Automated Test Case Generation With AI

Prof. Andrea Arcuri

Kristiania University College, Oslo, Norway

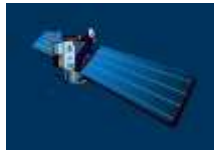
Oslo Metropolitan University, Oslo, Norway

In this talk

1. Software Testing
2. REST APIs
3. Search Algorithms
4. *EvoMaster* tool
5. Demo

Software Testing

12:00



Are software applications doing what are they supposed to do?

Ariane 5 – ESA

On June 4, 1996, the flight of the Ariane 5 launcher ended in a failure.

$500 millions in cost

Software bug

Fatal Therac-25 Radiation

1986, Texas, person died

Power Shutdown in 2003

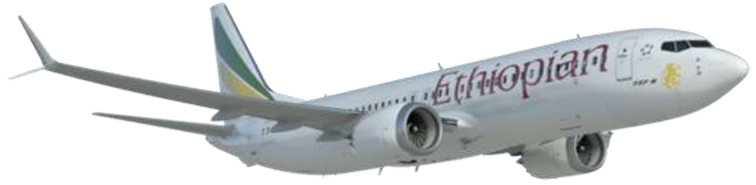
Nearly 50 millions persons affected in Canada/US

2010, Toyota, bug in braking system, 200 000 cars recalled

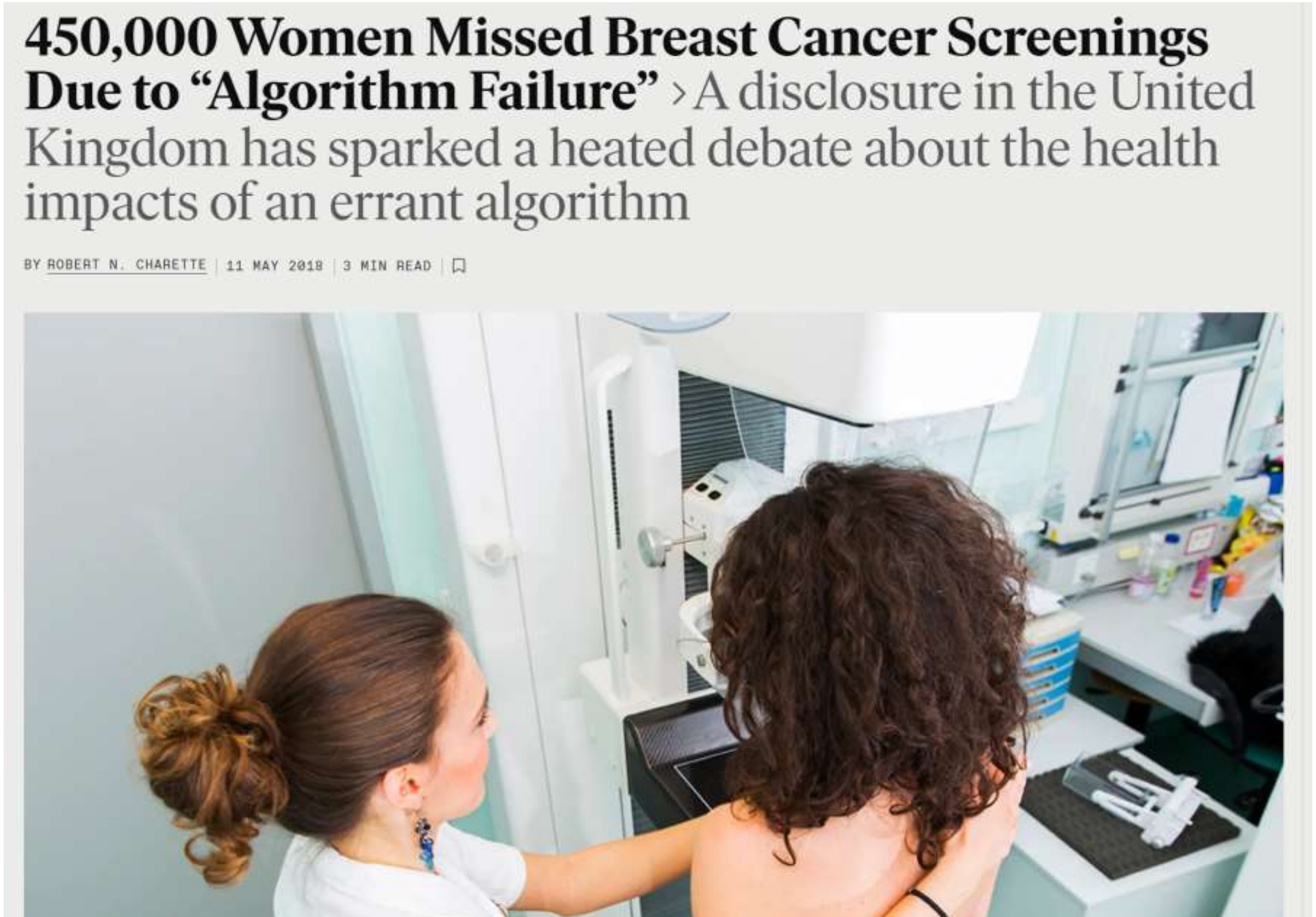
Knight Capital Group 2012

$460 millions lost in 45 minutes of trading due to bug

March 2019, Boeing 737 Max **crashed** due to **software problems**; all **157 people on board died**.



