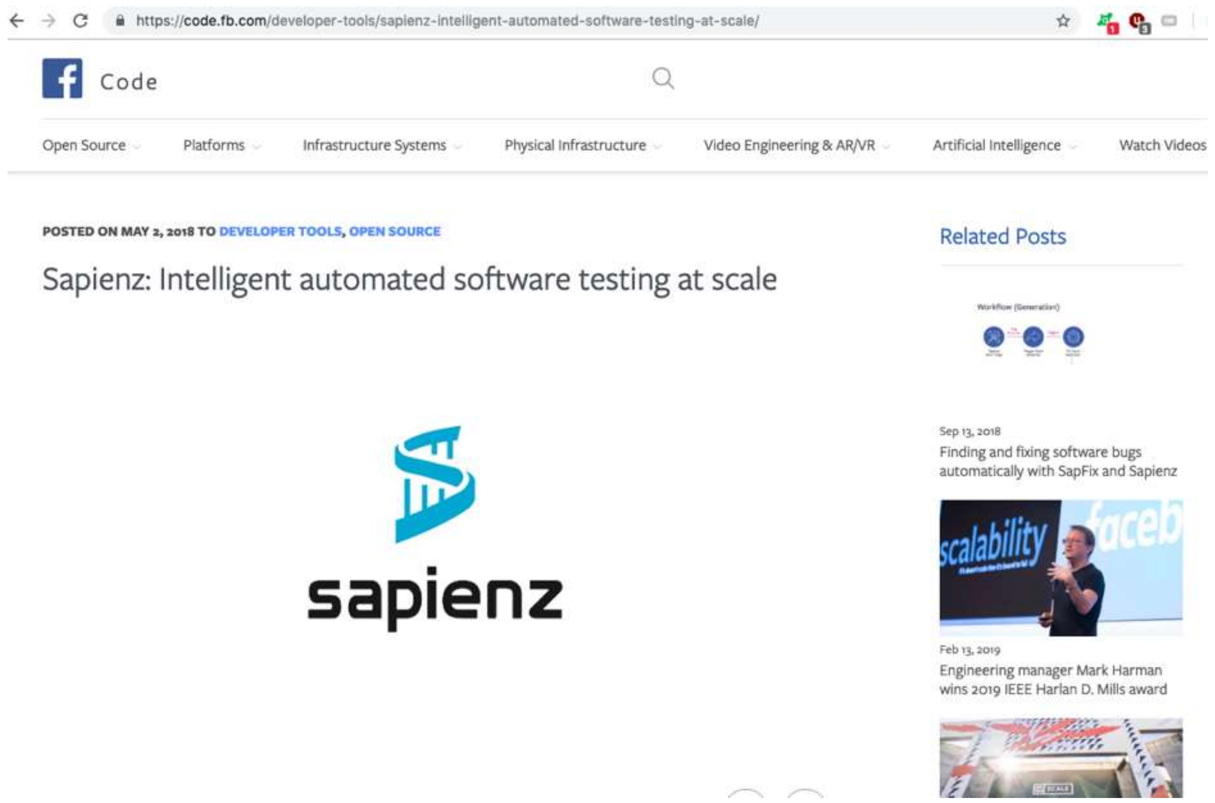
- Biology meets Software Engineering (SE)
- Casting SE problems into *Optimization Problems*
- *Genetic Algorithms*: one of most famous optimization algorithm, based on theory of evolution
- *Evolve* test cases



Success Stories: **Facebook**

Facebook uses SBST for automatically testing their software, especially their mobile apps

- eg, tools like *Sapienz* and *SapFix*



https://code.fb.com/developer-tools/sapienz-intelligent-automated-software-testing-at-scale/

Code

Open Source · Platforms · Infrastructure Systems · Physical Infrastructure · Video Engineering & AR/VR · Artificial Intelligence · Watch Videos

POSTED ON MAY 2, 2018 TO DEVELOPER TOOLS, OPEN SOURCE

Sapienz: Intelligent automated software testing at scale

sapienz

Related Posts

Sep 13, 2018
Finding and fixing software bugs automatically with SapFix and Sapienz

Feb 13, 2019
Engineering manager Mark Harman wins 2019 IEEE Harlan D. Mills award

Properties of Optimization Problems

- 2 main components: *Search Space* and *Fitness Function*
- **Goal**: find the best solution from the search space such that the fitness function is minimized/maximized

Search Space

- Set X of all possible solutions for the problem
- If a solution can be represented with 0/1 bit sequence of length N, then search space is all possible bit strings of size N
    - any data on computer can be represented with bitstrings
- Search space is usually huge, eg 2^N
    - Otherwise use brute force, and so would not be a problem

Fitness Function

- *$f(x)=h$*
- Given a solution *x* in X, calculate an heuristic *h* that specifies how good the solution is
- Problem dependent, to minimize or maximize:
  - Maximize code coverage
  - Maximize fault finding
  - Minimize test suite size
  - etc.

