

SBST Workshop at Huawei 2022

Prof. Andrea Arcuri

Kristiania University College, Oslo, Norway

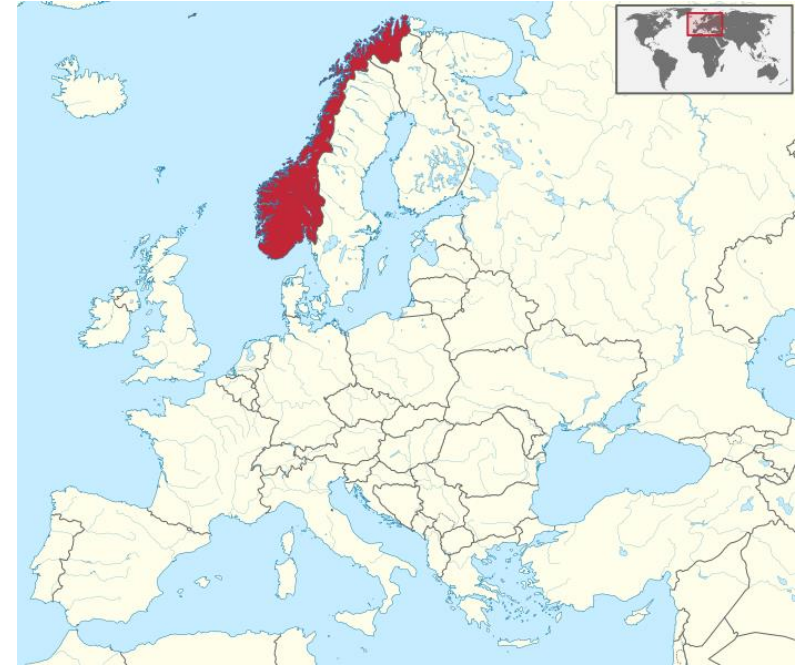
Oslo Metropolitan University, Oslo, Norway

In this Workshop

- Introduction: Prof. Arcuri
- SBST: Theory and Practice
- SBST: **Unit** Test Generation with **EvoSuite**
- SBST: **System** Testing of Web APIs with **EvoMaster**

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

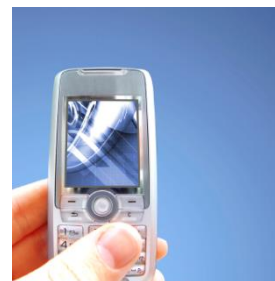
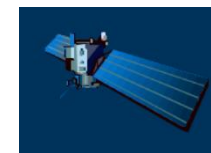


AISE Lab

- 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

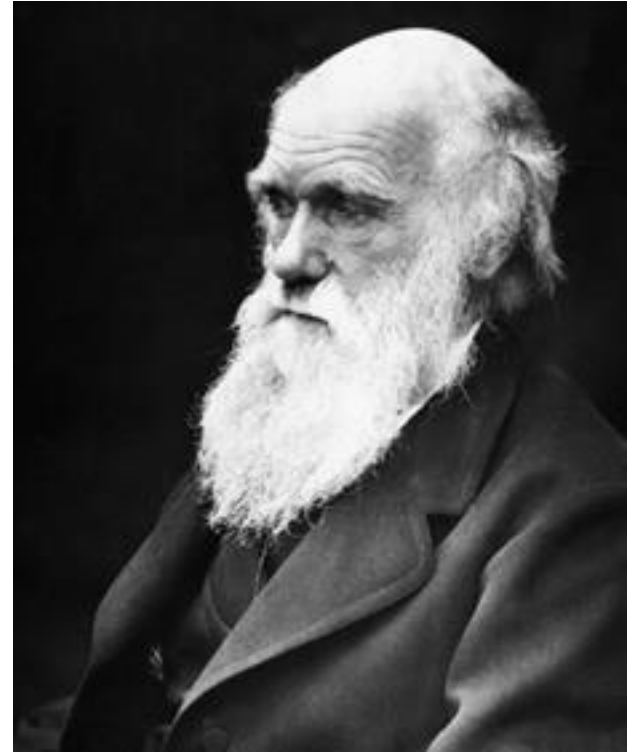
- **Automatically generate test cases**
- Model software testing as an **optimization problem**
 - Maximize code coverage
 - Find bugs
 - Etc.
- Use optimization algorithms
- Benefits: *cheaper and more effective than manual testing*
- *Hard problem to automate*
 - given a non-linear constraint, there 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)