2009-2018: Estimated 135-270 deaths in UK



450,000 Women Missed Breast Cancer Screenings Due to "Algorithm Failure" › A disclosure in the United Kingdom has sparked a heated debate about the health impacts of an errant algorithm

BY ROBERT N. CHARETTE | 11 MAY 2018 | 3 MIN READ

And I could go on the whole day…

- As of 2013, estimated that software testing costing **$312 billions** worldwide
- In 2016, 548 recorded and documented software failures impacted **4.4 billion** people and **$1.1 trillion** in assets worldwide

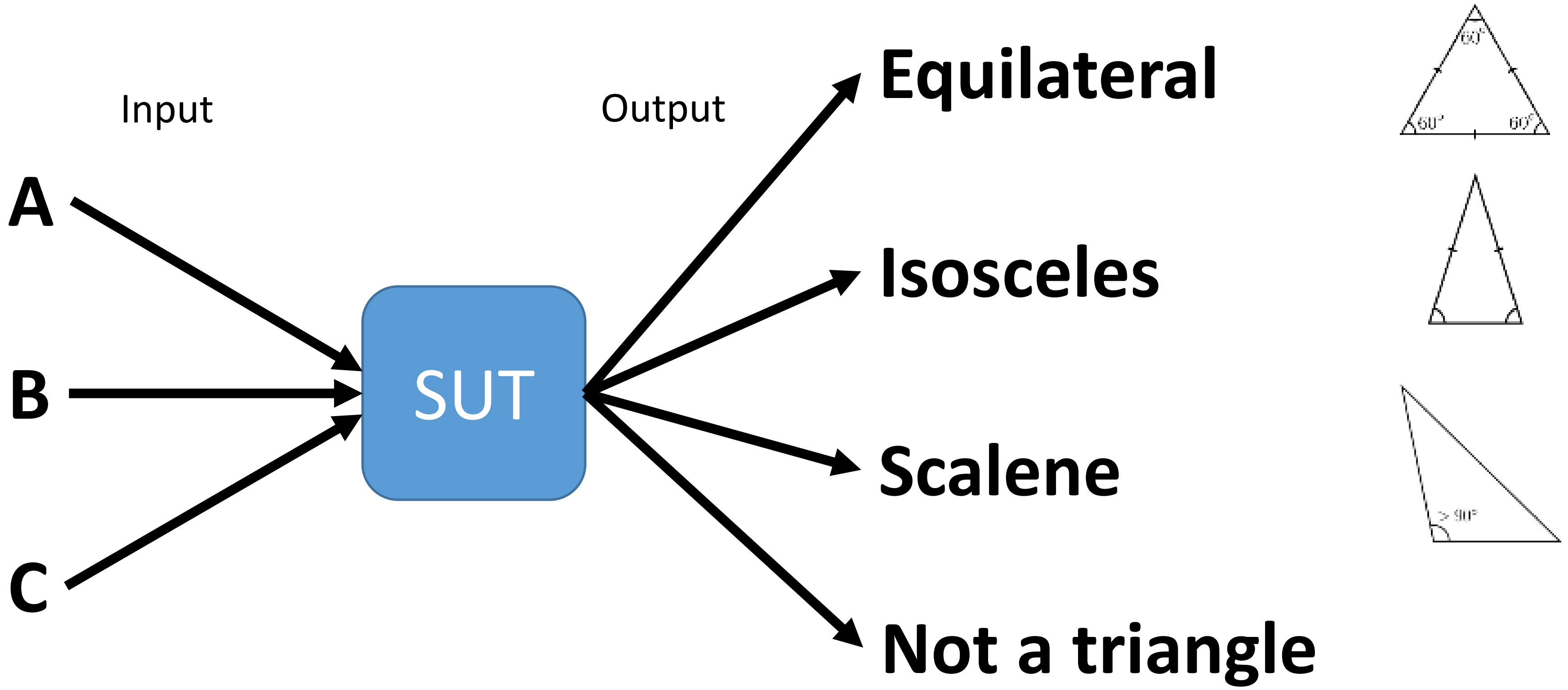
But what about every-day life in Oslo???

What to do? **Test** the software

But how to test "*properly*"?

Manual testing is expensive, tedious and of limited effect

Example: Triangle Classification (TC)



- 3 integer numbers (A, B and C) as input representing the length of the *edges*
- 4 possible outcomes
- Does the *system under test* (SUT) give the right answer?

How to test TC?