Optimization Algorithms

- Algorithm that explores the search space X
- Only a tiny sample of X can be evaluated
- Use fitness *f(x)* to guide the exploration to fitter areas of the search space with better solutions
- Stopping criterion: after evaluating K solutions (or K amount of time is passed), return best *x* among the evaluated solutions
- Many different kinds of optimization algorithms…
    - But as a user, still need to provide the representation and f(x)

Trivial Example

- Search space: ~4 billion values
- Only 1 value cover the *if* branch
- Covering *“OK”* at random is extremely unlikely
- Need some heuristics to driver the search

```
public String foo(int x) {
  if(x == 42)
    return “OK”;
  return “NOPE”;
}
```

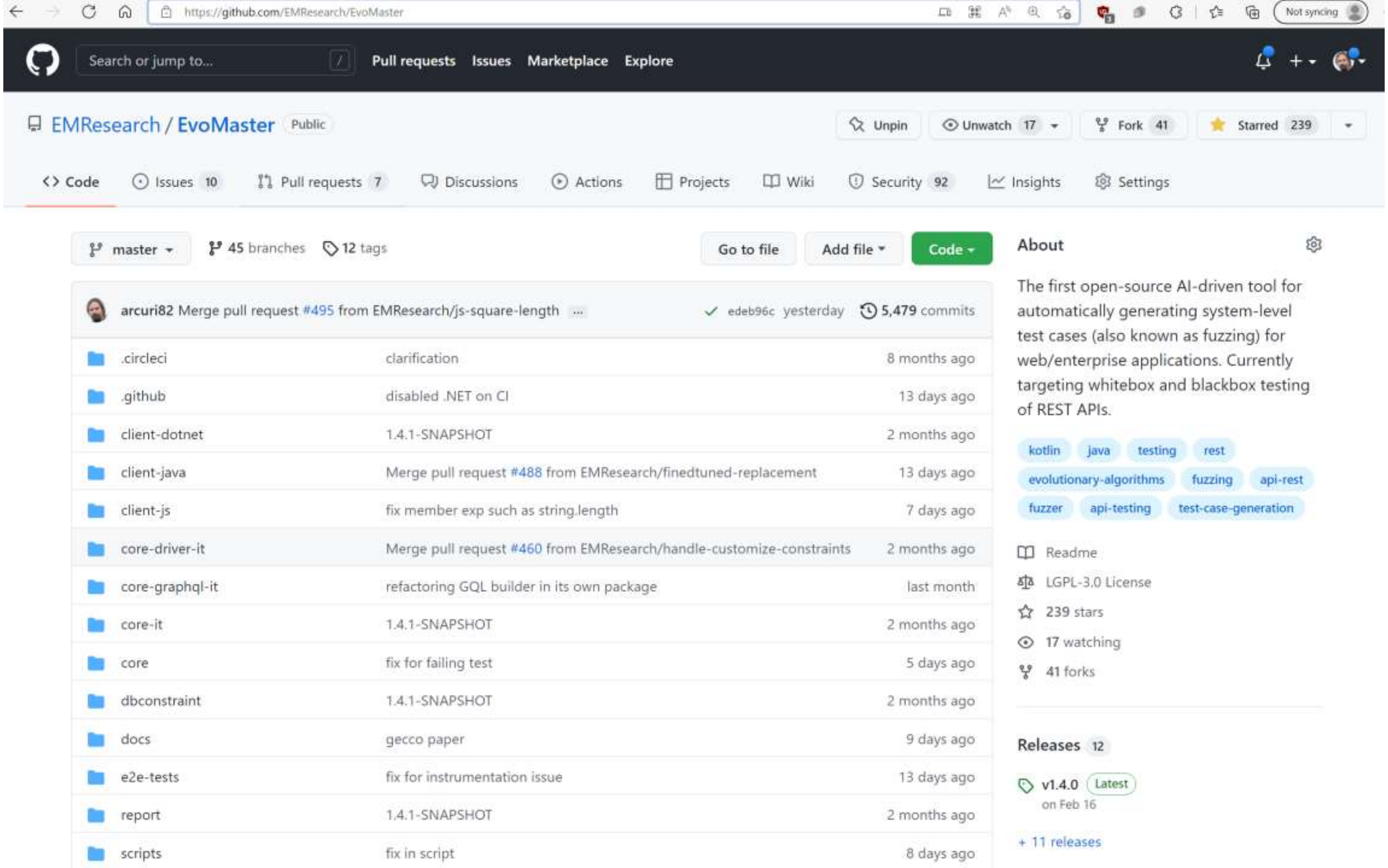
SBST Heuristics: Branch Distance

- Standard technique in the SBST literature
- Example: *if(x==42)*
- Both 5 and 900 do not solve the constraint, but 5 is *heuristically* closer
  - *d*(x==42)=|x-42|
  - *d* function to minimize
- Not just for integers, but also all other types, eg strings
- Need to *instrument* the code to calculate those branch distances
- **Trivial example, but there are many more sophisticated heuristics**

