

AI-based Automated Test Case Generation for REST APIs with EvoMaster

Prof. Andrea Arcuri

Kristiania University College, Oslo, Norway

Oslo Metropolitan University, Oslo, Norway

In this talk

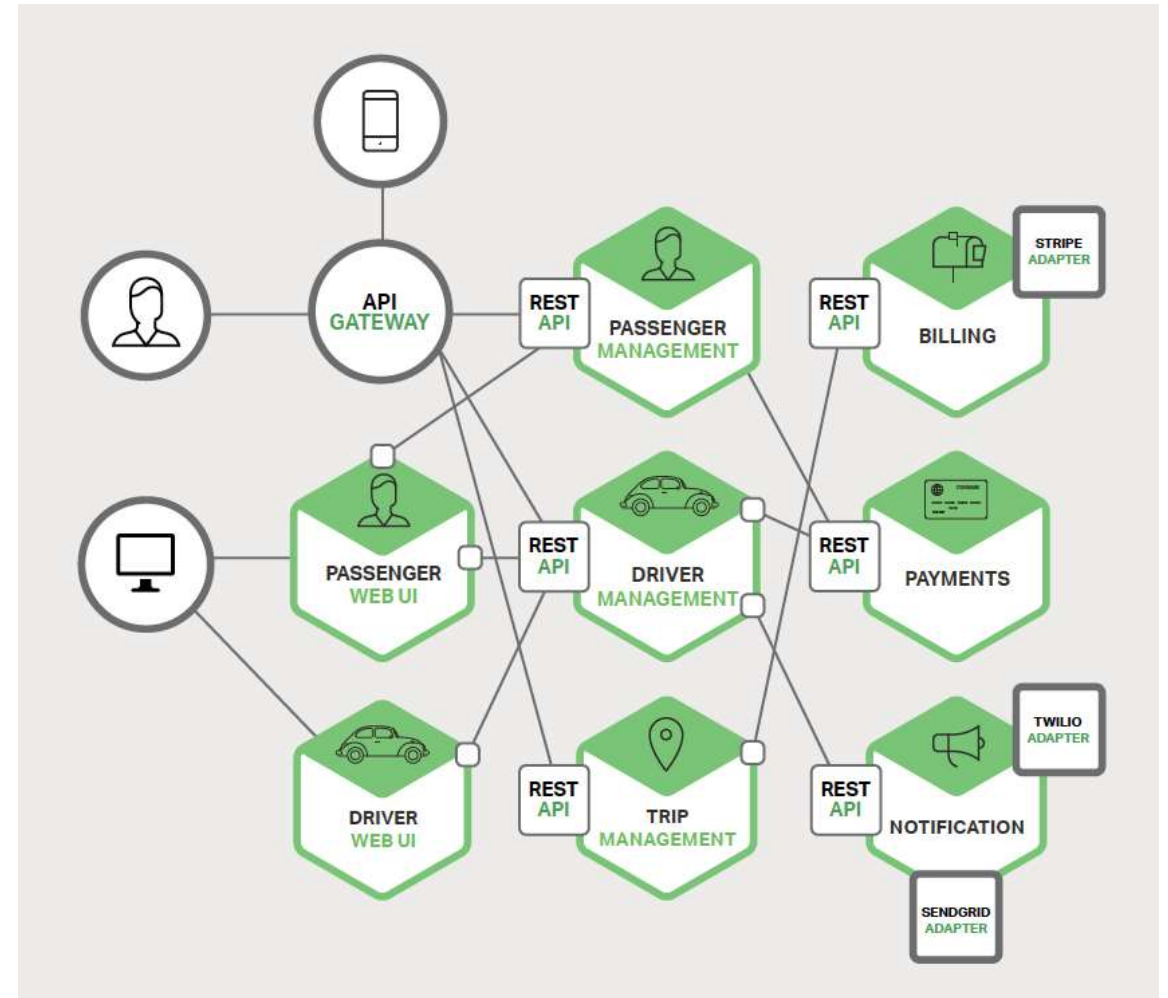
1. REST web services
2. Search Algorithms
3. *EvoMaster* tool
4. Demo

