Тестирование систем с внешними зависимостями

Проблемы, решения, Mountebank.

Глазков Андрей
Paysystem.tech
andrewglazkov@gmail.com
@glazz87

План доклада

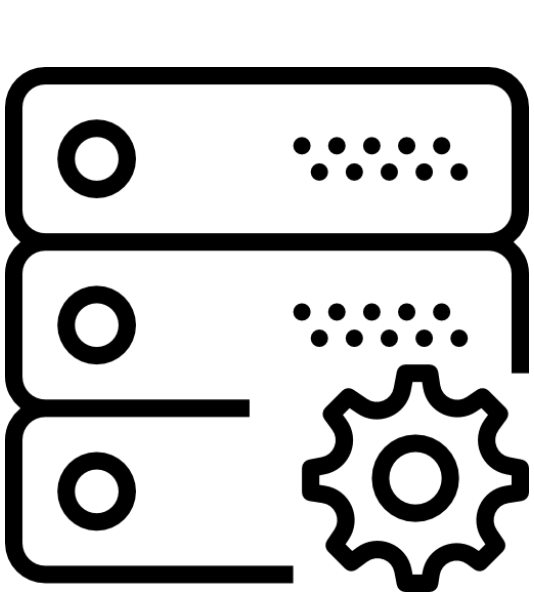
1. Проблемы тестирования систем с внешними зависимостями
2. Различные подходы к решению этих проблем
3. Мок-сервер как универсальное решение
4. Mountebank
5. Ограничения подхода

Обо мне

- Более 10 лет в тестировании ПО
- Пишу на python
- Сферы интереса: эффективная автоматизация в QA, процессы обеспечения качества в целом
- Руководитель отдела качества в компании Paysystem.tech

PAY SYSTEM .TECH

Внешние зависимости



HTTP

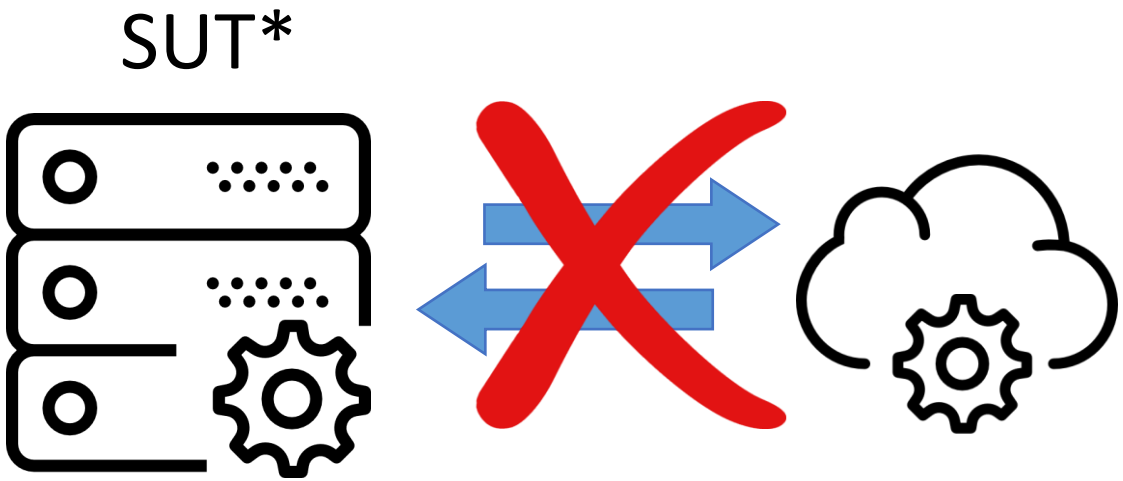
TCP

HTTPS

Проблемы

Проблемы

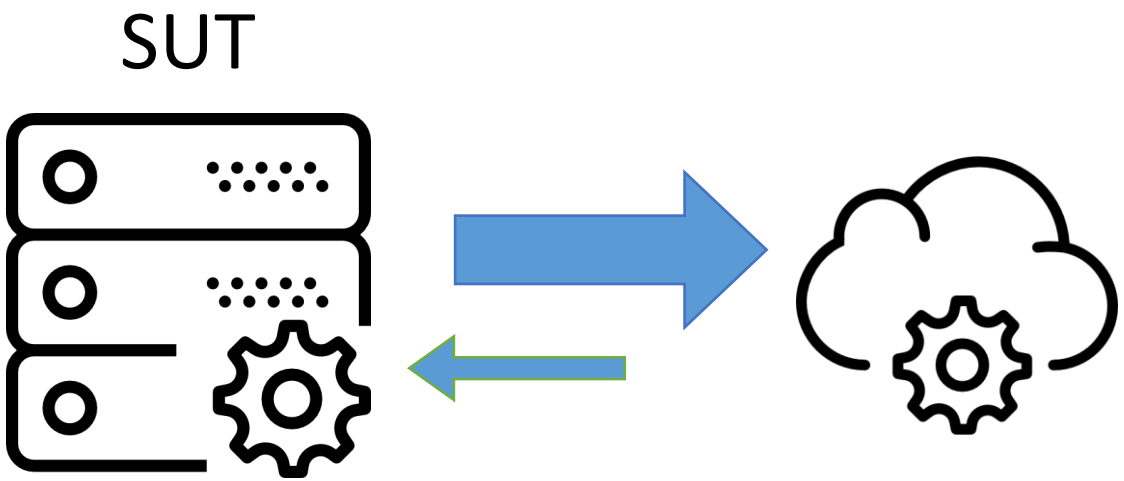
1. Внешняя система недоступна, либо работает некорректно



* - System Under Test

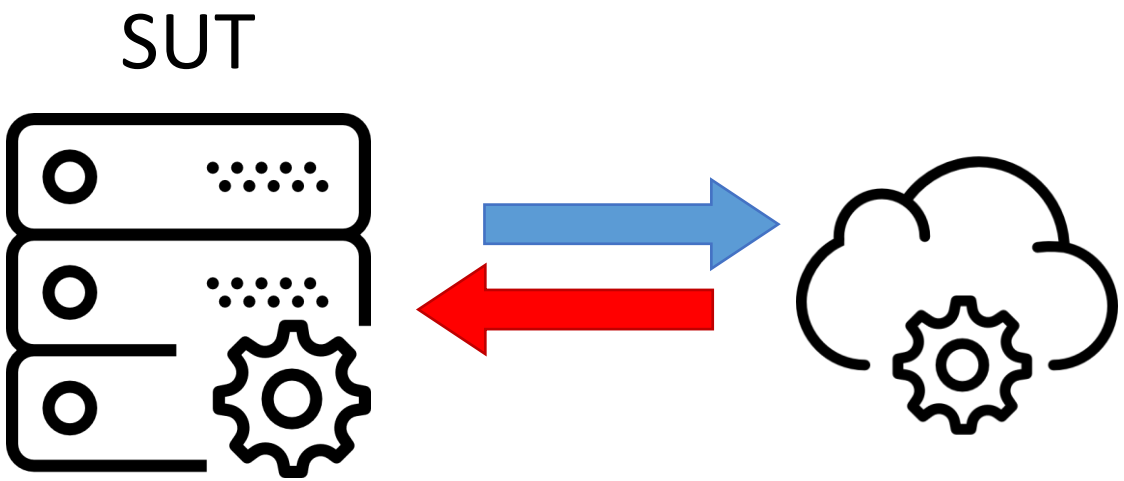
Проблемы

1. Внешняя система недоступна, либо работает некорректно
2. Ограниченный набор данных во внешней системе



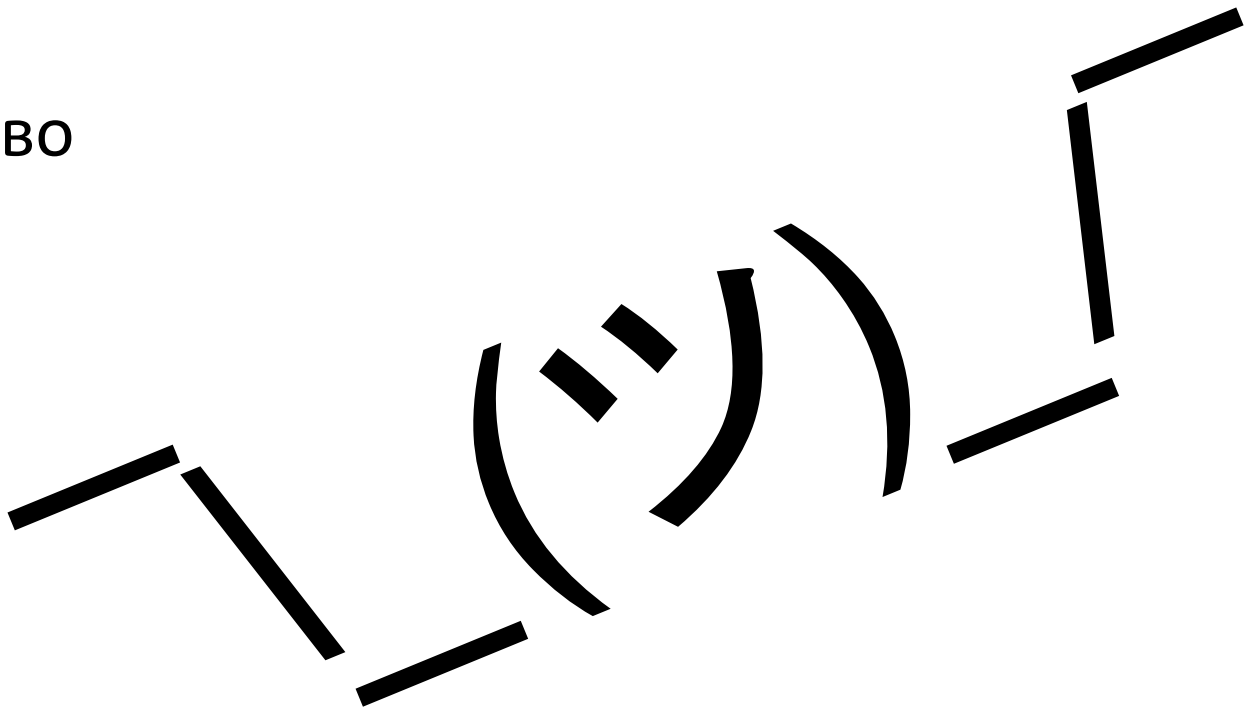
Проблемы

1. Внешняя система недоступна, либо работает некорректно
2. Ограниченный набор данных во внешней системе
3. Исключительные ситуации тоже надо проверять



Проблемы

1. Внешняя система недоступна, либо работает некорректно
2. Ограниченный набор данных во внешней системе
3. Исключительные ситуации тоже надо проверять