EvoMaster

EvoMaster

- SBST Tool to automatically generate *system* tests for Web APIs
- **White Box**
  - can exploit structural and runtime information of the SUT
  - currently targeting JVM languages (eg **Java** and **Kotlin**) and NodeJS (**JavaScript** and **TypeScript**)
- **Black Box**
  - can be used regardless of programming language
  - worse performance
- Search-based testing technique (**SBST**)
- **Open-source** since 2016



https://github.com/EMResearch/EvoMaster

EMResearch / EvoMaster Public

45 branches 12 tags

arcuri82 Merge pull request #495 from EMResearch/js-square-length ... ✓ edeb96c yesterday 5,479 commits

.circleci	clarification	8 months ago
.github	disabled .NET on CI	13 days ago
client-dotnet	1.4.1-SNAPSHOT	2 months ago
client-java	Merge pull request #488 from EMResearch/finedtuned-replacement	13 days ago
client-js	fix member exp such as string.length	7 days ago
core-driver-it	Merge pull request #460 from EMResearch/handle-customize-constraints	2 months ago
core-graphql-it	refactoring GQL builder in its own package	last month
core-it	1.4.1-SNAPSHOT	2 months ago
core	fix for failing test	5 days ago
dbconstraint	1.4.1-SNAPSHOT	2 months ago
docs	gecco paper	9 days ago
e2e-tests	fix for instrumentation issue	13 days ago
report	1.4.1-SNAPSHOT	2 months ago
scripts	fix in script	8 days ago

About

The first open-source AI-driven tool for automatically generating system-level test cases (also known as fuzzing) for web/enterprise applications. Currently targeting whitebox and blackbox testing of REST APIs.

kotlin java testing rest evolutionary-algorithms fuzzing api-rest fuzzer api-testing test-case-generation