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

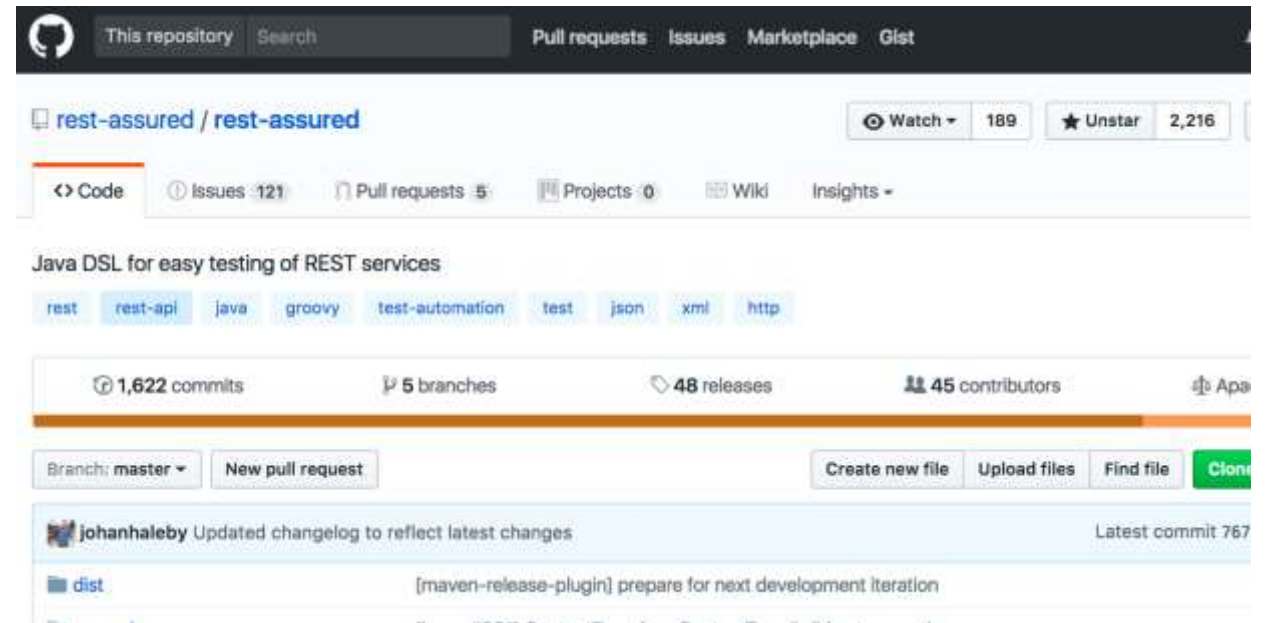
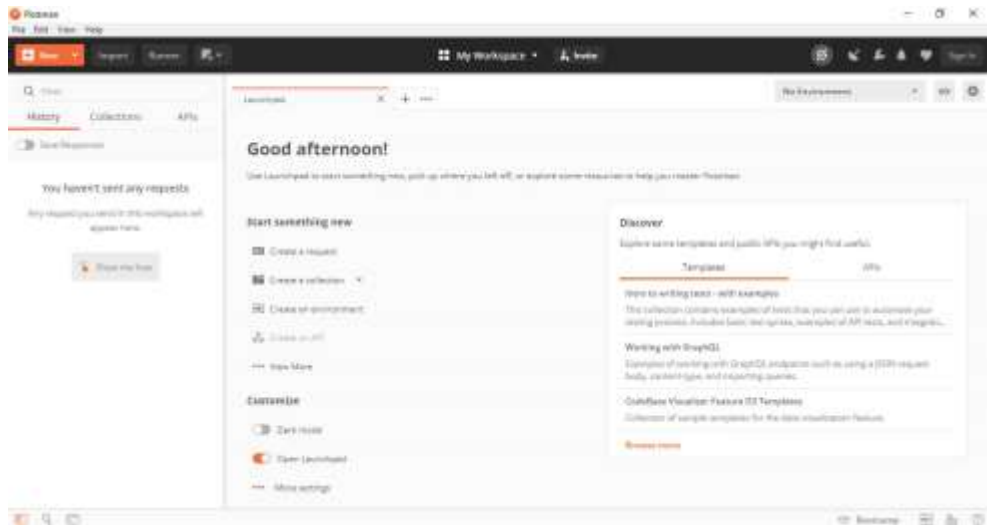
REST in Microservices