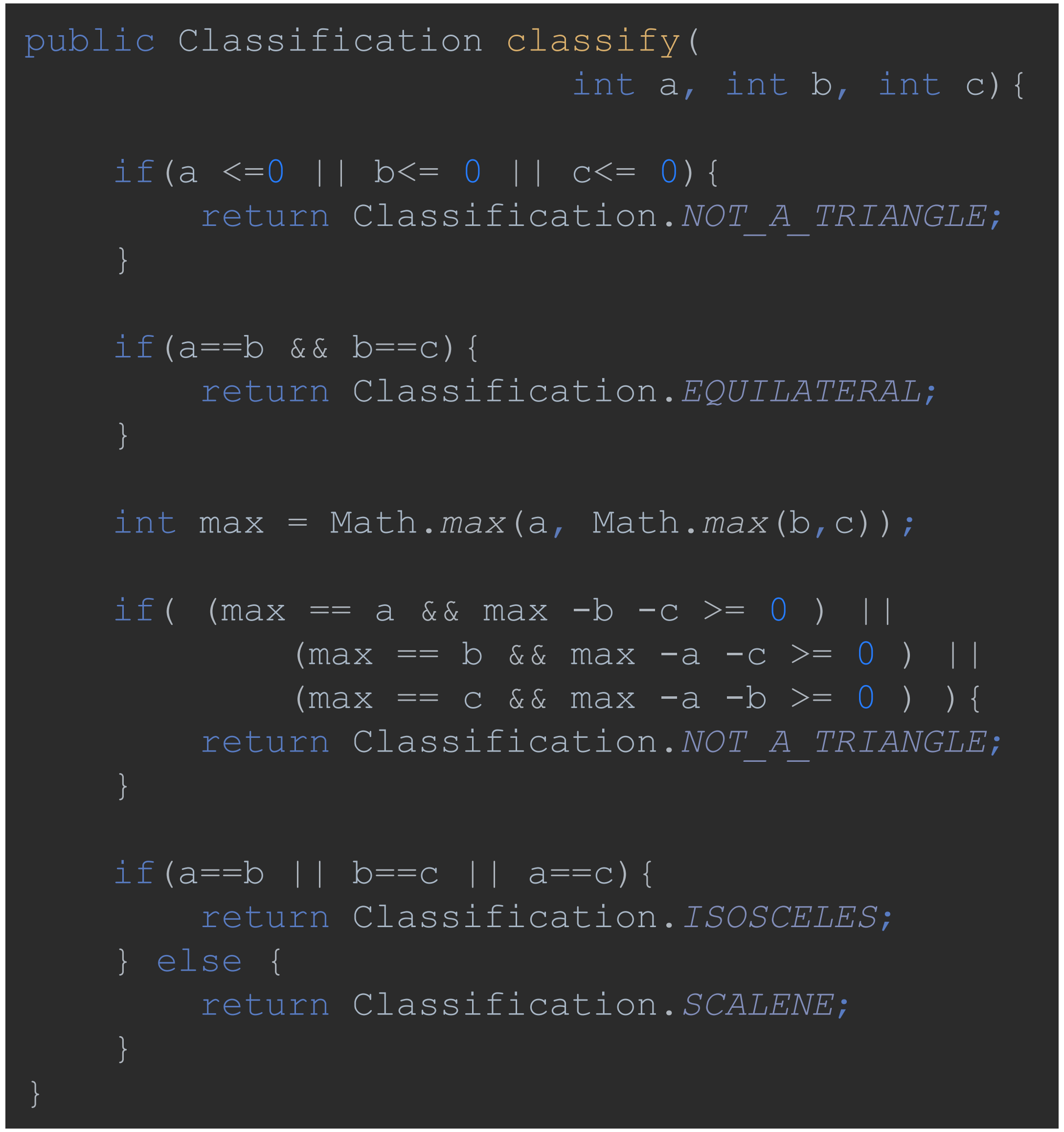
- If numbers are 32 bit integers, there are $2^{32} * 2^{32} * 2^{32} = 2^{96} = 79{,}228{,}162{,}514{,}264{,}337{,}592{,}626{,}226{,}666$ possible combinations
  - ie, 79 Octillion possible combinations of edge lengths
- Cannot test all of them
- Need to define some *test criteria* to decide a good enough test suite which is:
  1. good at finding bugs
  2. small enough to be manageable

1 Test per Output

- **t0:(A=42, B=42, C=42) => EQUILATERAL**
- **t1:(A=42, B=42, C=5) => ISOSCELES**
- **t2:(A=42, B=43, C=44) => SCALENE**
- **t3:(A=42, B=42, C=12345) = NOT A TRIANGLE**
- *Would such 4 test cases be enough?*
- What if the EQUILATERAL case is implemented with just something as naïve as “**if A==B and B==C then EQUILATERAL**”?
  - (**A=-3, B=-3, C=-3**) would wrongly return EQUILATERAL instead of NOT A TRIANGLE
  - Just checking basic scenarios is not enough

White-Box Testing

- Code can have bugs
- *To trigger a bug, the code must be executed*
- But code can have very complex control flow
- Some rare “paths” in the code might be executed only in very complex scenarios
- *Goal: in a test suite, have each single line and branch be executed at least once*



```
public Classification classify(
                            int a, int b, int c){

    if(a <=0 || b<= 0 || c<= 0){
        return Classification.NOT_A_TRIANGLE;
    }

    if(a==b && b==c){
        return Classification.EQUILATERAL;
    }

    int max = Math.max(a, Math.max(b,c));

    if( (max == a && max -b -c >= 0 ) ||
            (max == b && max -a -c >= 0 ) ||
            (max == c && max -a -b >= 0 ) ){
        return Classification.NOT_A_TRIANGLE;
    }

    if(a==b || b==c || a==c){
        return Classification.ISOSCELES;
    } else {
        return Classification.SCALENE;
    }
}
```

Example

-
  ```
  if( (max == a && max -b -c >= 0 ) ||
          (max == b && max -a -c >= 0 ) ||
          (max == c && max -a -b >= 0 ) )
  ```
- In this disjunction of 3 different clauses, if in your test suite the first clause is always true, the other 2 would never be executed
  - so if wrong, you would not know
- This is a TRIVIAL example… real industrial software can be way more complex…
- Writing tests for each path is not only tedious, but can be quite hard as well

Oracle Problem

- Given **f(x)=y**, how do I know that **y** is the correct output for **x**???
- Need an *“oracle”* to determine the correctness of output
- Easiest oracle: *has the program crashed*?
  - In this case, **y** is not correct and we have a bug
  - But not all bugs lead to a program crash…
- We get an output, might not always be easy to tell if correct

Is this correct?