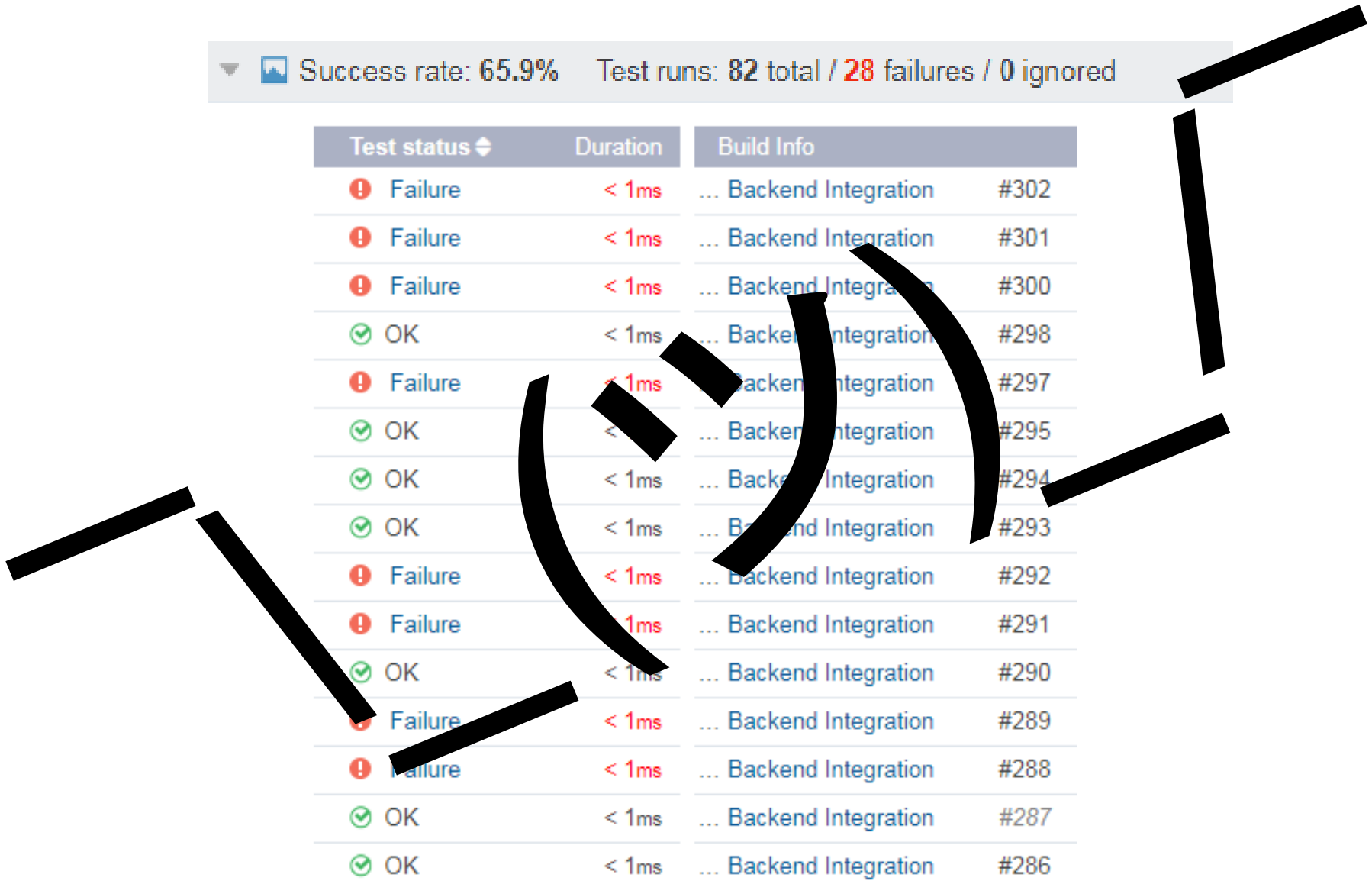


Автотесты -> проблемы^2

Success rate: 65.9% Test runs: 82 total / 28 failures / 0 ignored

Test status	Duration	Build Info	
Failure	< 1ms	... Backend Integration	#302
Failure	< 1ms	... Backend Integration	#301
Failure	< 1ms	... Backend Integration	#300
OK	< 1ms	... Backend Integration	#298
Failure	< 1ms	... Backend Integration	#297
OK	< 1ms	... Backend Integration	#295
OK	< 1ms	... Backend Integration	#294
OK	< 1ms	... Backend Integration	#293
Failure	< 1ms	... Backend Integration	#292
Failure	< 1ms	... Backend Integration	#291
OK	< 1ms	... Backend Integration	#290
Failure	< 1ms	... Backend Integration	#289
Failure	< 1ms	... Backend Integration	#288
OK	< 1ms	... Backend Integration	#287
OK	< 1ms	... Backend Integration	#286

Автотесты -> проблемы^2



Success rate: 65.9% Test runs: 82 total / 28 failures / 0 ignored

Test status	Duration	Build Info
Failure	< 1ms	... Backend Integration #302
Failure	< 1ms	... Backend Integration #301
Failure	< 1ms	... Backend Integration #300
OK	< 1ms	... Backend Integration #298
Failure	< 1ms	... Backend Integration #297
OK	< 1ms	... Backend Integration #295
OK	< 1ms	... Backend Integration #294
OK	< 1ms	... Backend Integration #293
Failure	< 1ms	... Backend Integration #292
Failure	< 1ms	... Backend Integration #291
OK	< 1ms	... Backend Integration #290
Failure	< 1ms	... Backend Integration #289
Failure	< 1ms	... Backend Integration #288
OK	< 1ms	... Backend Integration #287
OK	< 1ms	... Backend Integration #286

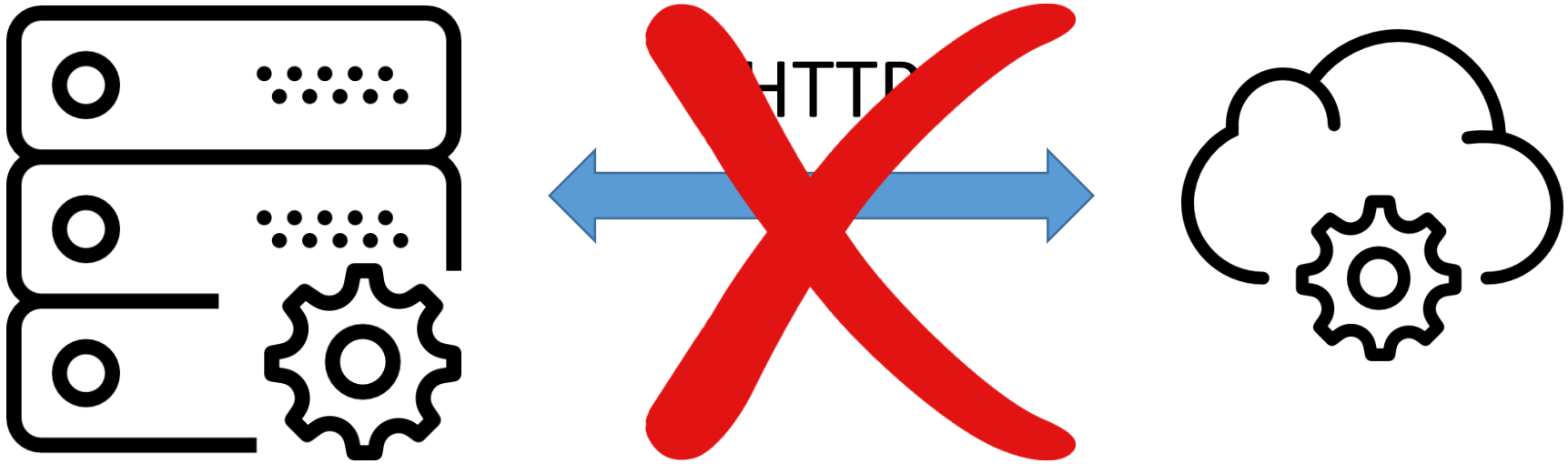
Результат

1. Сроки разработки срываются
2. Тесты не покрывают весь необходимый функционал
3. Автотесты нестабильны
4. **Напряжение в команде растет**

Мокирование на уровне кода

SUT

Внешняя система



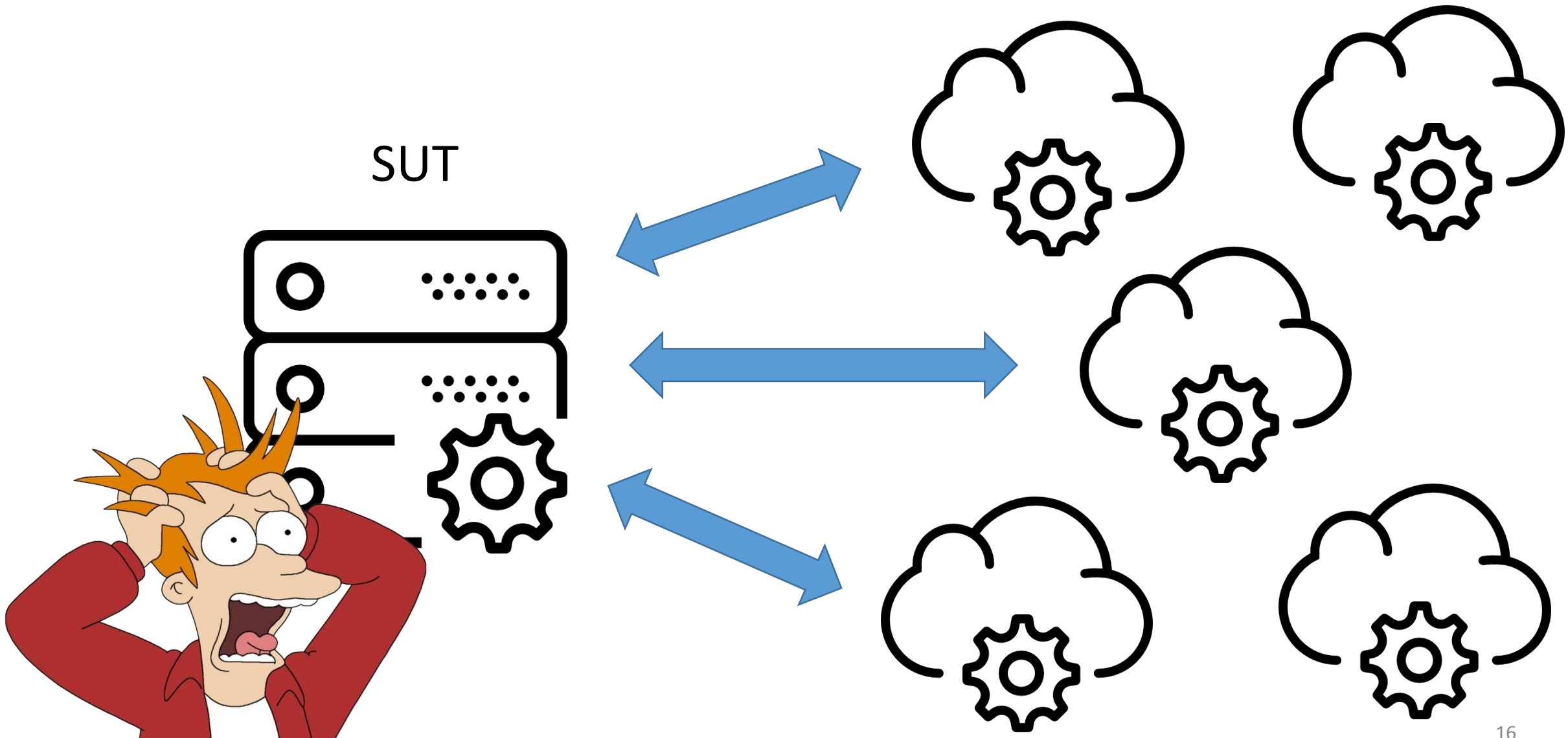
Мокирование на уровне кода

```
from unittest.mock import Mock
ext_sys = ExternalSystemClass()
ext_sys.method = Mock(return_value=(100,"rub"))
```

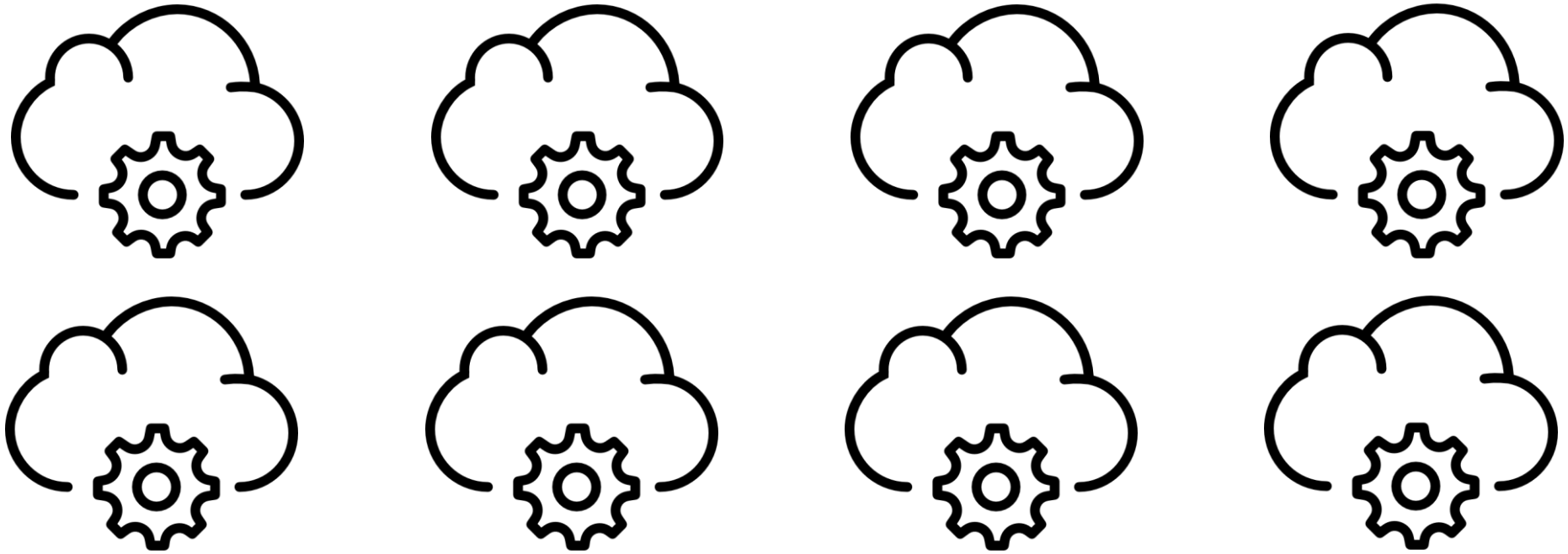
Мокирование на уровне кода

```
from unittest.mock import Mock
ext_sys = ExternalSystemClass()
if user.number == "9267851111"
  ext_sys.method = Mock(return_value=(100,"rub"))
if user.number == "9267852222"
  ext_sys.method = Mock(return_value=(0,"rub"))
if user.number == "9267853333"
  ext_sys.method = Mock(return_value=(-100,"rub"))
...
...
...
```