- 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**



```
@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

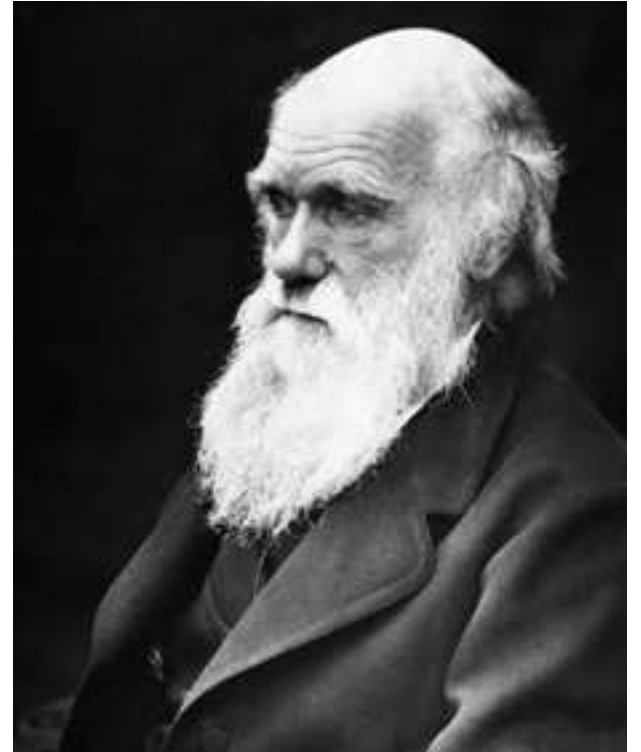
- How to choose **query** and **path** parameters?
- How to prepare **body payloads** (e.g. JSON)?
- How to choose data to insert into **SQL** databases?
- Goals:
 - **Finding faults** (eg crashes)
 - **Maximize code coverage** (eg, regression tests)
- Writing high coverage tests *by hand* for every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

Search-Based Software Testing (SBST)

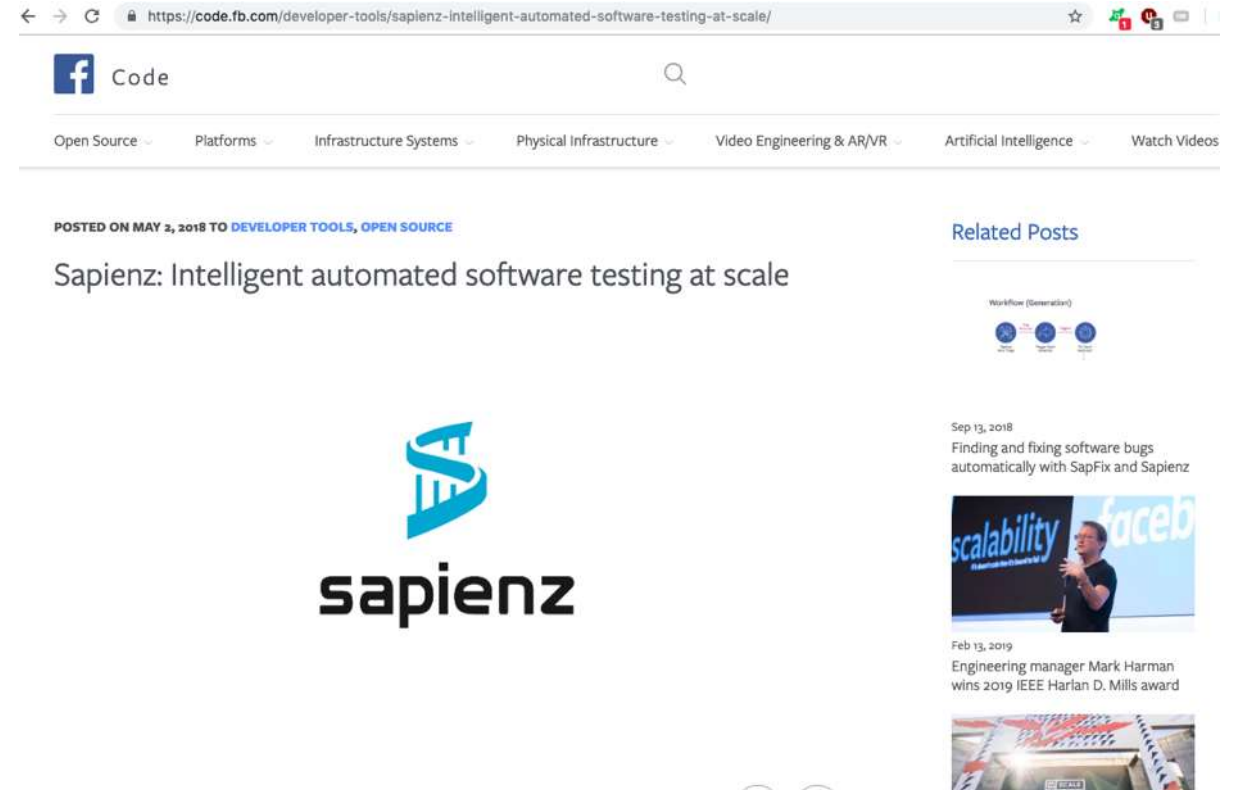
- Biology meets Software Engineering (SE)
- Casting SE problems into *Optimization Problems*
- *Genetic Algorithms*: one of most famous optimization algorithm, based on theory of evolution
- *Evolve* test cases



Success Stories: Facebook

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

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



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

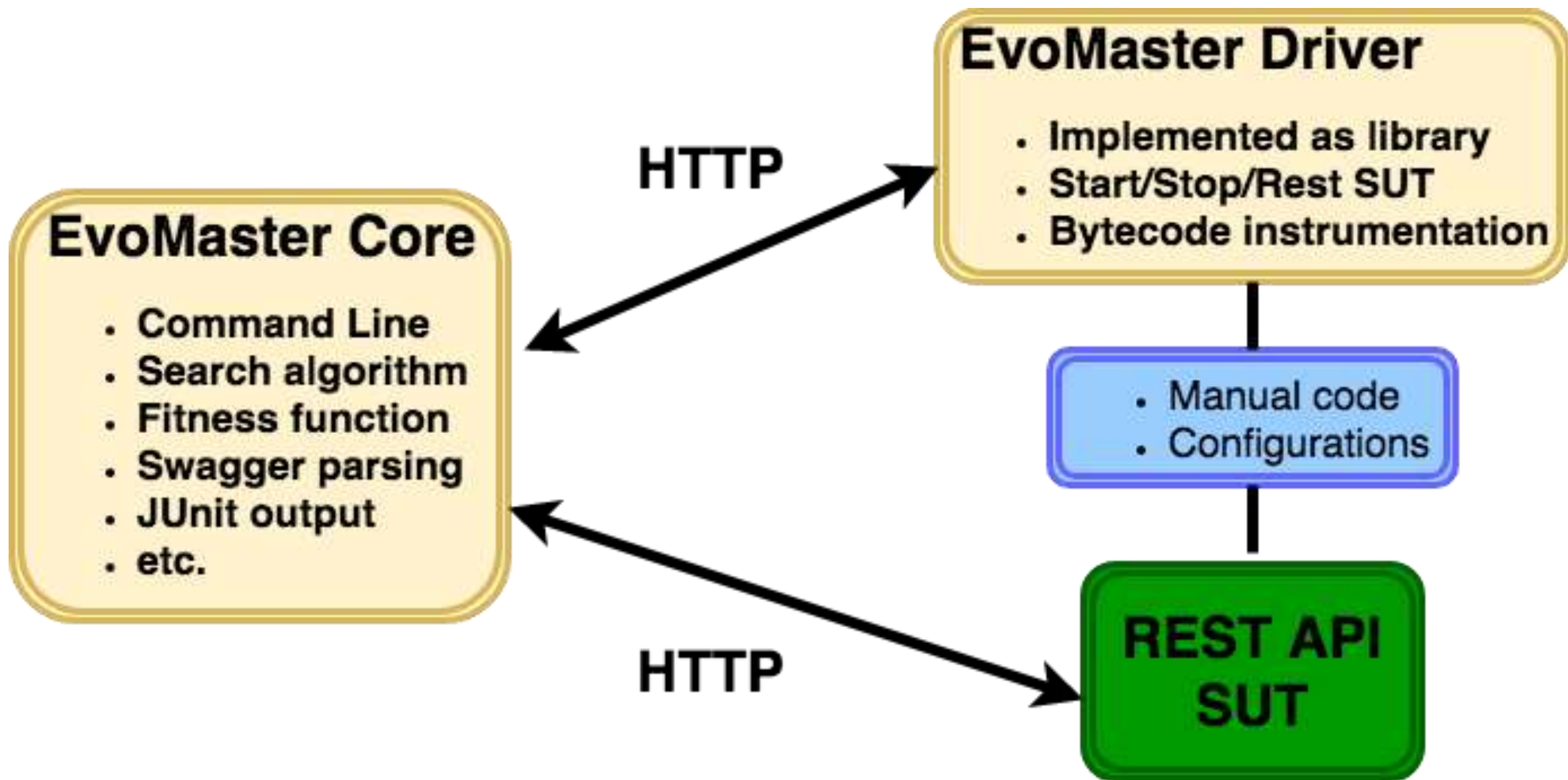
```
public String foo(int x) {  
    if(x == 42)  
        return “OK”;  
    return “NOPE”;  
}
```

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

- Tool to automatically generate tests for REST APIs
- **White Box**
 - can exploit structural and runtime information of the SUT
 - currently targeting JVM languages (eg **Java** and **Kotlin**)