(A=42, B=42, C=12345) = NOT A TRIANGLE

What about this one?

(A=890321, B=1661466711, C=7711452) = NOT A TRIANGLE

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

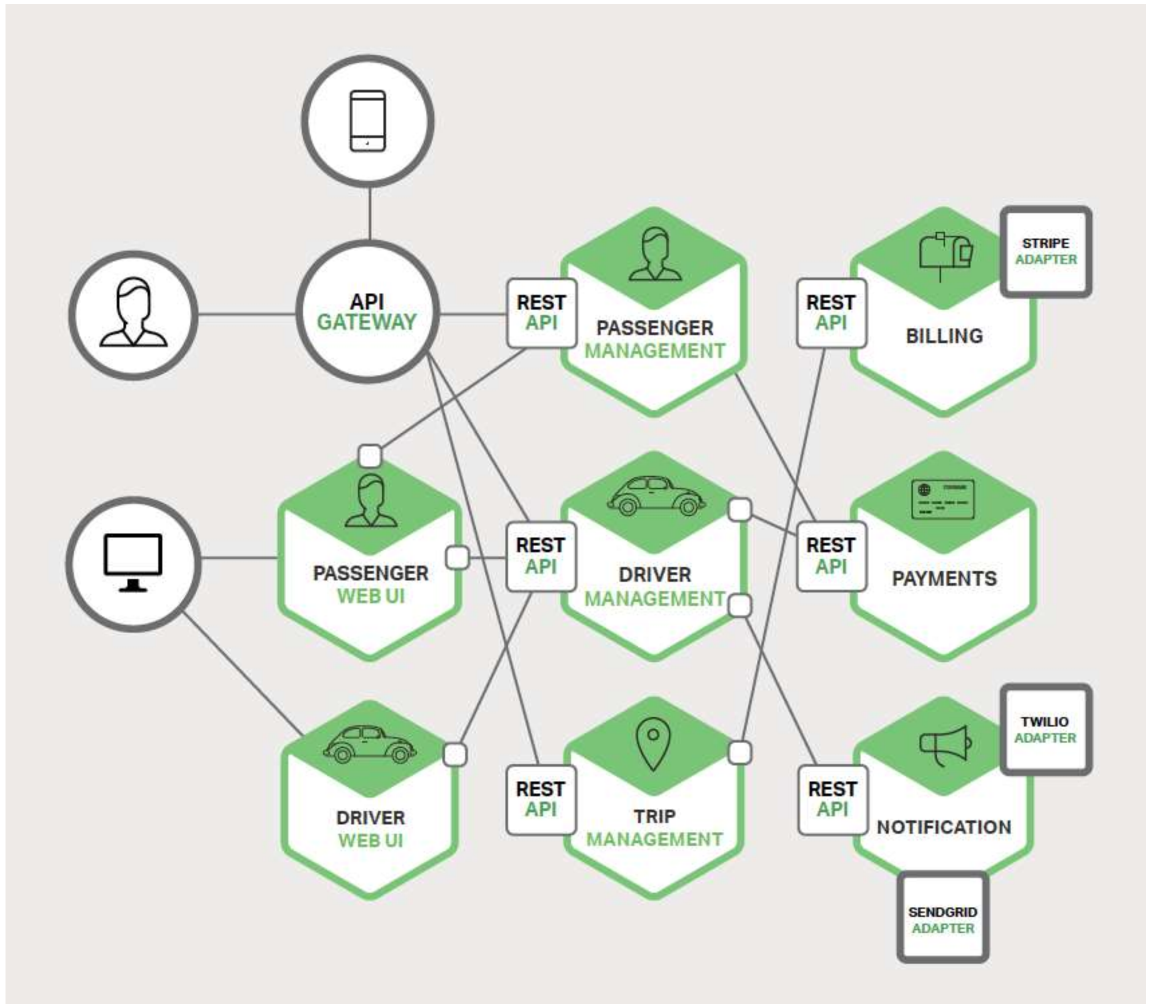
REST APIs

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

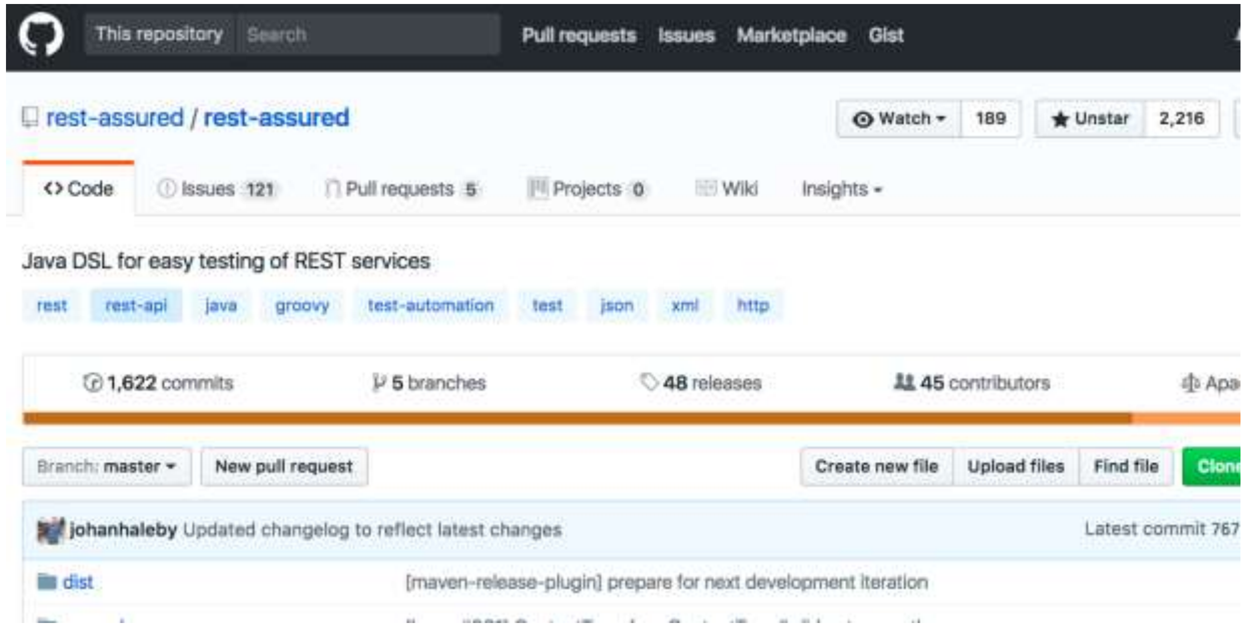
REST in Microservices

- 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**



This repository Search Pull requests Issues Marketplace Gist

rest-assured / rest-assured

Watch 189 Unstar 2,216