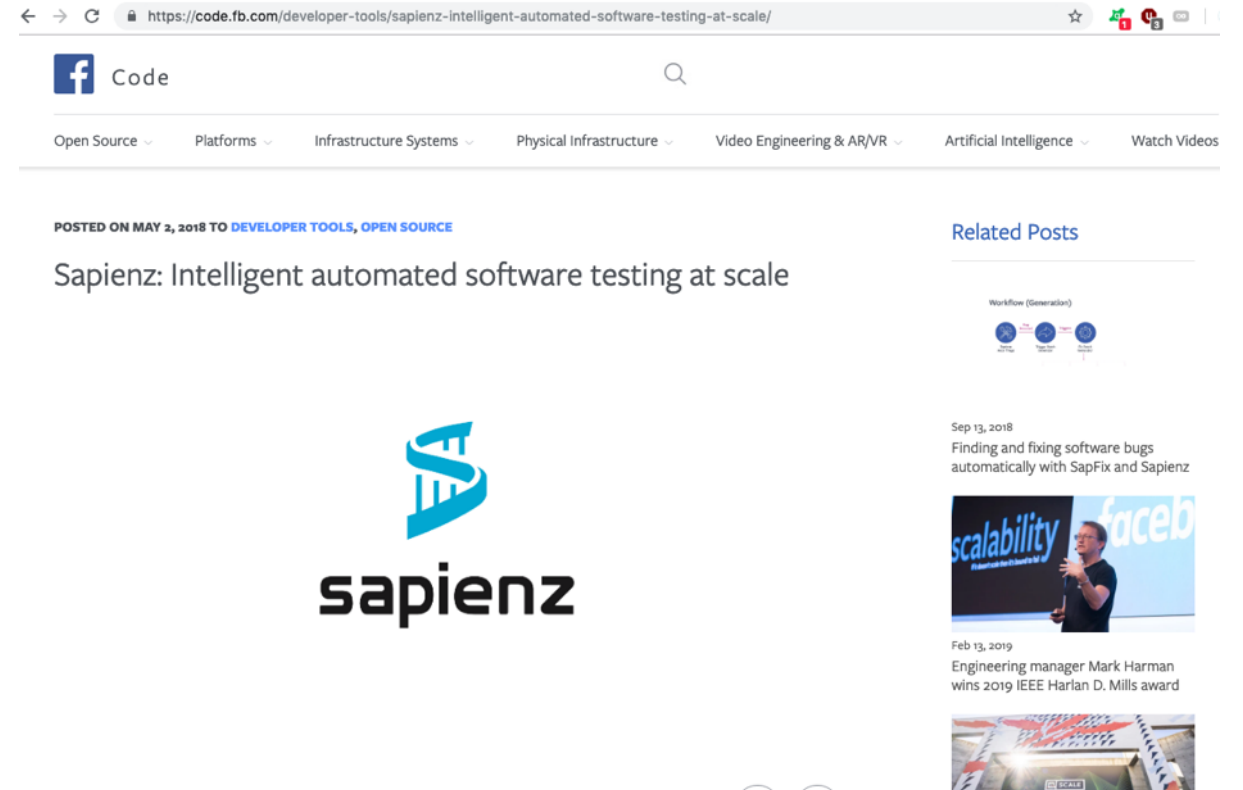
- Biology meets Software Engineering (SE)
- Casting SE problems into *Optimization Problems*
- *Genetic Algorithms*: one of most famous optimization algorithm, based on theory of evolution
- *Evolve* test cases



Success Stories: Facebook

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

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



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**

EvoSuite

EvoSuite Tool

- Targeting **Unit Testing** for **Java** programs
- SBST tool
- Objective: maximize code coverage
- Collaboration with Prof. Gordon Fraser (and many others)
- Under development since 2010
 - I have been active only till 2016
- Open-source on GitHub

Any time

Since 2022

Since 2021

Since 2018

Custom range...