Зависимостей – много



Комбинаторный взрыв



5 вариаций ^ 3 внешних системы = 125 комбинаций

5 вариаций ^ 8 внешних систем = 390625 комбинаций

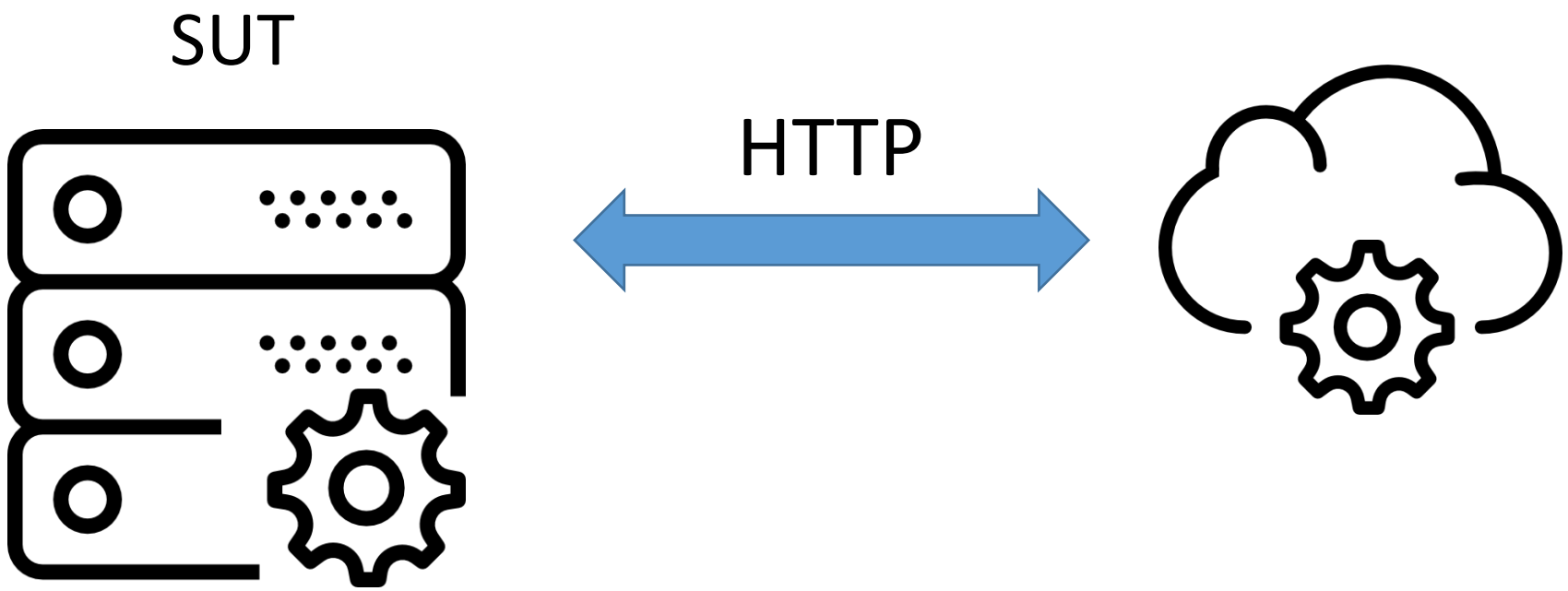
Мокирование на уровне кода

```
from unittest.mock import Mock
ext_sys = ExternalSystemClass()
if user.number == "926425138"
    ext_sys.method = Mock(return_value=(100,"rub"))
if user.number == "926785349"
    ext_sys.method = Mock(return_value=(0,"rub"))
if user.number == "926235191"
    ext_sys.method = Mock(return_value=(-100,"rub"))
...
...
...
```

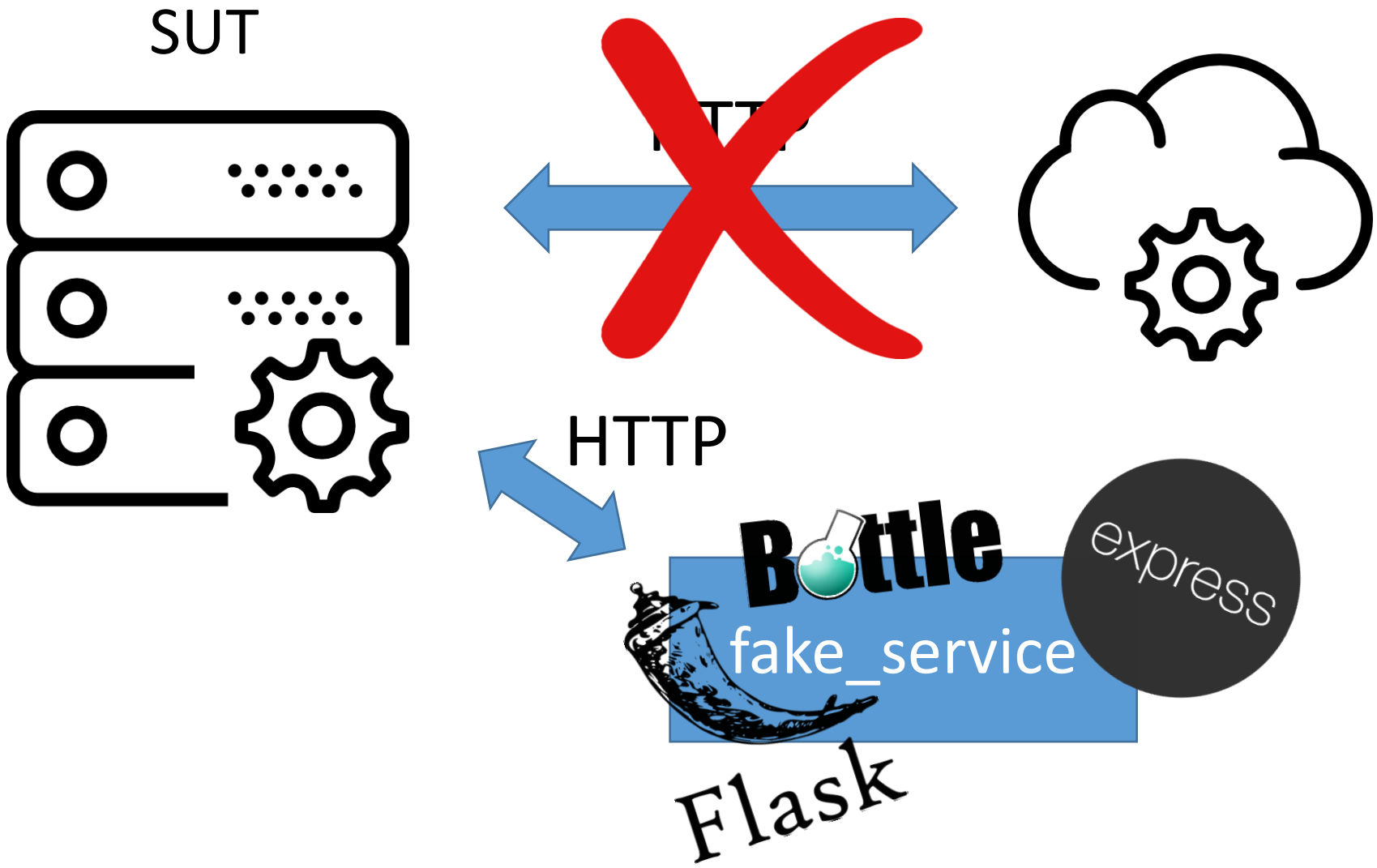
Мокирование на уровне кода: результат

1. Паразитный тестовый код внутри приложения
2. Тестировщик не управляет тестовыми данными
3. Не проверяется взаимодействие с реальной системой
4. Не тестируются исключительные ситуации

Своя реализация внешнего сервиса



Своя реализация внешнего сервиса



Своя реализация внешнего сервиса

Фейки делятся на две категории:

«глупые» и «умные»

Своя реализация внешнего сервиса

Глупый фейк

- Быстро пишутся и очень простые
- Ничего не умеют кроме конкретного тестового случая
- Приходится писать новый фейк для каждого тестового случая
- Сложная перенастройка в процессе тестирования

```
from flask import Flask, jsonify
app = Flask(__name__)

@app.route('/balance/9260219812',
           methods=['GET'])
def payment():
    response = {
        'balance': '100'
    }
    return jsonify(response), 200
```

Своя реализация внешнего сервиса

Умный фейк

- Умеют всё так, как настоящая внешняя система
- Не нужно перенастраивать
- Может быть **очень** сложная логика
- Может быть **очень** много кода
- Не позволяют тестировать «особенные» случаи

```python
from app import app, db
from flask import request
from xml.etree import ElementTree
from app.models import Subscriber, BillingService
from app.responses import SERVICE_NOT_FOUND, ALREADY_DEACTIVATED, SUCCESS_ACTIVATE, \
    ALREADY_ACTIVATED, SUCCESS_DEACTIVATE, FAULT, UNKNOWN_SERVICE_TYPE, ONLY_ONE_PARAMETER

resp_header = '''<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<s:Header xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" />
<SOAP-ENV:Body xmlns:ber-ns0="http://www.bercut.com/schema/PCServices">
'''

resp_footer = '''
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
'''

ns = {'s': 'http://schemas.xmlsoap.org/soap/envelope/',
      'ber-ns0': 'http://www.bercut.com/schema/PCServices'}

@app.before_request
def log_request_info():
    app.logger.info('Request: {} {}'.format(request.method, request.path))
    app.logger.info('Request headers: %s', request.headers)
    app.logger.info('Request body: %s', request.get_data().decode('utf8'))

@app.after_request
def log_response_info(response):
    app.logger.info('Response status: {}'.format(response.status))
    app.logger.info('Response body: %s', response.get_data().decode('utf8'))
    return response

@app.after_request
def apply_caching(response):
    response.headers['Content-Type'] = 'text/xml;charset=UTF-8'
    return response


@app.route('/wsdl/PCServices/PCServices', methods=['POST', 'GET'])
def index():
    incoming_data = request.data.decode()
    root = ElementTree.fromstring(incoming_data)

    # getsubscriberservicesrequest
    if root.find('s:Body', ns).find('ber-ns0:getSubscriberServicesRequest', ns) is not None:
        msisdn = root.find('s:Body/'
                           'ber-ns0:getSubscriberServicesRequest/'
                           'ber-ns0:msisdn', ns).text

        service_type = root.find('s:Body/'
                                 'ber-ns0:getSubscriberServicesRequest/'
                                 'ber-ns0:serviceType', ns).text

        all_subscribers = [subscriber.msisdn for subscriber in Subscriber.query.all()]

        if service_type == 'all':
            all_services = BillingService.query.all()
            all_services_details = ''
            for service in all_services:
                all_services_details += service.details
            return resp_header \
                   + '<ber-ns0:getSubscriberServicesResponse>' \
                   + all_services_details \
                   + '</ber-ns0:getSubscriberServicesResponse>' \
                   + resp_footer

        elif service_type == 'available':
            if (msisdn not in all_subscribers):
                all_services = BillingService.query.all()
                all_services_details = ''
                for service in all_services:
                    all_services_details += service.details
                return resp_header \
                       + '<ber-ns0:getSubscriberServicesResponse>' \
                       + all_services_details \
                       + '</ber-ns0:getSubscriberServicesResponse>' \
                       + resp_footer
            else:
                subscriber = Subscriber.query.filter_by(msisdn=msisdn).first()
                enabled_services = subscriber.enabled_services
                enabled_services_ids = [s.id for s in enabled_services]

                available_services = BillingService.query.filter(
                    BillingService.id.notin_(enabled_services_ids))
                available_services_details = ''
                for s in available_services:
                    available_services_details += s.details
                return resp_header \
                       + '<ber-ns0:getSubscriberServicesResponse>' \
                       + available_services_details \
                       + '</ber-ns0:getSubscriberServicesResponse>' \
                       + resp_footer

        elif service_type == 'enabled':
            if (msisdn not in all_subscribers):
                services_details = ''
                return resp_header + services_details + resp_footer
            else:
                subscriber = Subscriber.query.filter_by(msisdn=msisdn).first()
                enabled_services = subscriber.enabled_services
                enabled_services_ids = [s.id for s in enabled_services]

                enabled_services = BillingService.query.filter(
                    BillingService.id.in_(enabled_services_ids))
                enabled_services_details = ''
                for s in enabled_services:
                    enabled_services_details += s.details
                return resp_header \
                       + '<ber-ns0:getSubscriberServicesResponse>' \
                       + enabled_services_details \
                       + '</ber-ns0:getSubscriberServicesResponse>' \
                       + resp_footer
        else:
            return UNKNOWN_SERVICE_TYPE
```