- **Black Box**
 - can be used regardless of programming language
 - worse performance
- Search-based testing technique (**SBST**)
 - Evolutionary Algorithms
- Fully automated
- **Open-source** on GitHub: *www.evomaster.org*



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

www.evomaster.org

The screenshot shows the GitHub repository page for **EMResearch / EvoMaster**. The repository is public and has 18 watchers, 201 stars, and 38 forks. The main navigation bar includes links for Code, Issues (11), Pull requests (6), Discussions, Actions, Projects, Wiki, Security, Insights, and Settings. The repository is currently on the **master** branch, with 42 branches and 10 tags. The commit history shows a recent merge of pull request #402 by **arcuri82** into the **issue_396_fix** branch, with 4,326 commits. The file list includes directories like **.circleci**, **.github/workflows**, **client-dotnet/src**, **client-java**, **client-js**, **core-driver-it**, **core-graphql-it**, **core-it**, **core**, **dbconstraint**, and **docs**, each with a brief description and a timestamp. The **About** section describes EvoMaster as the first open-source AI-driven tool for automatically generating system-level test cases (also known as fuzzing) for web/enterprise applications. It currently targets whitebox and blackbox testing of REST APIs. The **Releases** section shows the latest release, **v1.2.1**, dated July 1.

Search or jump to... Pull requests Issues Marketplace Explore

EMResearch / EvoMaster Public

Unwatch 18 Unstar 201 Fork 38

Code Issues 11 Pull requests 6 Discussions Actions Projects Wiki Security Insights Settings

master 42 branches 10 tags

Go to file Add file Code

arcuri82 Merge pull request #402 from EMResearch/issue_396_fix ✓ acacea3 6 hours ago 4,326 commits

.circleci	clarification	3 months ago
.github/workflows	setting up profile for JDK 17 build	last month
client-dotnet/src	fix for name refactoring	14 days ago
client-java	update Postgres to v14	7 days ago
client-js	fixed regression in JS	8 days ago
core-driver-it	refactoring package name	3 months ago
core-graphql-it	1.2.2-SNAPSHOT	5 months ago
core-it	add missing fun of seeding tests for resource-based solution	28 days ago
core	Merge branch 'master' into issue_396_fix	yesterday
dbconstraint	Merge pull request #408 from EMResearch/update_h2	8 days ago
docs	Merge pull request #168 from agusaldaoro/add-snapshots-v2	13 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 api-rest api-testing test-case-generation

Readme

LGPL-3.0 License

Releases 10

v1.2.1 Latest on Jul 1

Ongoing Work

- Support for **C#** and **JS**
- Support for **GraphQL** and **RPC**
- Support for mocking external APIs
- Improve bytecode analysis

Demo