Sort by relevance

Sort by date

Any type

Review articles

☐ include patents

☒ include citations

📧 Create alert

Evosuite: automatic test suite generation for object-oriented software

[PDF] acm.org

[G Fraser](#), [A Arcuri](#) - Proceedings of the 19th ACM SIGSOFT symposium ..., 2011 - dl.acm.org

... **EvoSuite**, a tool that automatically generates test cases with assertions for classes written in Java code. To achieve this, **EvoSuite** ... For the produced test suites, **EvoSuite** suggests ...

☆ Save 📄 Cite Cited by 872 Related articles All 12 versions 🔗

A large-scale evaluation of automated unit test generation using **evosuite**

[PDF] acm.org

[G Fraser](#), [A Arcuri](#) - ACM Transactions on Software Engineering and ..., 2014 - dl.acm.org

... of **EVOSUITE** in the scope of the experiments described in this article is that **EVOSUITE** only ... conveniently through **EVOSUITE**'s command line interface, and **EVOSUITE** automatically ...

☆ Save 📄 Cite Cited by 170 Related articles All 7 versions 🔗

Evosuite: On the challenges of test case generation in the real world

[PDF] ieee.org

[G Fraser](#), [A Arcuri](#) - 2013 IEEE sixth international conference on ..., 2013 - ieeeexplore.ieee.org

... a major engineering effort to simply make **EVOSUITE** run on these projects, which we call ... we face with **EVOSUITE**. In this paper, we describe the challenges **EVOSUITE** had to take in ...

☆ Save 📄 Cite Cited by 62 Related articles All 9 versions 🔗

Evosuite at the sbst 2016 tool competition

[PDF] acm.org

[G Fraser](#), [A Arcuri](#) - Proceedings of the 9th International Workshop on ..., 2016 - dl.acm.org

... This paper summarizes the results and experiences of **EvoSuite**'s participation at the fourth unit testing competition at SBST 2016, where **EvoSuite** achieved the highest overall ...

☆ Save 📄 Cite Cited by 74 Related articles All 18 versions 🔗

1600 faults in 100 projects: automatically finding faults while achieving high coverage with **evosuite**

[PDF] whiterose.ac.uk

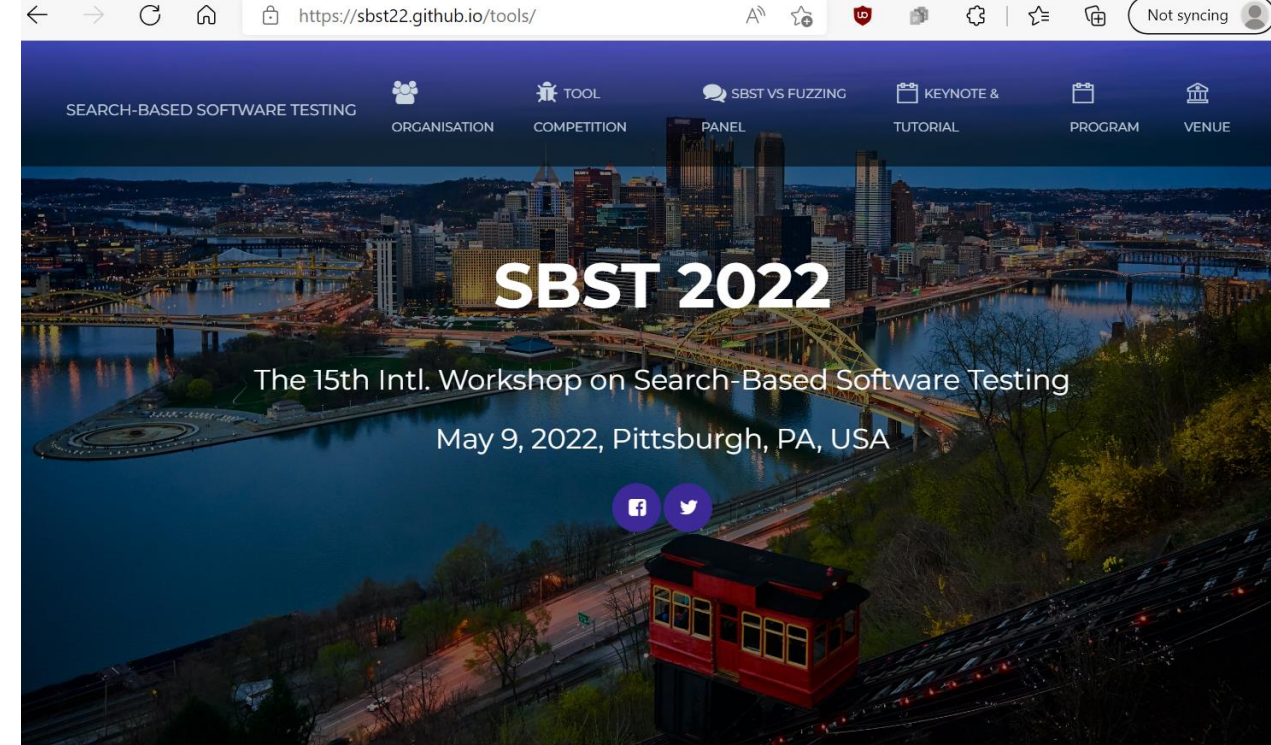
[G Fraser](#), [A Arcuri](#) - Empirical software engineering, 2015 - Springer