Readme
LGPL-3.0 License
239 stars
17 watching
41 forks

Releases 12

v1.4.0 Latest
on Feb 16

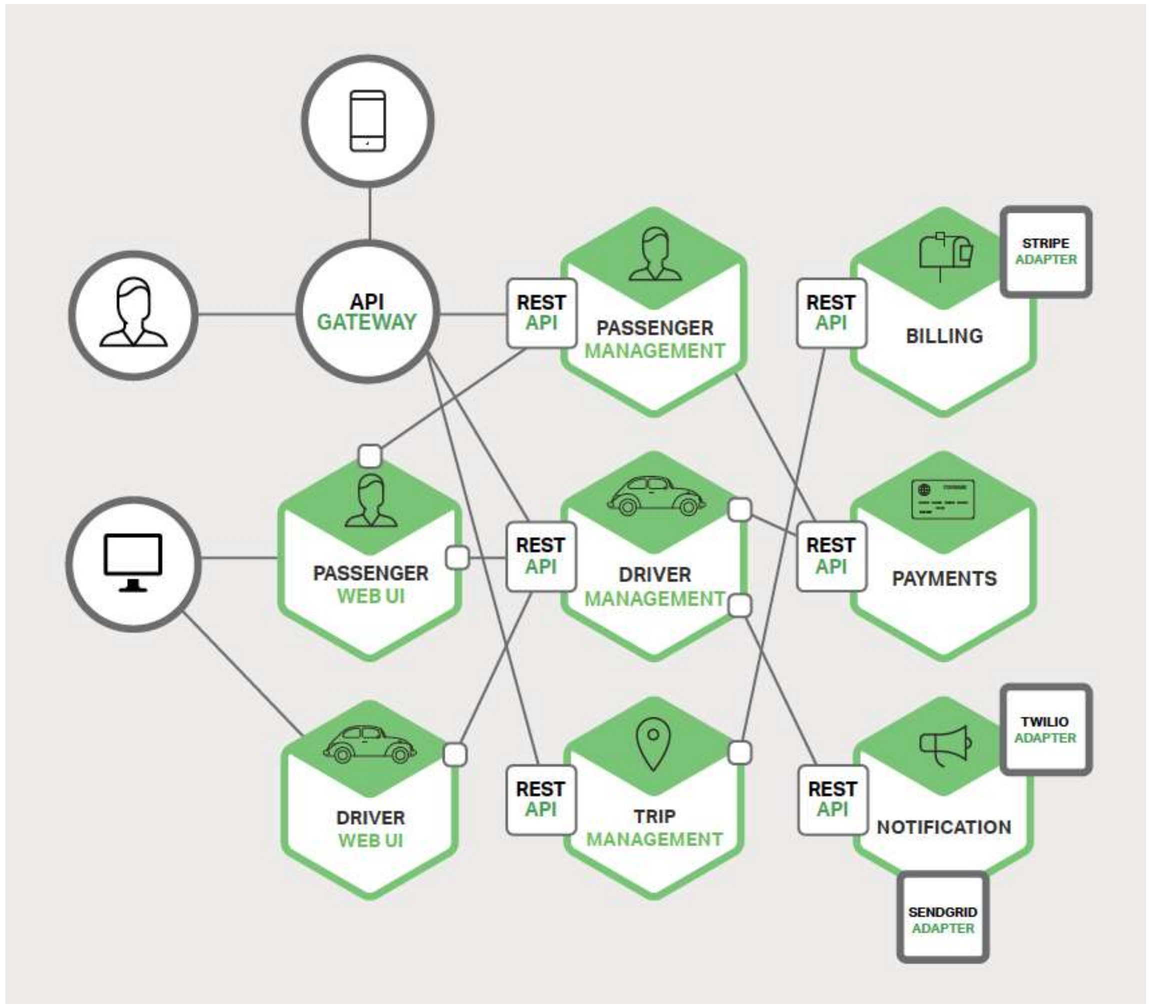
+ 11 releases

RESTful APIs

- Most common type of web services
  - others are *SOAP, GraphQL* and *RPC*
- Access of set of resources using HTTP
- REST is not a protocol, but just architectural guidelines on how to define HTTP endpoints
  - hierarchical URLs to represent resources
  - HTTP verbs (GET, POST, PUT, DELETE, etc.) as “actions” on resources

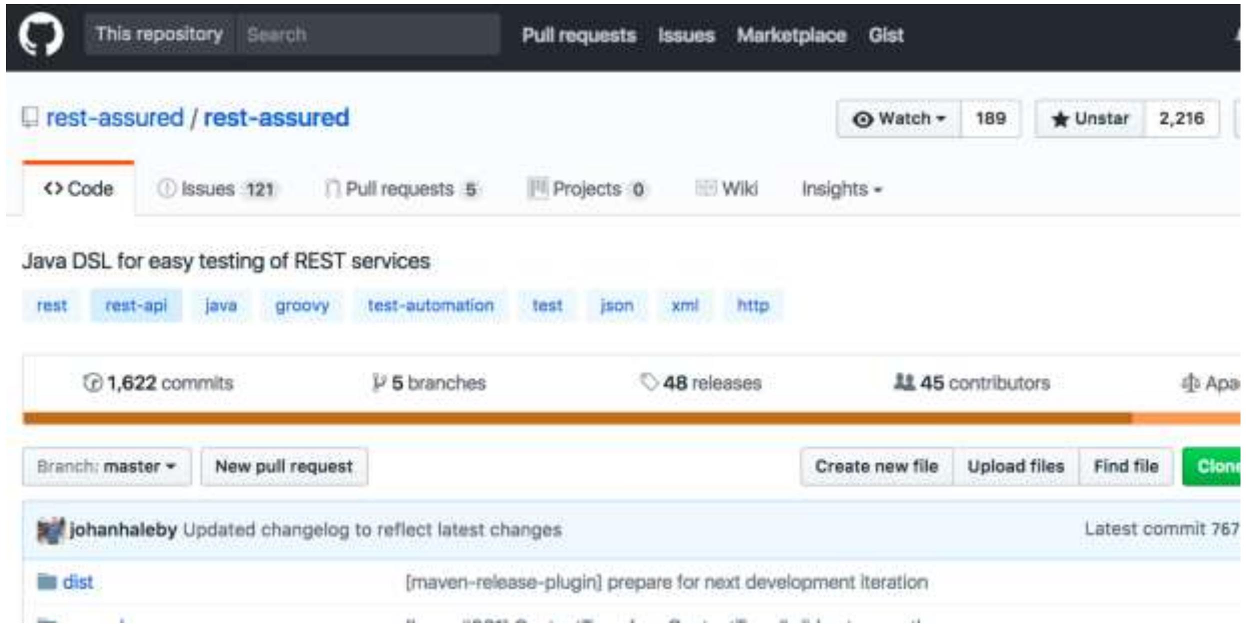
REST in Microservices

- Common trend in enterprises
- Split application in many small web services, often REST
- Easier to scale and maintain