Своя реализация внешнего сервиса

Глупый/умный фейк: общие проблемы

- Требуют от тестировщика компетенций в написании web-сервисов
- Масса дополнительного кода в проекте
- Стабильность, скорость, удобство — на плечах тестера
- Дополнительная инфраструктура

Своя реализация внешнего сервиса

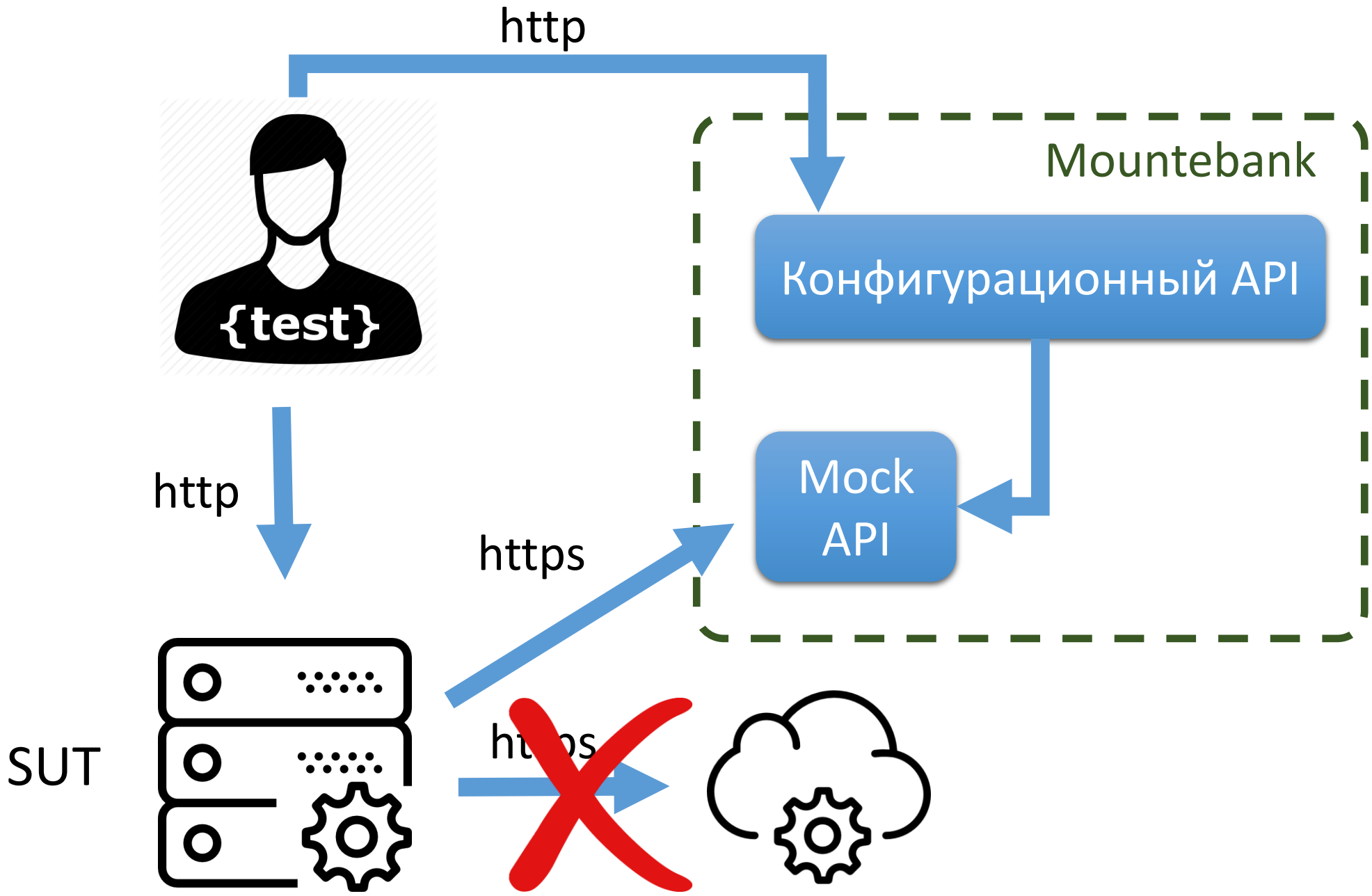
Как должно выглядеть идеальное решение?

- Фейк с возможностью переопределения функциональности «на лету»
- Возможность переопределения из кода автотестов
- Возможность проксирования
- Возможность работы не только по HTTP протоколу, но и по TCP
- Ведение логов прошедших запросов
- Возможность работы не только автоматизаторам, но и ручные тестировщики

Mock-сервер Mountebank

http://www.mbtest.org/

https://github.com/bbyars/mountebank


http
{test}
Mountebank
Конфигурационный API
http
Mock
API
https
SUT

Конфигурация mock-api

```
"port" : 44002,
"protocol" : "https",
"stubs" : [{
    "predicates" : [{
        "and" : [ {"equals" : {"path" : "/balance/79260219812",
                                "method" : "GET"} } ]
      }
    ],
    "responses" : [{
        "is" : { "body" : { "balance" : 100 },
                 "headers" : { "Content-Type" : "application/json" }
        }
      }
    ]
  }
]
```

Конфигурация mock-api

```
"port" : 44002,
"protocol" : "https",
"stubs" : [{
    "predicates" : [{
        "and" : [ {"equals" : {"path" : "/balance/79260219812",
                                "method" : "GET"} } ]
      }
    ],
    "responses" : [{
        "is" : { "body" : { "balance" : 100 },
                 "headers" : { "Content-Type" : "application/json" }
        }
      }
    ]
  }
]
```

Конфигурация mock-api

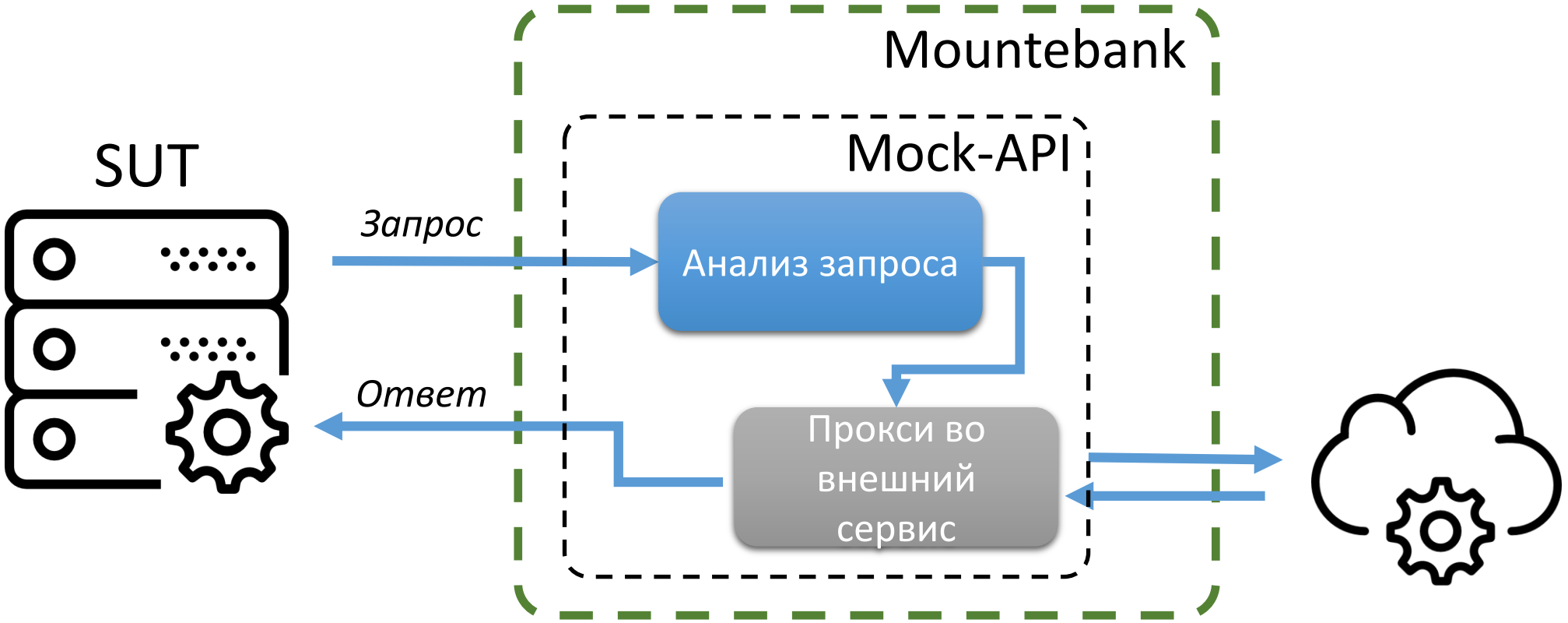
```
"port" : 44002,
"protocol" : "https",
"stubs" : [{
    "predicates" : [{
        "and" : [ {"equals" : {"path" : "/balance/79260219812",
                                        "method" : "GET"} } ]
      }
    ],
    "responses" : [{
        "is" : { "body" : { "balance" : 100 },
                 "headers" : { "Content-Type" : "application/json" }
        }
      }
    ]
  }
]
```

Конфигурация mock-api

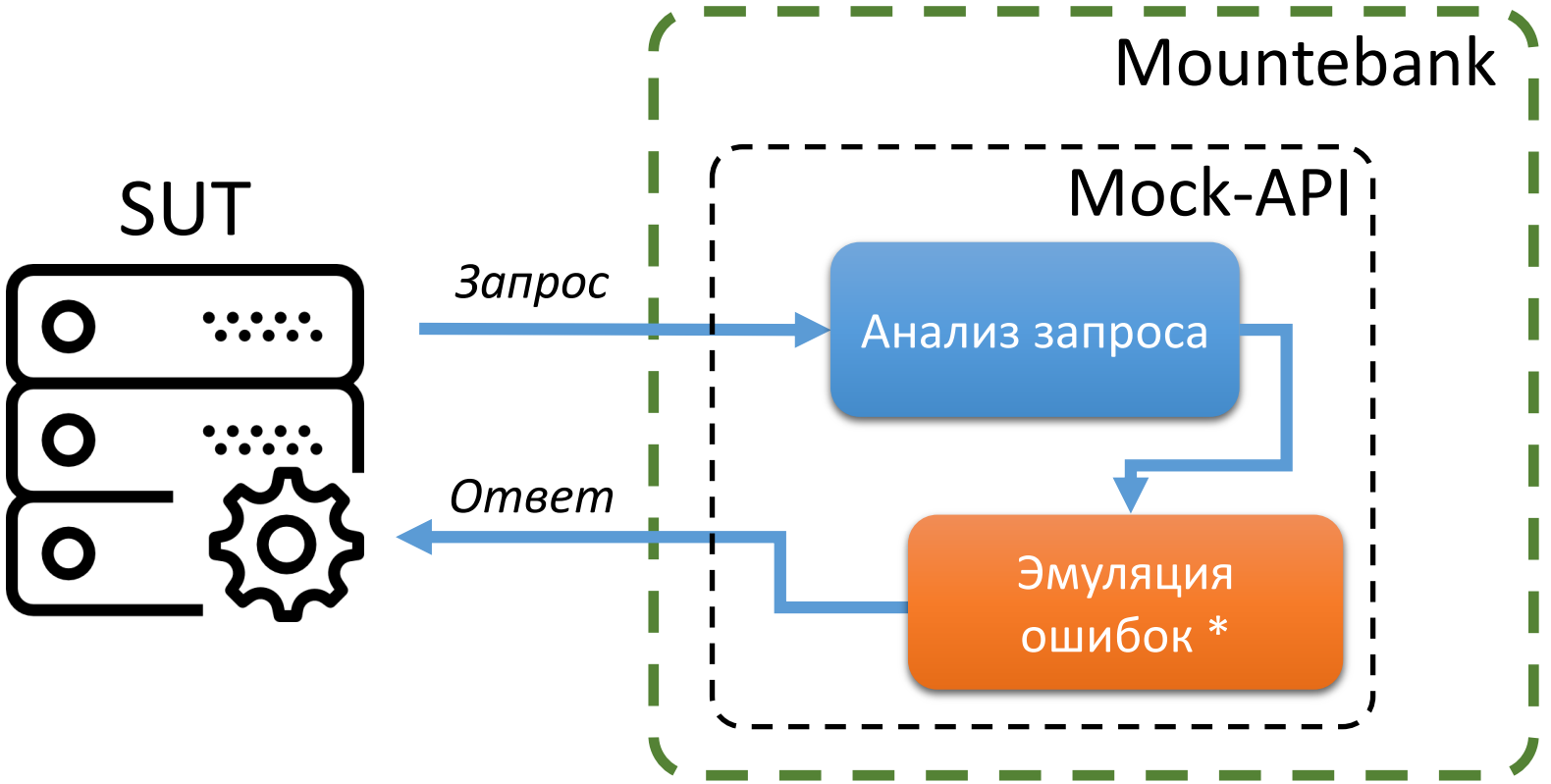
```
"port" : 44002,
"protocol" : "https",
"stubs" : [{
    "predicates" : [{
        "and" : [ {"equals" : {"path" : "/balance/79260219812",
                                        "method" : "GET"} } ]
      }
    ],
    "responses" : [{
        "is" : { "body" : { "balance" : 100 },
                 "headers" : { "Content-Type" : "application/json" }
        }
      }
    ]
  }
]
```


SHOW
TIME!