... , implemented in the **EvoSuite** tool. An empirical study applying **EvoSuite** on 100 randomly ... In our study, **EvoSuite** detected twice as many failures in terms of undeclared exceptions as ...

☆ Save 📄 Cite Cited by 110 Related articles All 10 versions 🔗

Tool Competition

- Each year at IEEE Workshop on Search-Based Software Testing
- Since 2013
- Different tools competed on same set of Java classes
 - E.g., **UtBot** from Huawei in 2021
- Selection of Java classes unknown to the competition participants
- **EvoSuite** won all editions but 1



Tool Competition

This year as well we are pleased to announce the tenth edition of the testing tool competition. The competition has the goal to experiment with testing tools for a diversified set of traditional and emerging systems and domains.

Java tool competition: As for recent years, we invite researchers to participate in the competition with their unit test generation tool for **Java**. Tools will be evaluated against a benchmark with respect to code coverage and mutation score.

Class Under Test (CUT)

```
class Triangle {
    int a, b, c; //sides
    String type = "NOT_TRIANGLE";

    Triangle (int a, int b, int c){...}

    void computeTriangleType() {
        1. if (a == b) {
        2.     if (b == c)
        3.         type = "EQUILATERAL";
        4.     else
        5.         type = "ISOSCELES";
        6.     } else {
        7.         if (a == c) {
        8.             type = "ISOSCELES";
        9.         } else {
        10.            type = "SCALENE";
        11.        }
        12.    }
    }
}
```

Test Case

```
#Test
public void test(){
    // Constructor (init)
    // Method Calls
    // Assertions (check)
}

#Test
public void test(){
    Triangle t = new Triangle (1,2,3);
    t.computeTriangleType();
    String type = t.getType();
    assertTrue(type.equals("SCALENE"));
}
```


Strengths of Automated Unit Test Generation

- Easy to apply
 - Eg, IntelliJ/Eclipse plugins, just right-click on a class
- Can achieve high code coverage
 - not uncommon > 70%

Drawbacks of Automated Unit Test Generation

- Can generate huge amount of tests
 - even when test suites minimized for code coverage
- Can struggle on enterprise software
 - dependency injection (Spring/JEE), databases, GUIs, etc
- No clear automated oracles
 - do the generated tests find faults?