Code Issues 121 Pull requests 5 Projects 0 Wiki Insights

Java DSL for easy testing of REST services

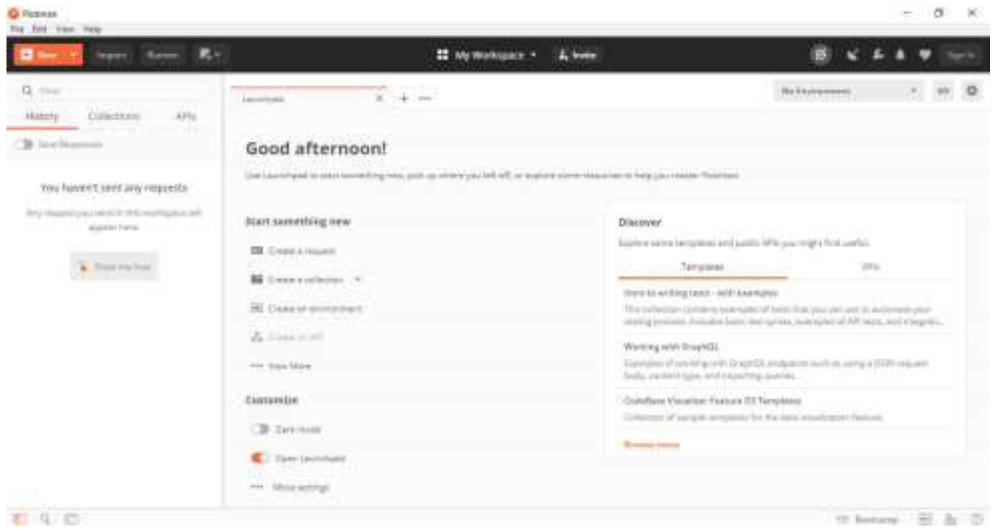
rest rest-api java groovy test-automation test json xml http

1,622 commits 5 branches 48 releases 45 contributors Apa

Branch: master New pull request Create new file Upload files Find file Clon

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

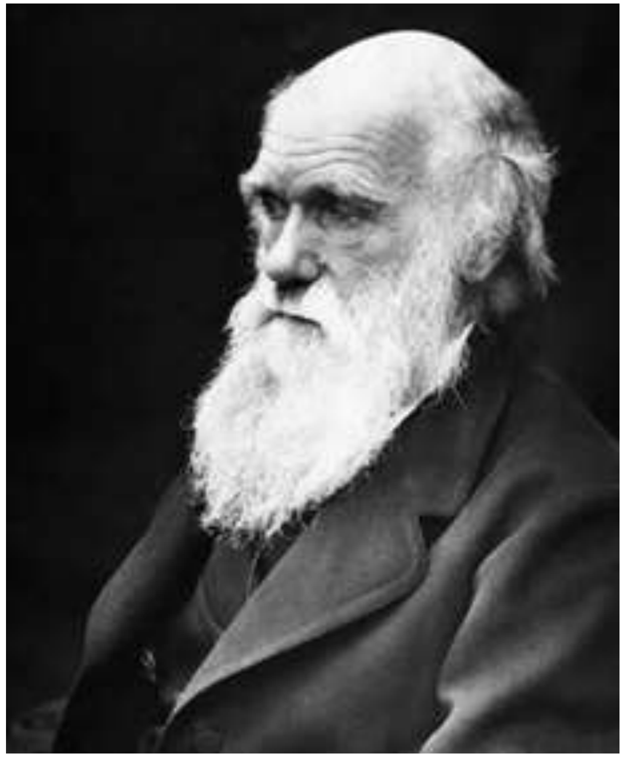
What about **Automated Test Generation** for RESTful APIs?