Mock-сервер: проксирование

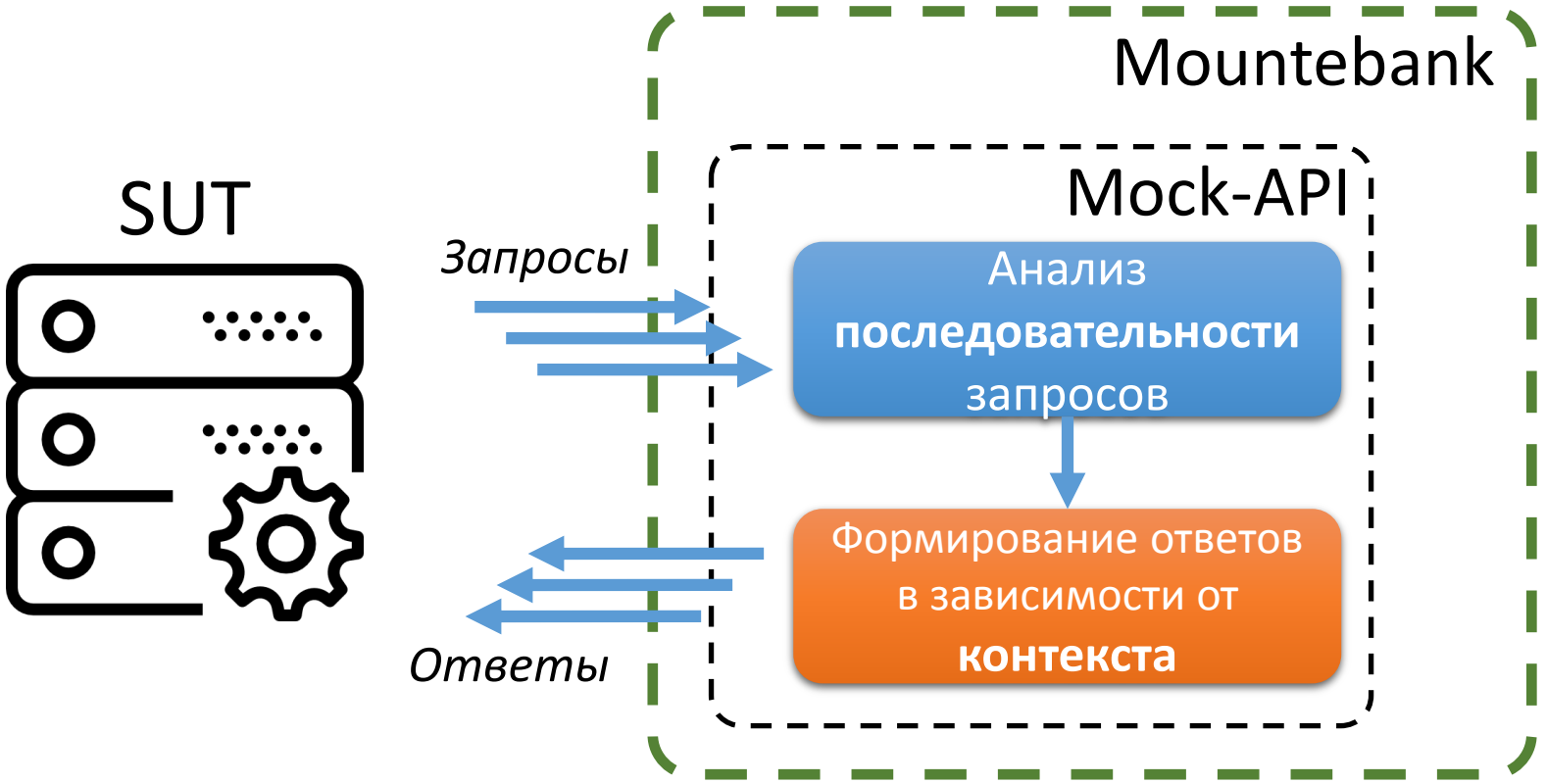


Mock-сервер: эмуляция ошибок

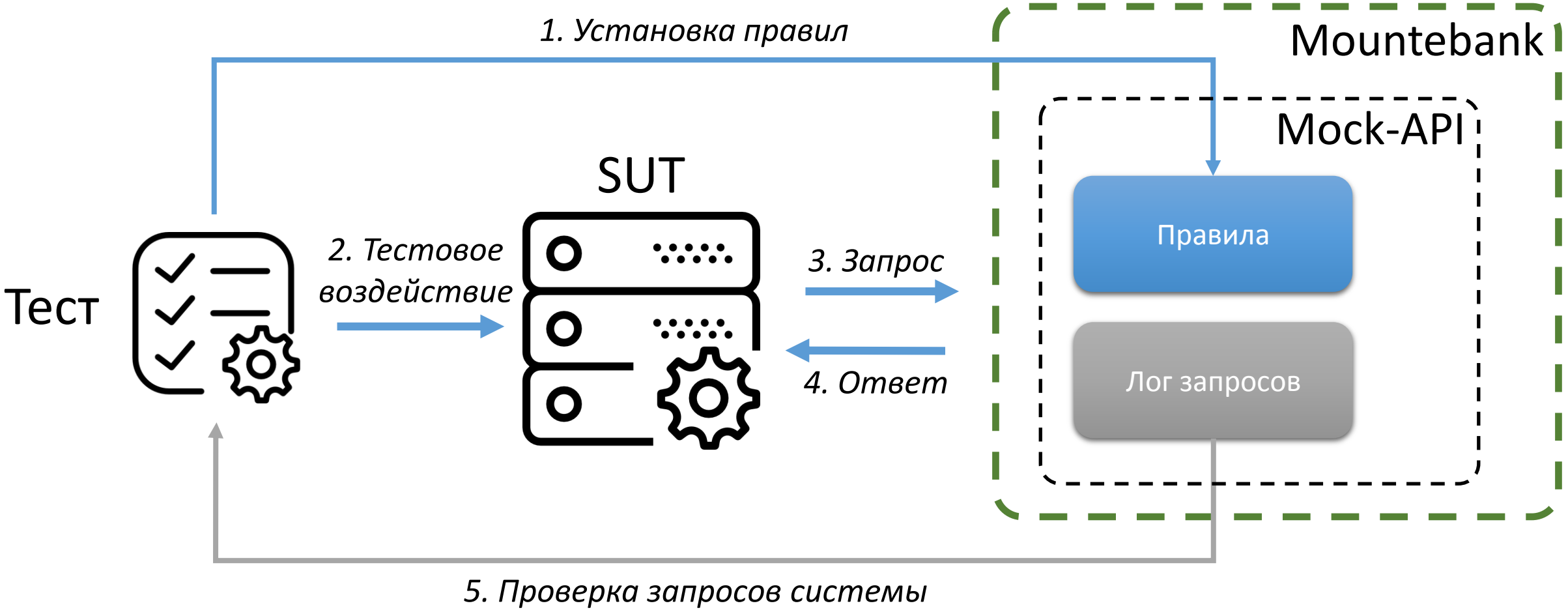


* - любой HTTP статус, задержка ответа, обрыв соединения

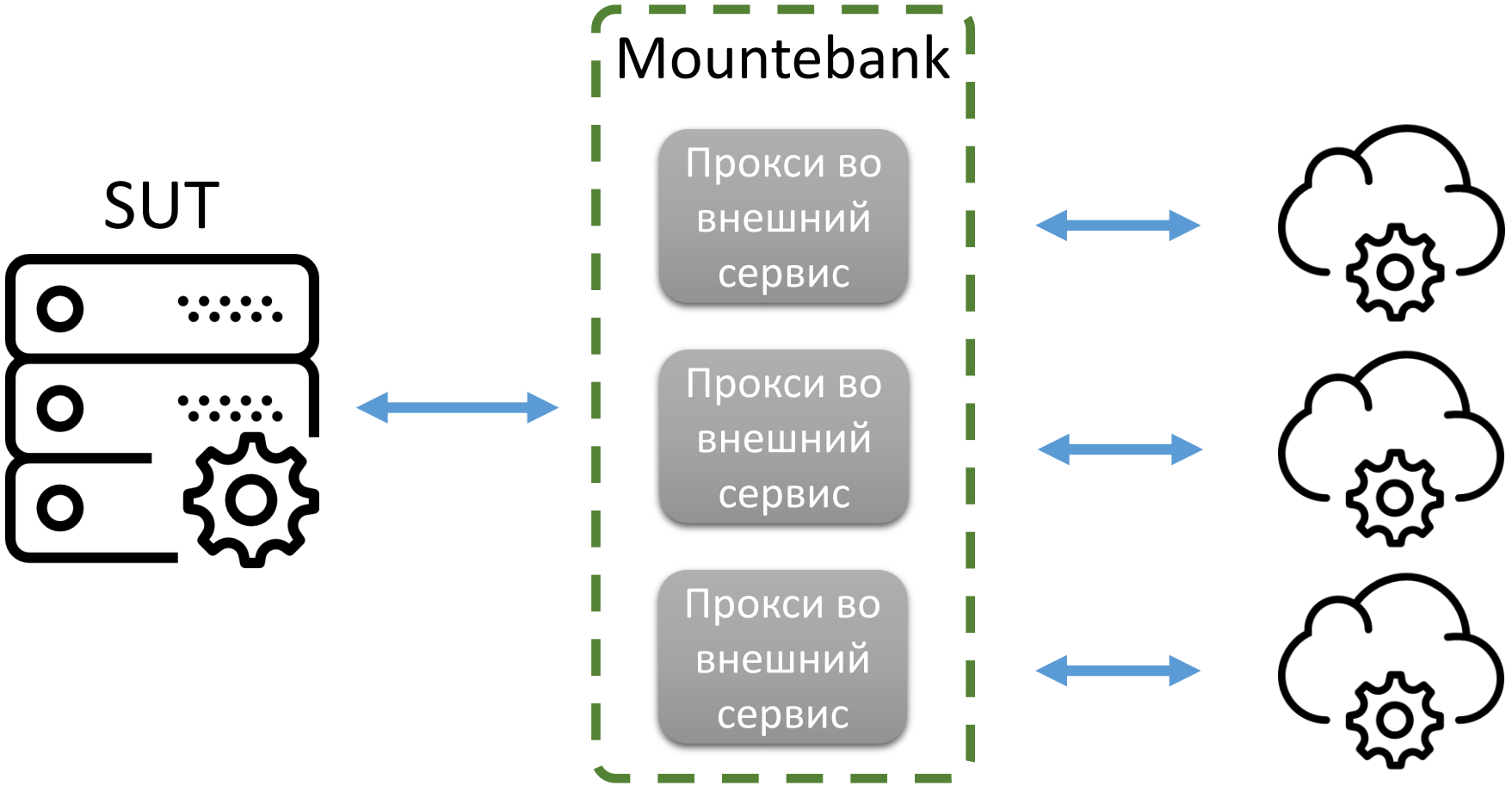
Mock-сервер: контекст запросов



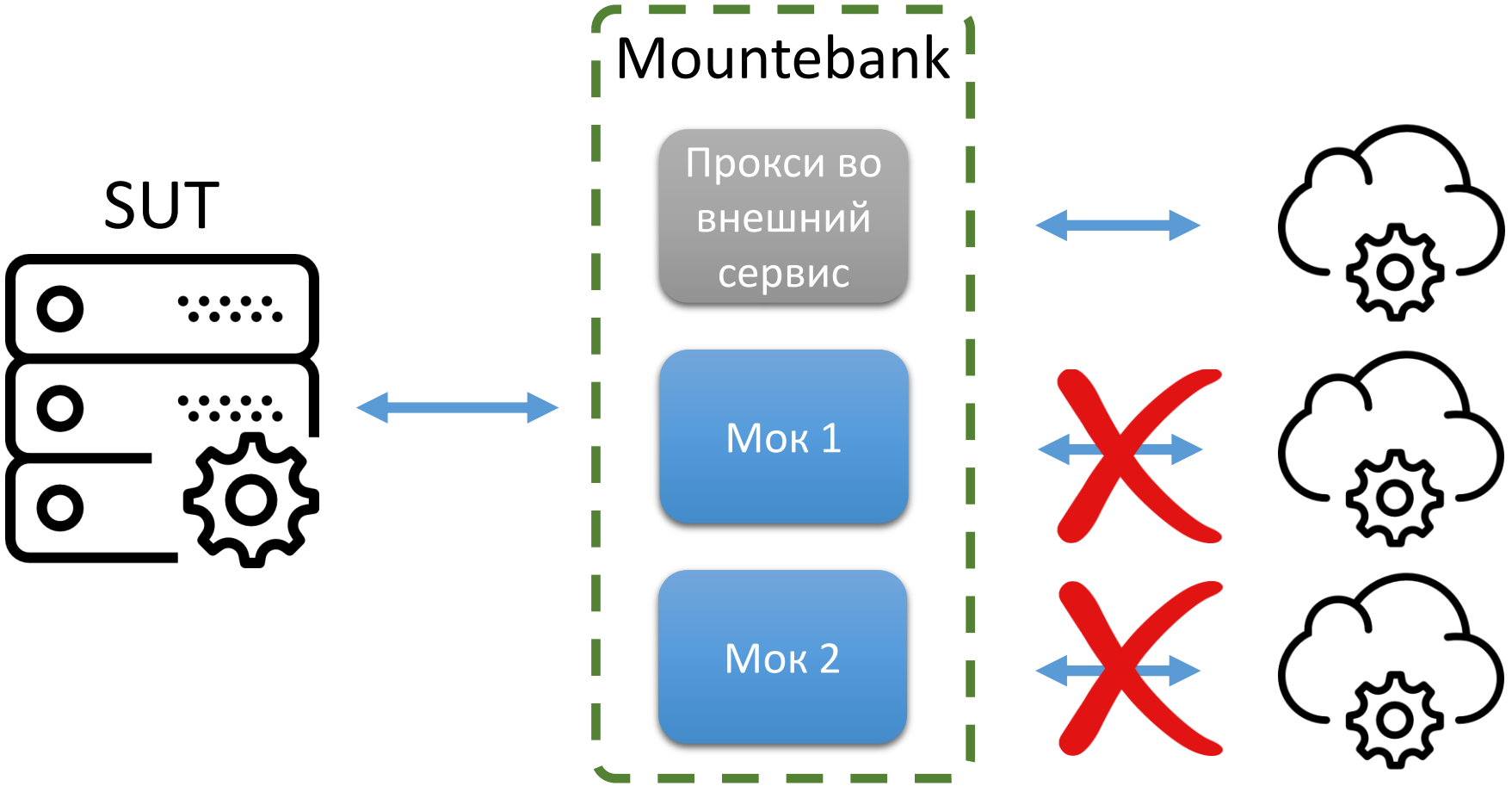
Mock-сервер: логирование запросов



Mock-сервер: изоляция внешних зависимостей



Mock-сервер: изоляция внешних зависимостей



Что ещё умеет Mountebank

- Не только HTTP/HTTPS, но и **TCP**
- JavaScript injection
- Надежность и скорость из коробки
- Отличная документация