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

🔗 master 45 branches 12 tags

Go to file Add file Code

About ⚙️

 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/finetuned-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

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

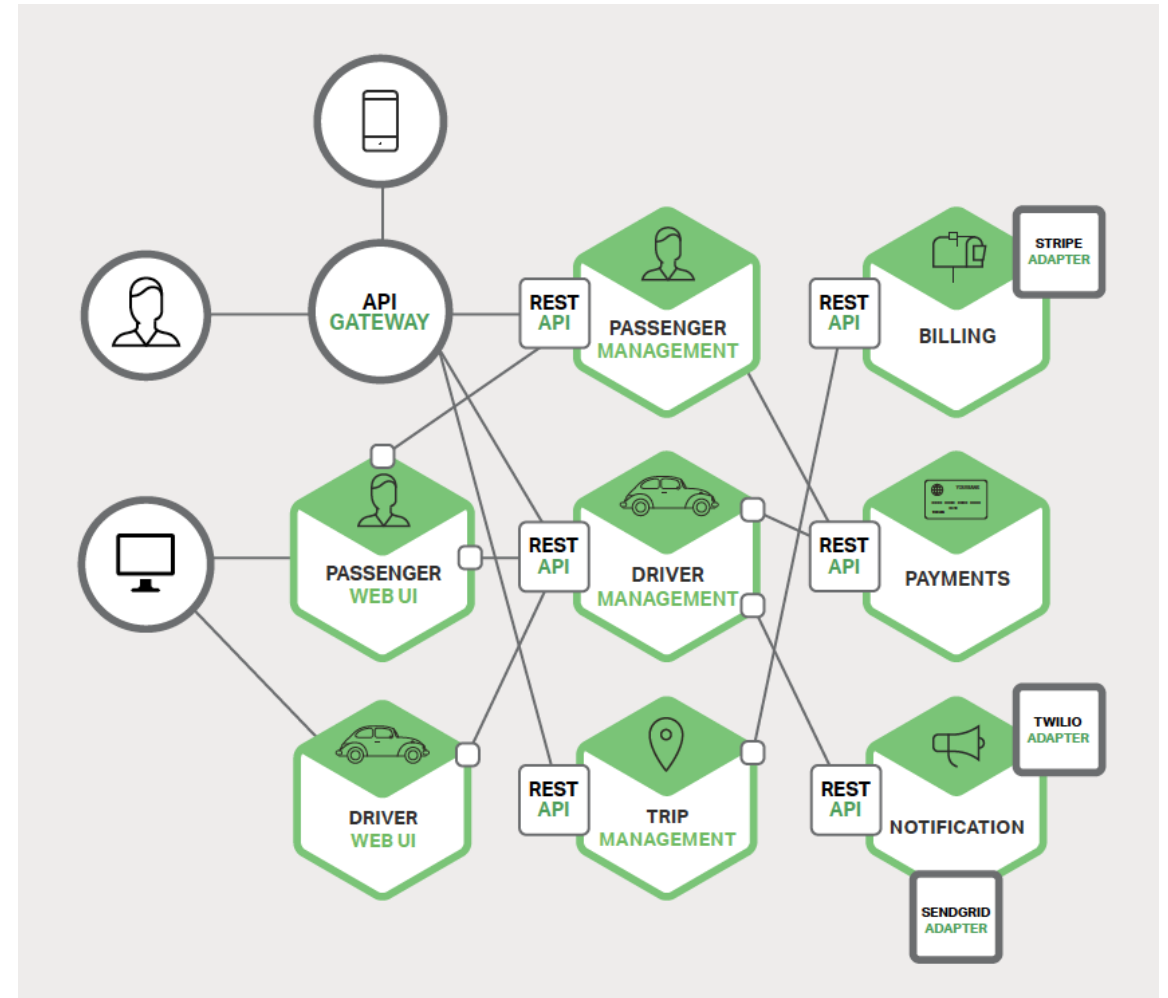
+ 11 releases

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

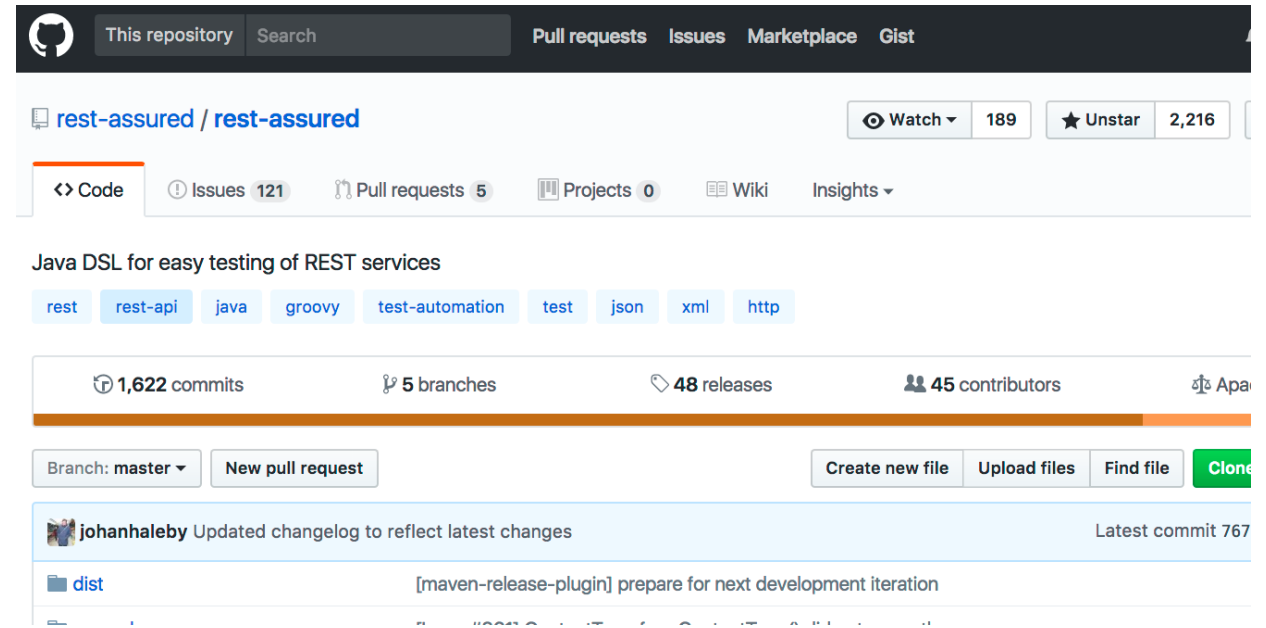
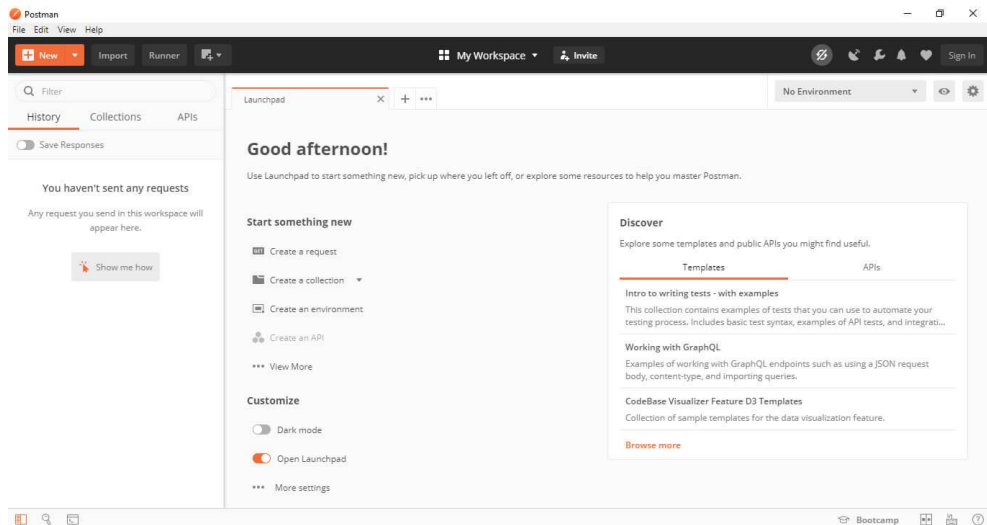
REST in Microservices