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

Search Algorithms

Search-Based Software Testing (SBST)

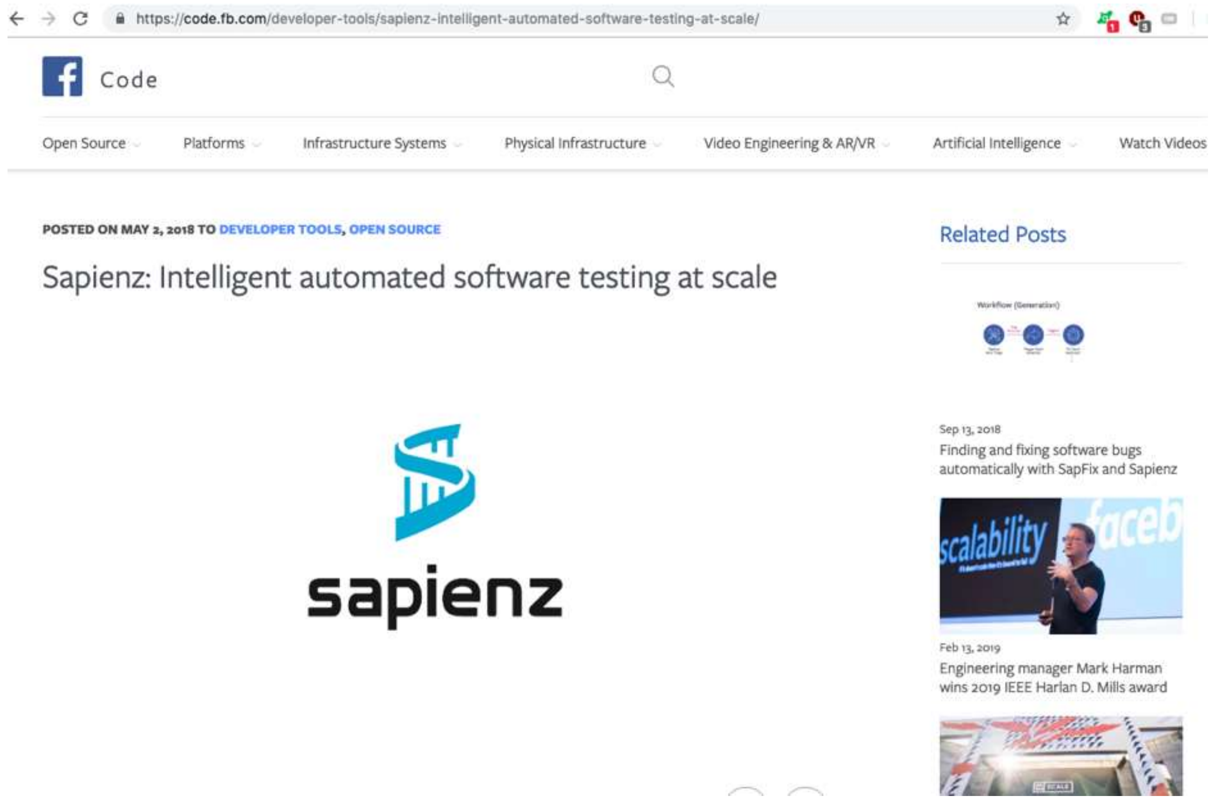
- Biology meets Software Engineering (SE)
- Casting SE problems into *Optimization Problems*
- *Genetic Algorithms*: one of most famous optimization algorithm, based on theory of evolution
- *Evolve* test cases