- Поддержка докер-контейнеризации
- Это opensource :)

Чего мы добились

- Эмуляция любого поведения внешних систем
- Нет паразитного кода в приложении
- Отсутствует необходимость поддержки тестового кода фейков
- Низкий порог входа

Автотесты: сравнение подходов

Мок на уровне кода:

недостаточная вариативность

не проверяем систему целиком

Глупый фейк:

ограниченная функциональность

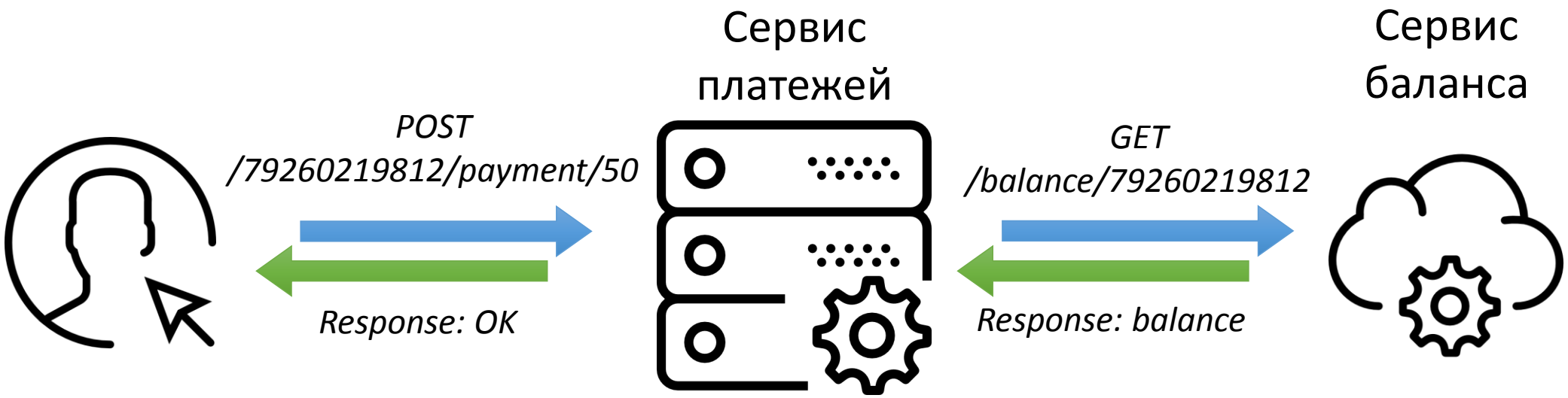
не управляется из тестов

Умный фейк:

только позитивная функциональность

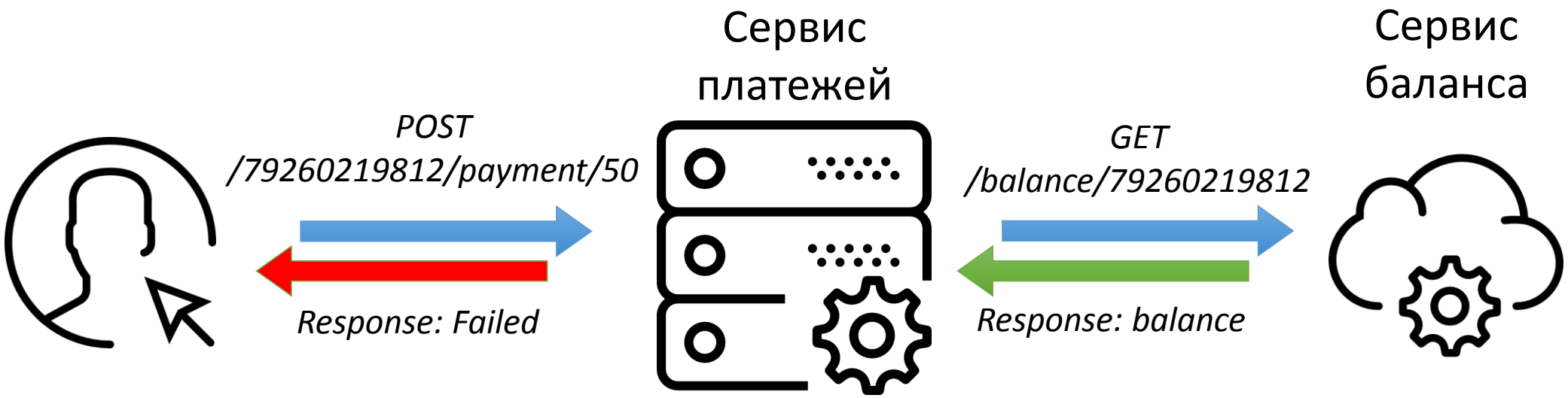
Мок-сервер (Mountebank)