- 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**



@Test

public void test0() throws Exception {

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

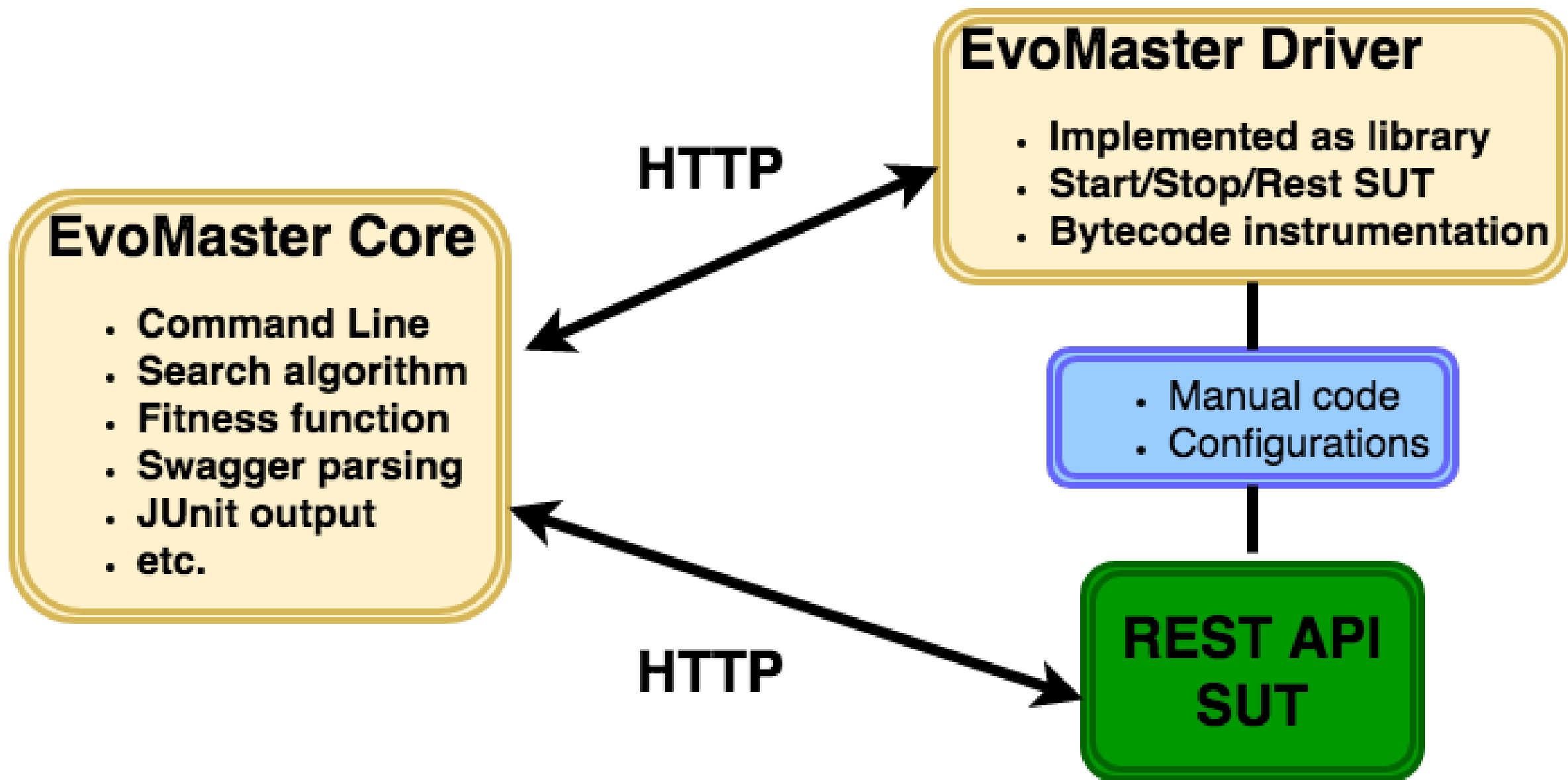
}

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



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

Ongoing Work

- Support for **C#** and **JS**
- Support for **GraphQL** and **RPC**
- Support for mocking external APIs
- Improve code/bytecode analysis
- Future: support for **Frontend Web GUIs** (eg, actions on browser)

Applications

- Found hundreds of bugs in open-source projects
- Tool comparisons: **EvoMaster** has been the best among existing fuzzers
- Industrial collaborations
 - eg integration in large e-commerce companies like Meituan
 - hundreds of web services, used by hundreds of millions of customers

👤 1 contributor

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!