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";
}
```

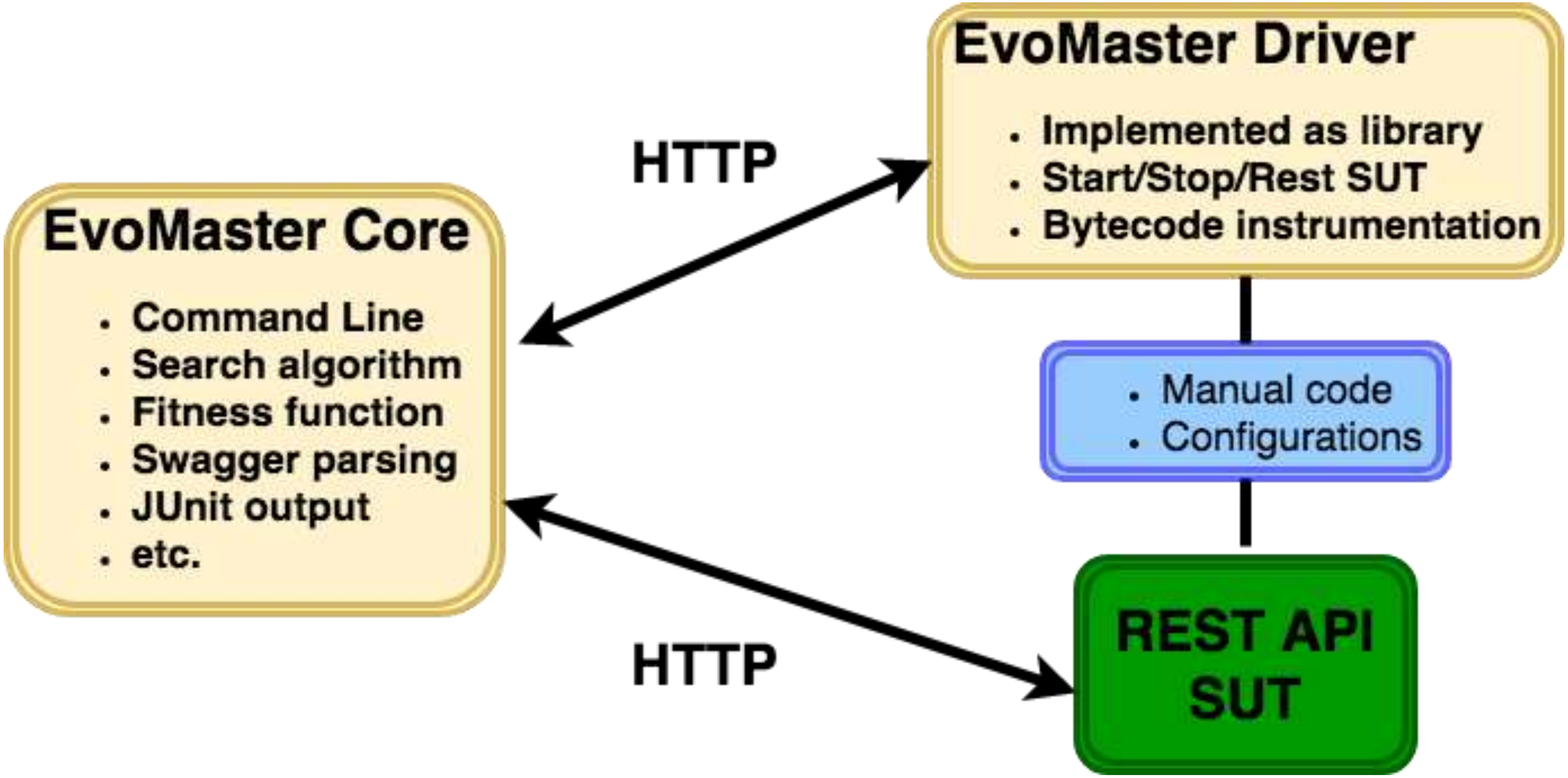
SBST Heuristics: Branch Distance

- 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

- 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*


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

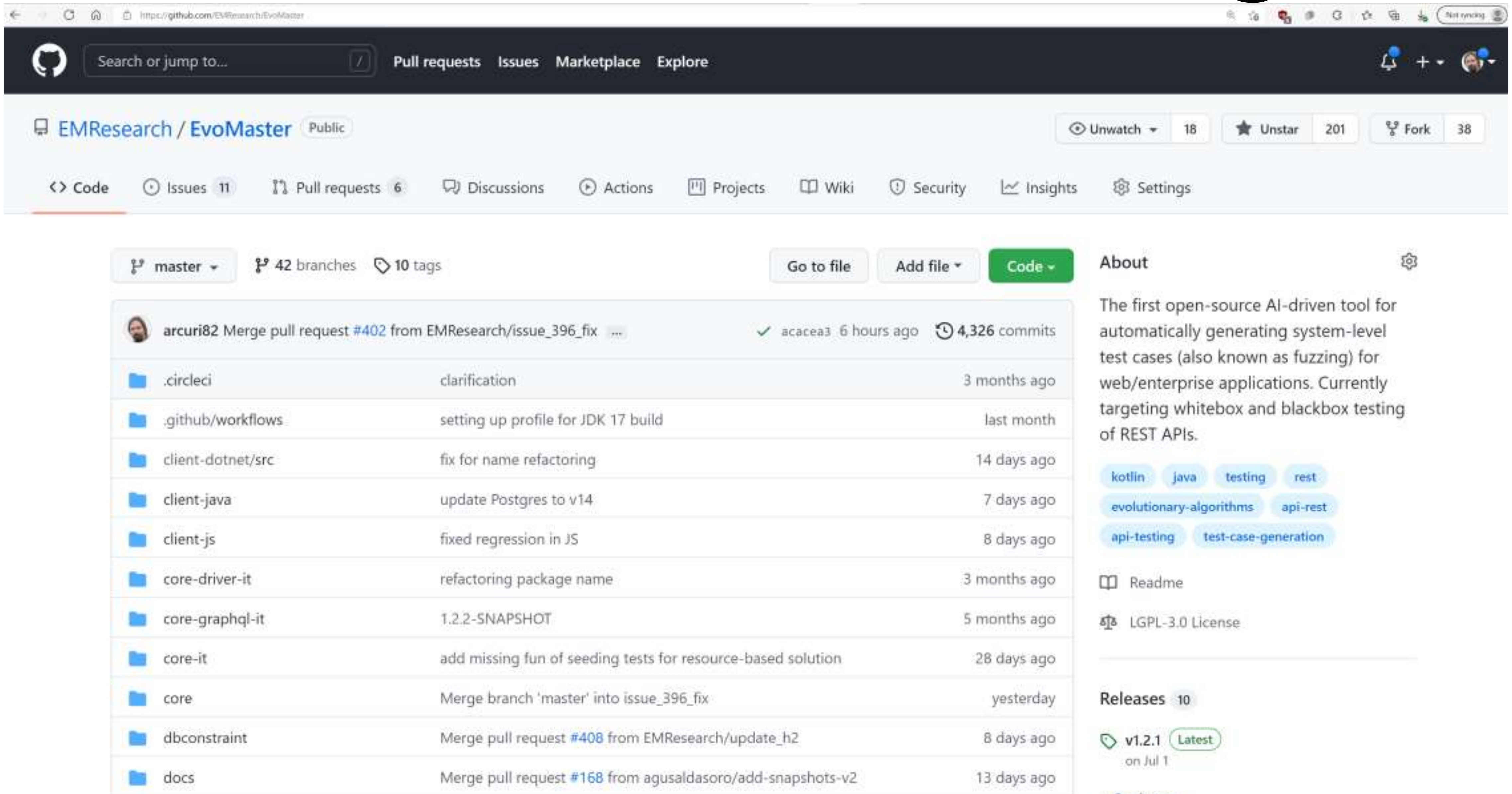
Fitness Function

- 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