Testing of REST APIs

- Do HTTP calls, read responses
- Setup database states
- Specialized libraries, eg in Java the popular **RestAssured**
- Specific tools like **Postman**



rest-assured / rest-assured

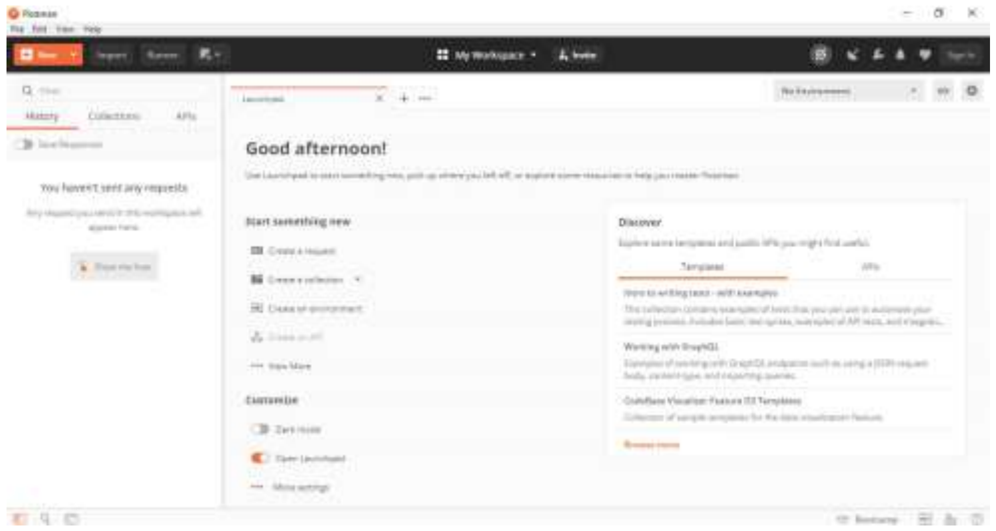
Java DSL for easy testing of REST services

rest rest-api java groovy test-automation test json xml http

1,622 commits 5 branches 48 releases 45 contributors Apa

johanhaleby Updated changelog to reflect latest changes Latest commit 767

dist [maven-release-plugin] prepare for next development iteration



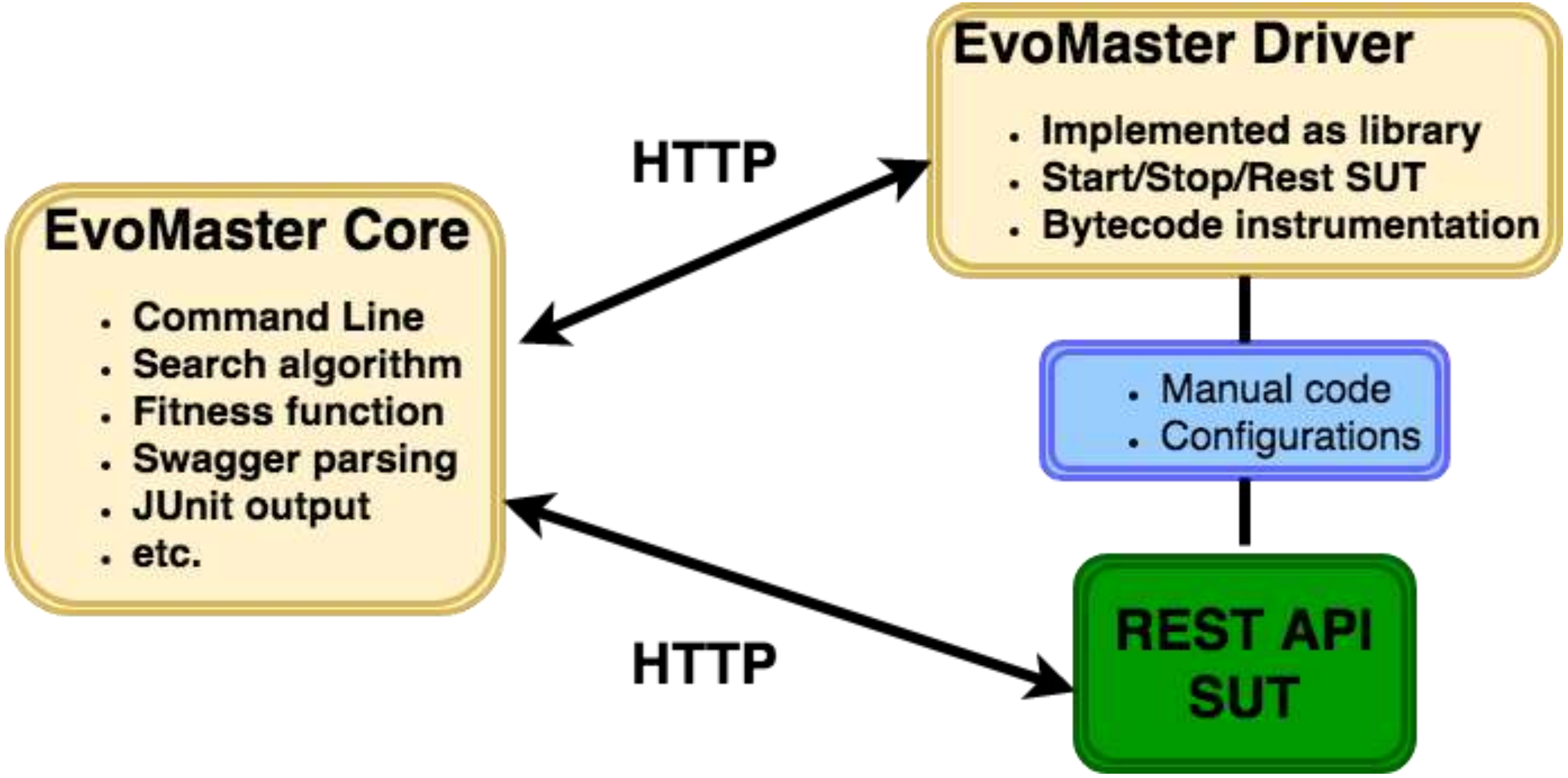
```
@Test
public void test0() throws Exception {

        given().header("Authorization", "ApiKey user")
                .accept("*/*")
                .get(”www.foo.com/api/v1/media_files/42”)
                .then()
                .statusCode(200);
}
```