Тестовый случай



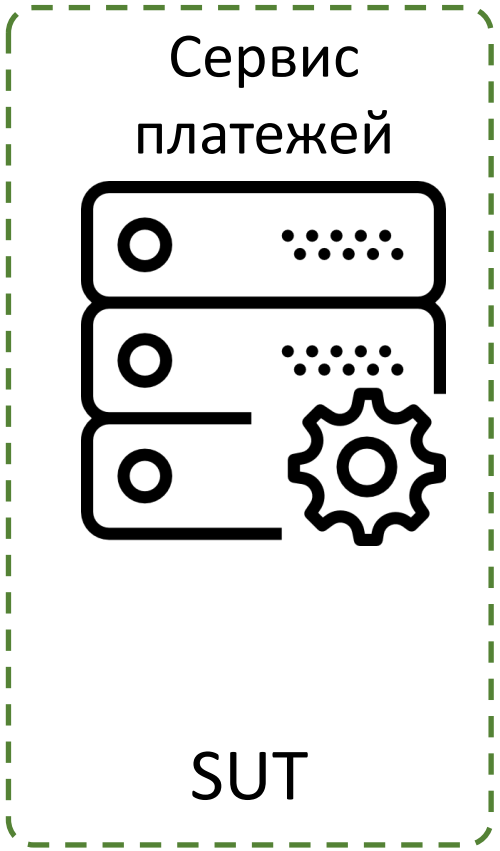
balance > payment

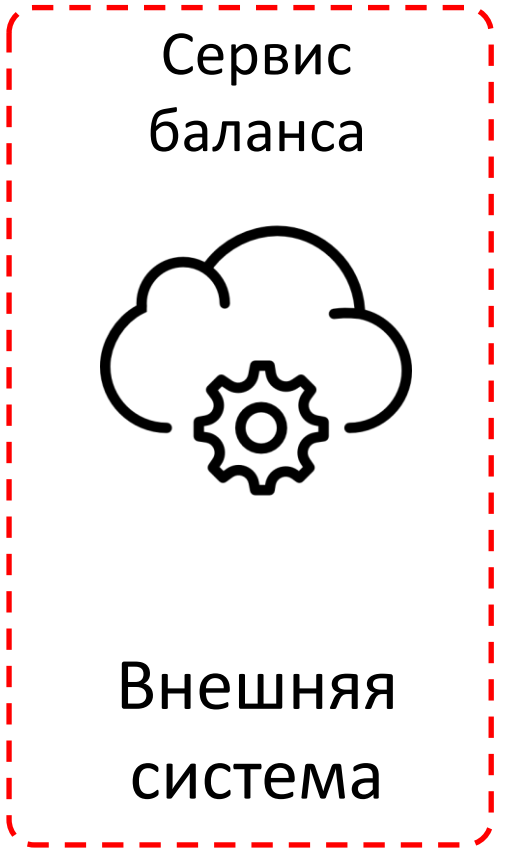
Тестовый случай



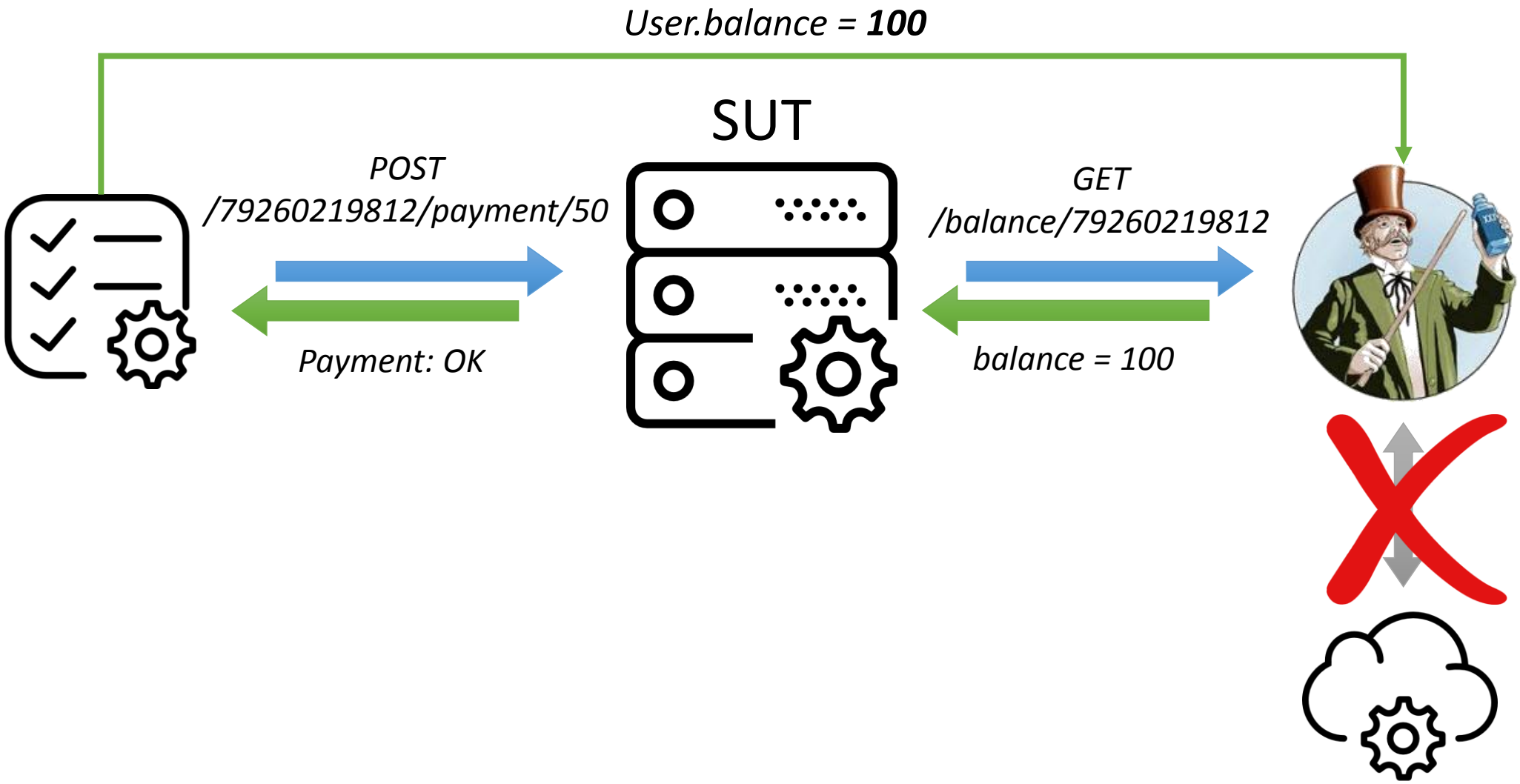
balance < payment

Тестовый случай

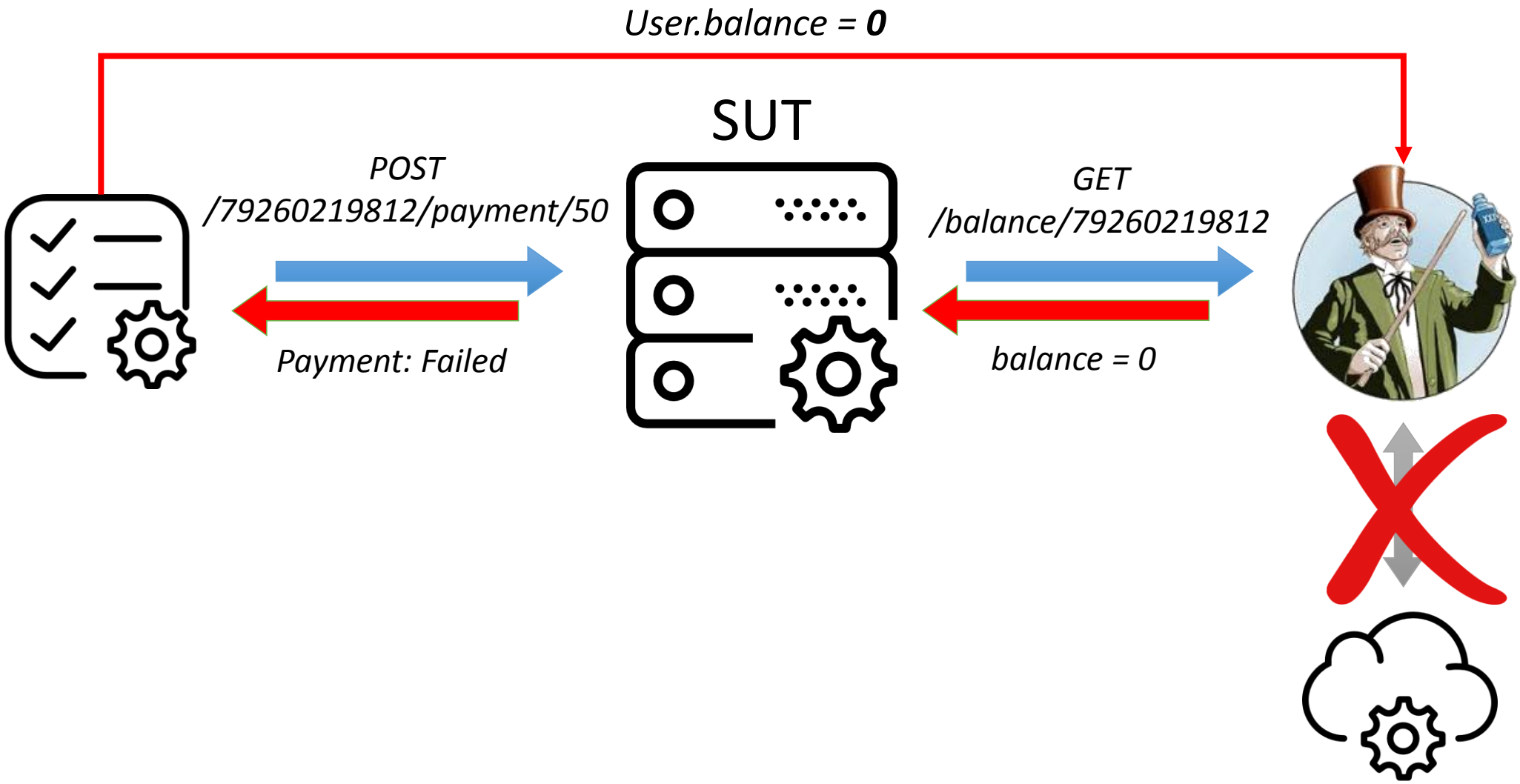




Автотесты + mock-сервер

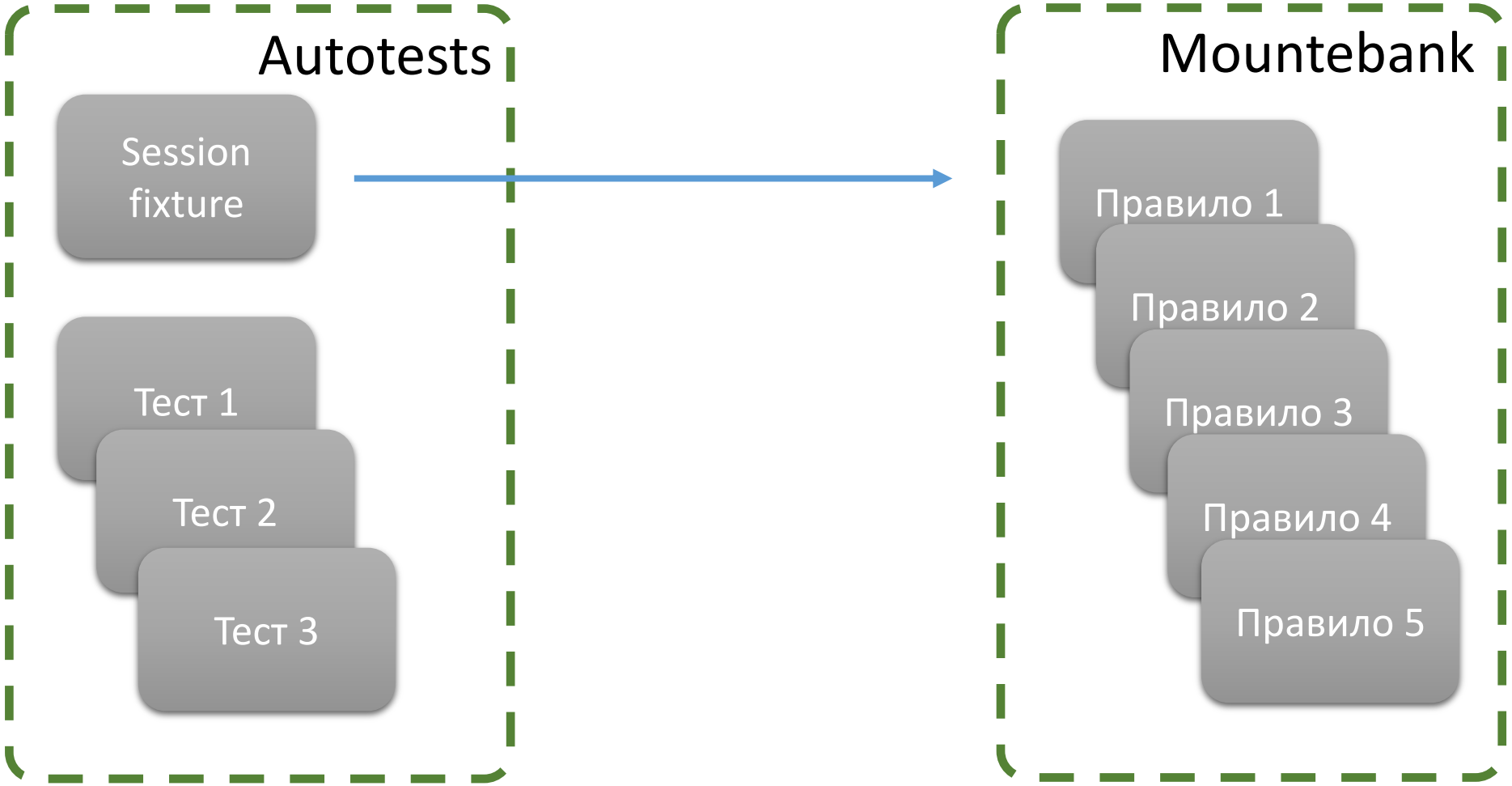


Автотесты + mock-сервер

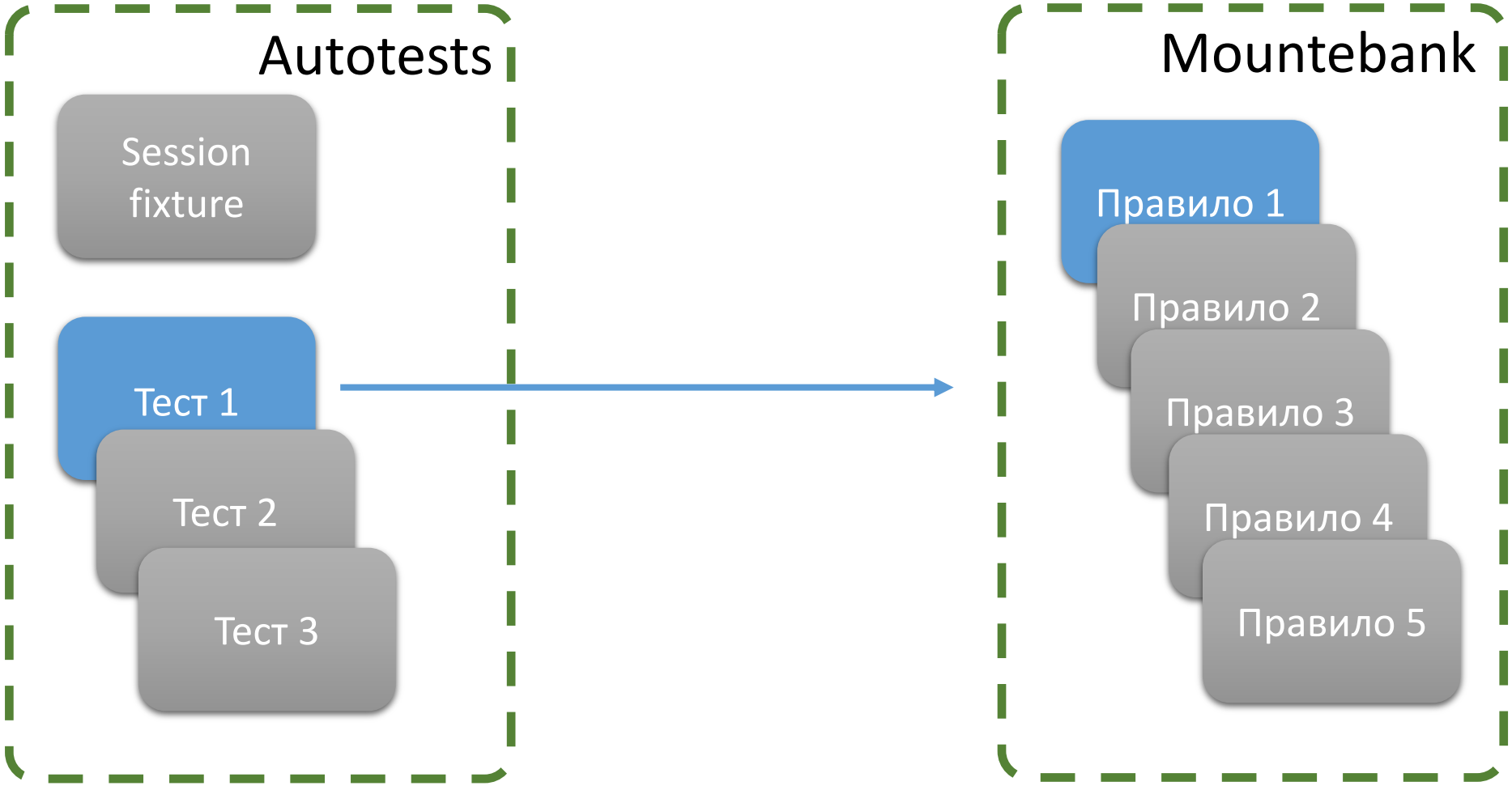



SHOW
TIME!