www.evomaster.org



https://github.com/EMResearch/EvoMaster

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 agusaldasoro/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

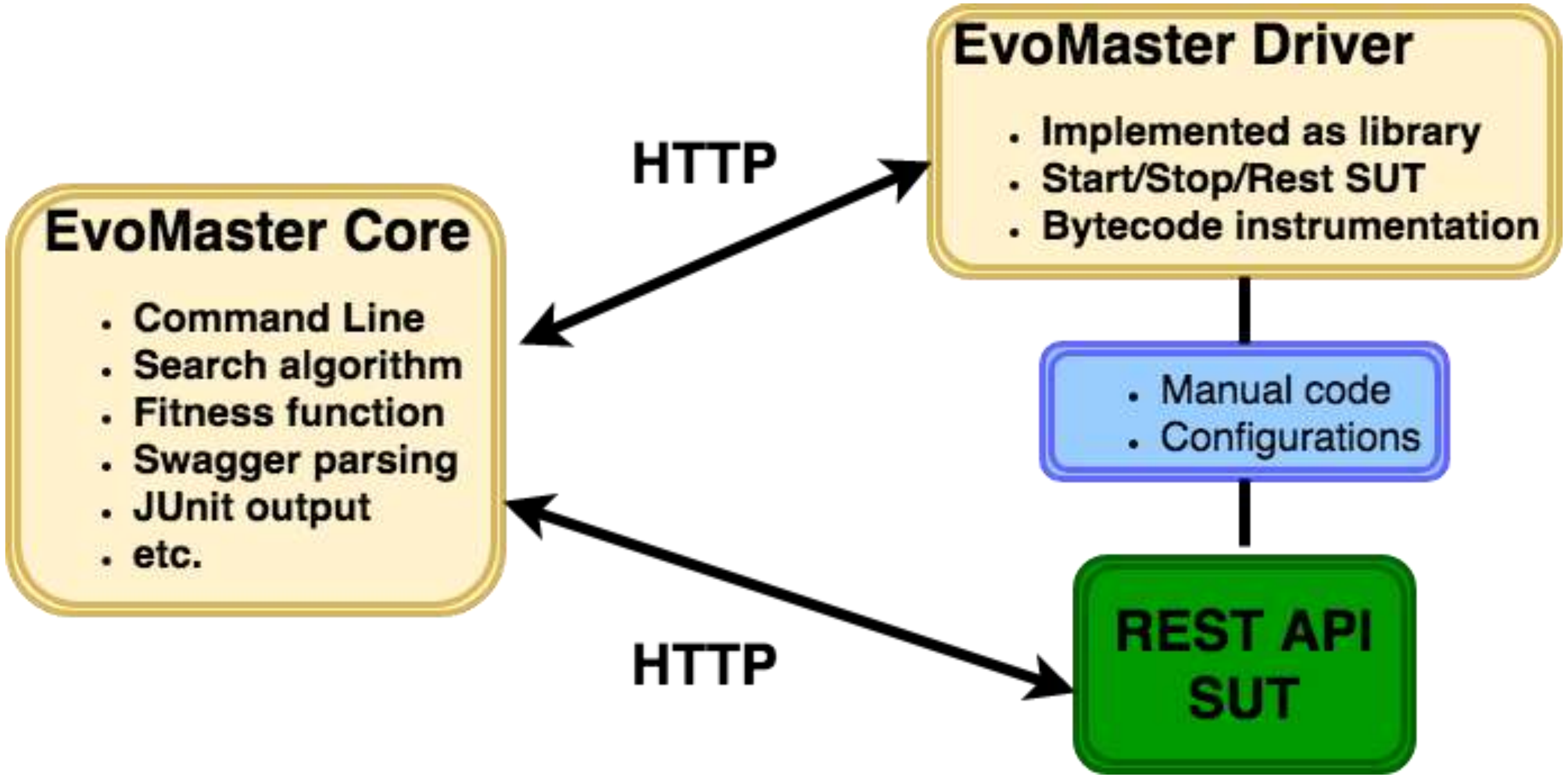
Releases 10

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 code/bytecode analysis
- Future: support for **Frontend Web GUIs** (eg, actions on browser)

Demo


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