REST Testing Challenges

- How to choose **query** and **path** parameters?
- How to prepare **body payloads** (e.g. JSON)?
- How to choose data to insert into **SQL** databases?
- Goals:
  - **Finding faults** (eg crashes)
  - **Maximize code coverage** (eg, regression tests)
- Writing high coverage tests *by hand* for every single endpoint is time consuming

What about **Automated Test Generation** for RESTful APIs?

- Automatically write all the test cases
- Not just execution, but choice of all the inputs
- Hard, complex problem
- Using **AI** techniques


EvoMaster Core
Command Line
Search algorithm
Fitness function
Swagger parsing
JUnit output
etc.
HTTP
EvoMaster Driver
Implemented as library
Start/Stop/Rest SUT
Bytecode instrumentation
Manual code
Configurations
HTTP
REST API SUT

OpenAPI/Swagger

- REST is not a protocol
- Need to know what endpoints are available, and their parameters
- Schema defining the APIs
- OpenAPI is the most popular one
- Defined as JSON file, or YAML
- Many REST frameworks can automatically generate OpenAPI schemas from code

EvoMaster Core

- From OpenAPI schema, defines set of endpoints that can be called
- Test case structure:
    1. setup initializing data in DB with SQL INSERTs
    2. sequence of HTTP calls toward such endpoints
- HTTP call has many components:
    - Verb (GET, POST, DELETE, etc.)
    - Headers
    - Query parameters
    - Body payload (JSON, XML, etc.)
- Evolutionary algorithm to evolve such sequences and their inputs
- Output: *self-contained* JUnit tests
- Code language of SUT is *irrelevant*, as we use HTTP to communicate with it

Fitness Function

- Needed to drive the evolution
- Reward **code coverage** and **fault detection**
- HTTP return statuses as *automated oracles*:
  - Eg 2xx if OK, 4xx are user errors, but **5xx** are server errors (often due to bugs)
- Need guidance to be able to solve constraints in code predicates
  - “*if(x == 123456 && complexPredicate(y))*”
- Unlikely to achieve high code coverage with just random inputs
  - using several different kinds of heuristics based on code analysis

Using EvoMaster

- No need to know anything about Search Algorithms nor AI in general
  - those are just internal details
  - but good to have a general idea of how this kind of tools work
- For White-Box Testing need to write a "*driver*"
  - small class to specify how to start/stop/reset the API
  - if using common frameworks like Spring, it is relatively easy
- Need to specify for *how long* to run the tool
  - The longer the better results
  - Eg, between 1 and 24 hours

Advance Topics

Dealing With SQL Databases

- Bytecode instrumentation to intercept all JDBC calls
- Find all SQL SELECT queries that return no data
  - eg due to WHERE clauses that are not satisfied
- Insert data directly into DB as part of the test case
  - Not always possible to create data with REST endpoints (eg POST/PUT)
  - using a JDBC connection
  - need to analyze DB’s schema
- *Goal*: insert data such that SELECT are not empty
- *Challenges*: WHERE clauses might have complex constraints. Need search
- *Why?* Can have impact on code execution flow

Java Example Using Spring

```
@RequestMapping(
        path = "/{x}/{y}",
        method = RequestMethod.GET,
        produces = MediaType.APPLICATION_JSON
)
public ResponseEntity get(@PathVariable("x") int x, @PathVariable("y") int y) {

    List<DbDirectIntEntity> list = repository.findByXIsAndYIs(x, y);
    if (list.isEmpty()) {
        return ResponseEntity.status(400).build();
    } else {
        return ResponseEntity.status(200).build();
    }
}
```

Generated Test

```
@Test @Timeout(60)
fun test_1() {
   val insertions = sql().insertInto("DB_DIRECT_INT_ENTITY", 14L)
         .d("ID", "-65536")
         .d("X", "-67108182")
         .d("Y", "0")
      .dtos()
   val insertionsresult = controller.execInsertionsIntoDatabase(insertions)

   given().accept("*/*")
         .get("${baseUrlOfSut}/api/db/directint/-67108182/0")
         .then()
         .statusCode(200)
         .assertThat()
         .body(isEmptyOrNullString())

}
```

- Arcuri et. *“Handling SQL Databases in Automated System Test Generation”.* TOSEM’20

Taint Analysis

- Inputs can have constraint checks
  - eg, strings matching a regex, numbers in a certain range and strings representing dates
- Constraints might be in code and NOT in the OpenAPI schema
- Can evolve inputs till satisfy constraints… eg using SBST heuristics
- … but what if inputs are not modified and used as they are? Can we do better?

Java Example Using Spring

```
@GetMapping(
    path = "/{date:\\d{4}-\\d{1,2}-\\d{1,2}}/{number}/{setting}",
    produces = MediaType.APPLICATION_JSON_VALUE)
public String getSeparated(
    @PathVariable("date") String date,
    @PathVariable("number") String number,
    @PathVariable("setting") String setting
){

  LocalDate d = LocalDate.parse(date);
  int n = Integer.parseInt(number);
  List<String> list = Arrays.asList("Foo", "Bar");

  if(d.getYear() == 2019 && n == 42 && list.contains(setting)){
    return "OK";
  }

  return "ERROR";
}
```

Solution

- Using bytecode instrumentation, check all JDK API usages
- Checking if input from HTTP is used without modification in a JDK call
- If yes, tell the search how input should be evolved
  - eg strings only representing valid dates, like for *LocalDate.parse(date)*
  - eg strings evolved always matching a particular regex
- Still need search to evolve the inputs
  - eg to handle constraints like *d.getYear() == 2019*
- Can dramatically boost the search efforts

Generated Test

```
@Test @Timeout(60)
fun test_4() {

   given().accept("application/json")
         .get("${baseUrlOfSut}/api/testability/2019-12-10/42/Bar")
         .then()
         .statusCode(200)
         .assertThat()
         .contentType("application/json")
         .body(containsString("OK"))

}
```

- Arcuri et al. *"Enhancing Search-Based Testing With Testability Transformations For Existing APIs".* TOSEM'21

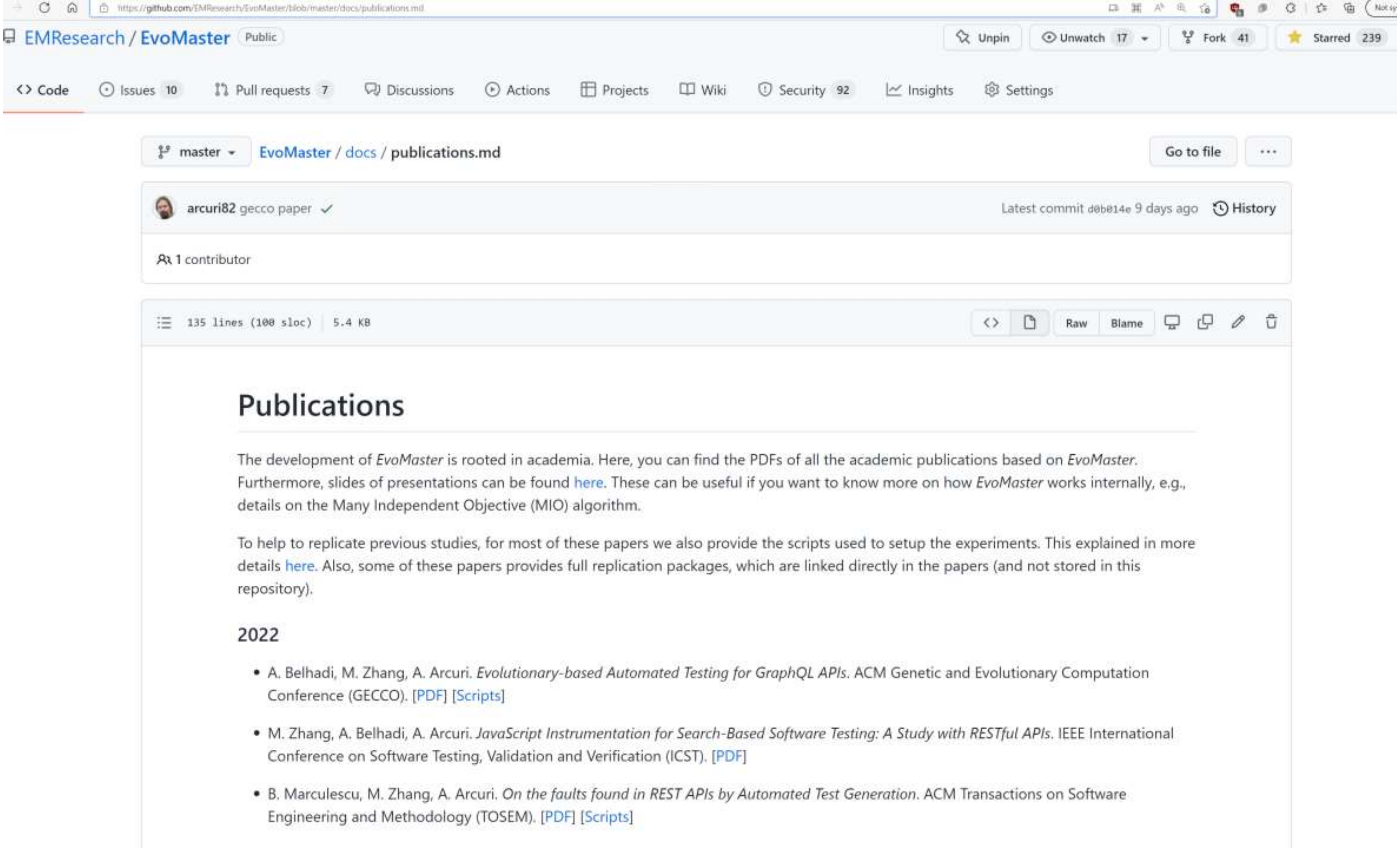
Concluding Remarks

Ongoing Work

- Not just REST: support for **GraphQL** and **RPC**
- Basic support for **C#** and **JS**
- Support for **mocking** external APIs
  - eg using WireMock
- Improve code/bytecode analysis
- Future: handling whole **Microservice Architectures**
  - ie., not just testing services in isolation
- Future: support for **Frontend Web GUIs** (eg, actions on browser)

Applications

- Found hundreds of bugs in open-source projects
  - EMB repository: https://github.com/EMResearch/EMB
- Tool comparisons: **EvoMaster** has been the **best** among existing fuzzers
  - Kim et al. *"Automated Test Generation for REST APIs: No Time to Rest Yet".* arXiv'22
  - Zhang et al. *"Open Problems in Fuzzing RESTful APIs: A Comparison of Tools".* arXiv'22
- Industrial collaborations
  - eg integration in large e-commerce companies like Meituan
    - hundreds of web services, used by hundreds of millions of customers


https://github.com/EMResearch/EvoMaster/blob/master/docs/publications.md
EMResearch / EvoMaster Public
Unpin
Unwatch 17
Fork 41
Starred 239
Code
Issues 10
Pull requests 7
Discussions
Actions
Projects
Wiki
Security 92
Insights
Settings
master
EvoMaster / docs / publications.md
Go to file
arcuri82 gecco paper
Latest commit d0b014e 9 days ago
History
1 contributor
135 lines (100 sloc) 5.4 KB
Raw
Blame

Publications

The development of *EvoMaster* is rooted in academia. Here, you can find the PDFs of all the academic publications based on *EvoMaster*. Furthermore, slides of presentations can be found here. These can be useful if you want to know more on how *EvoMaster* works internally, e.g., details on the Many Independent Objective (MIO) algorithm.

To help to replicate previous studies, for most of these papers we also provide the scripts used to setup the experiments. This explained in more details here. Also, some of these papers provides full replication packages, which are linked directly in the papers (and not stored in this repository).

2022

- A. Belhadi, M. Zhang, A. Arcuri. *Evolutionary-based Automated Testing for GraphQL APIs*. ACM Genetic and Evolutionary Computation Conference (GECCO). [PDF] [Scripts]
- M. Zhang, A. Belhadi, A. Arcuri. *JavaScript Instrumentation for Search-Based Software Testing: A Study with RESTful APIs*. IEEE International Conference on Software Testing, Validation and Verification (ICST). [PDF]
- B. Marculescu, M. Zhang, A. Arcuri. *On the faults found in REST APIs by Automated Test Generation*. ACM Transactions on Software Engineering and Methodology (TOSEM). [PDF] [Scripts]

Q/A

Thanks!