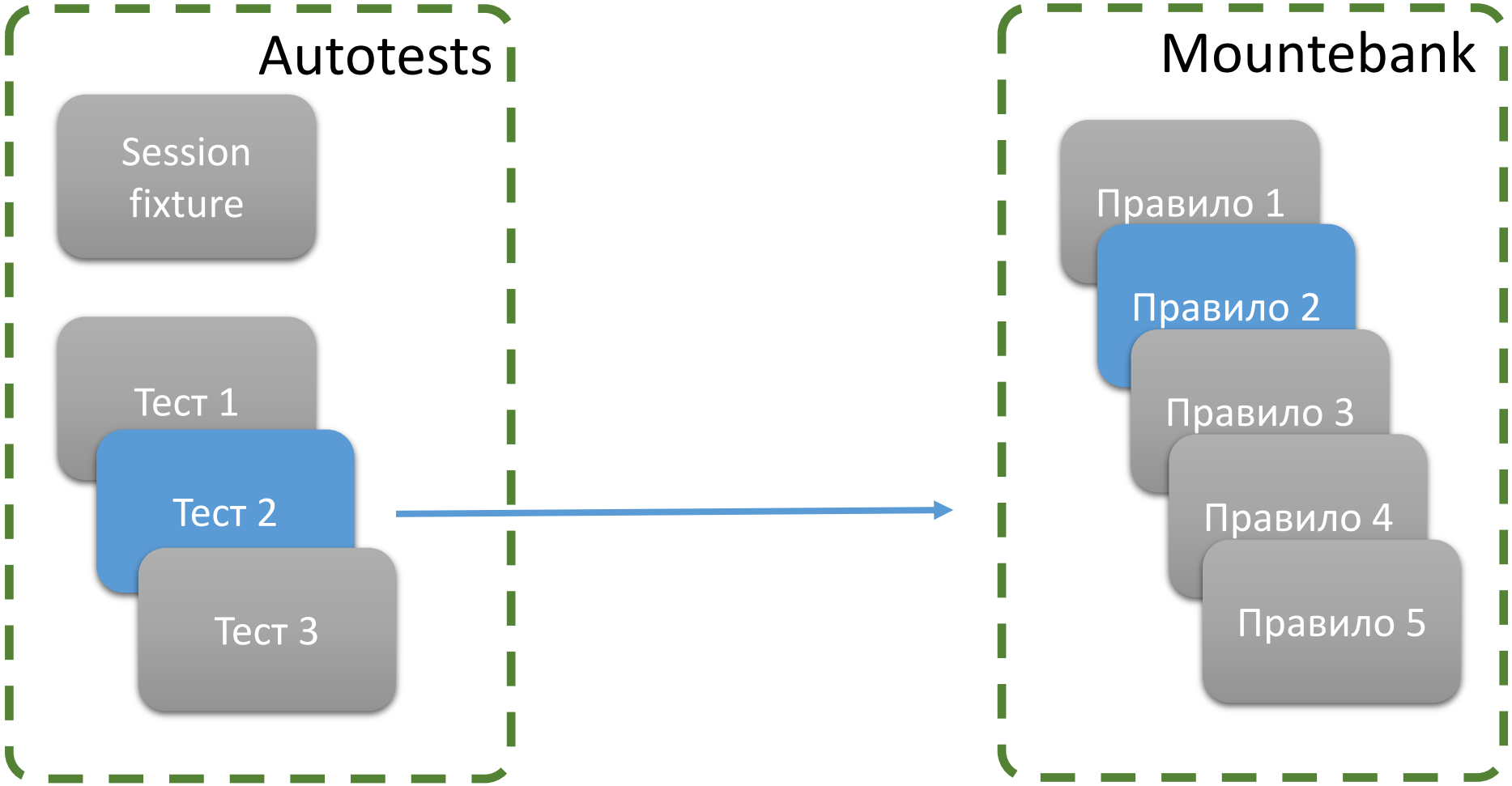
Не делайте моков «прозапас»



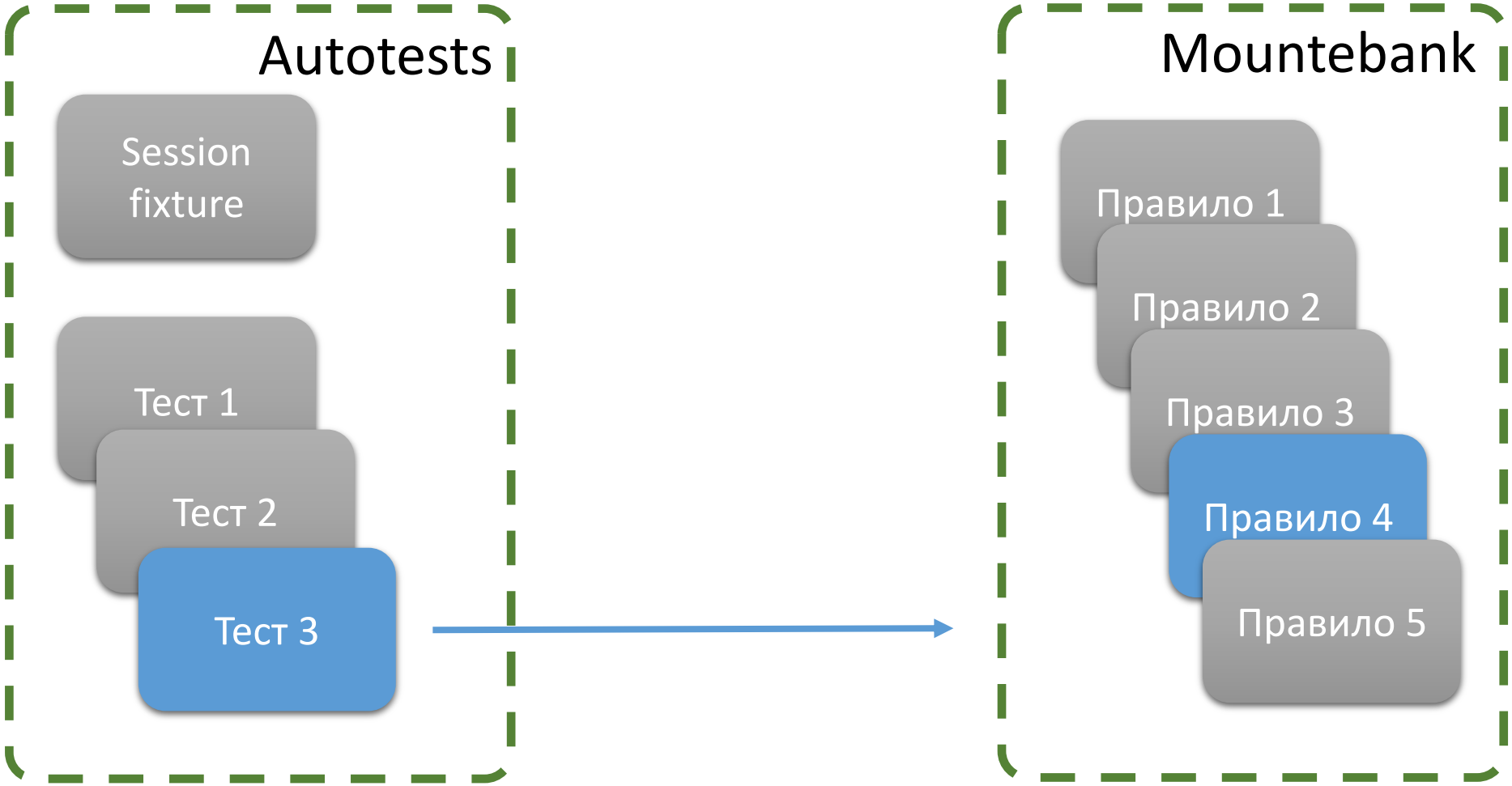
Не делайте моков «прозапас»



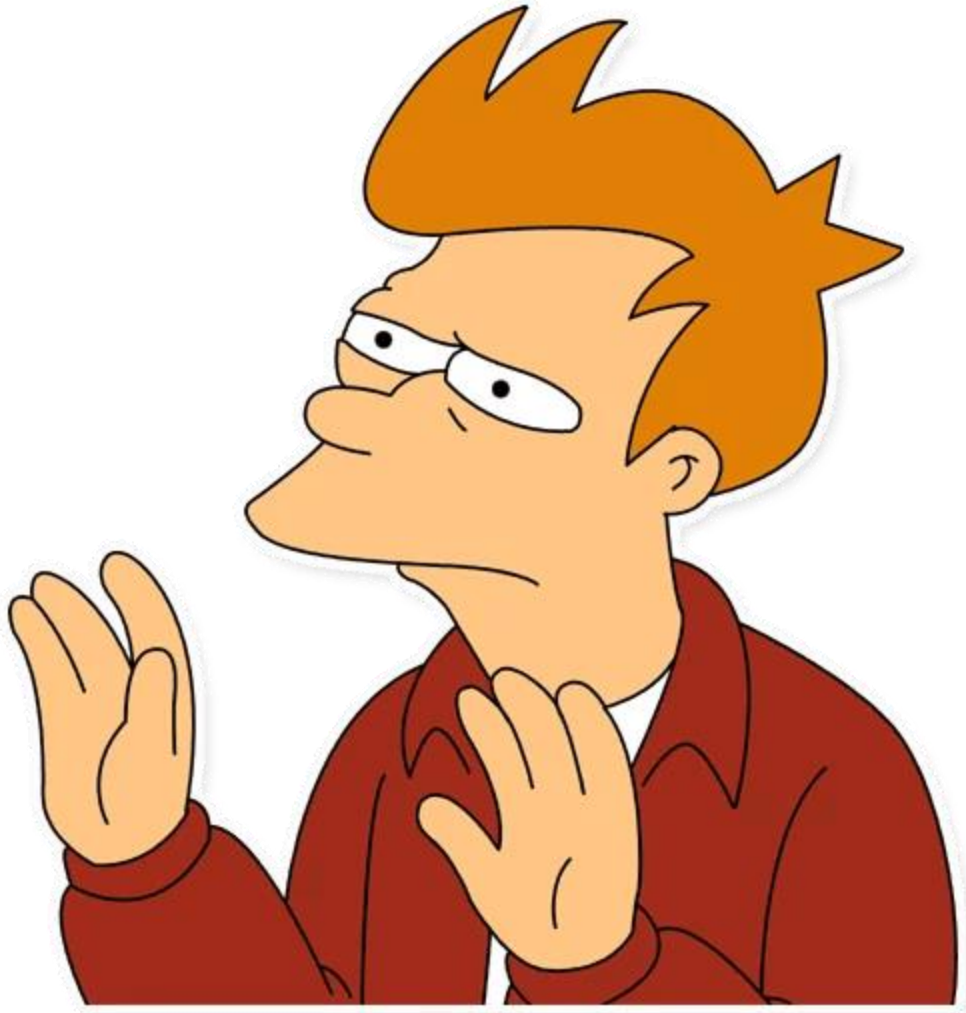
Не делайте моков «прозапас»



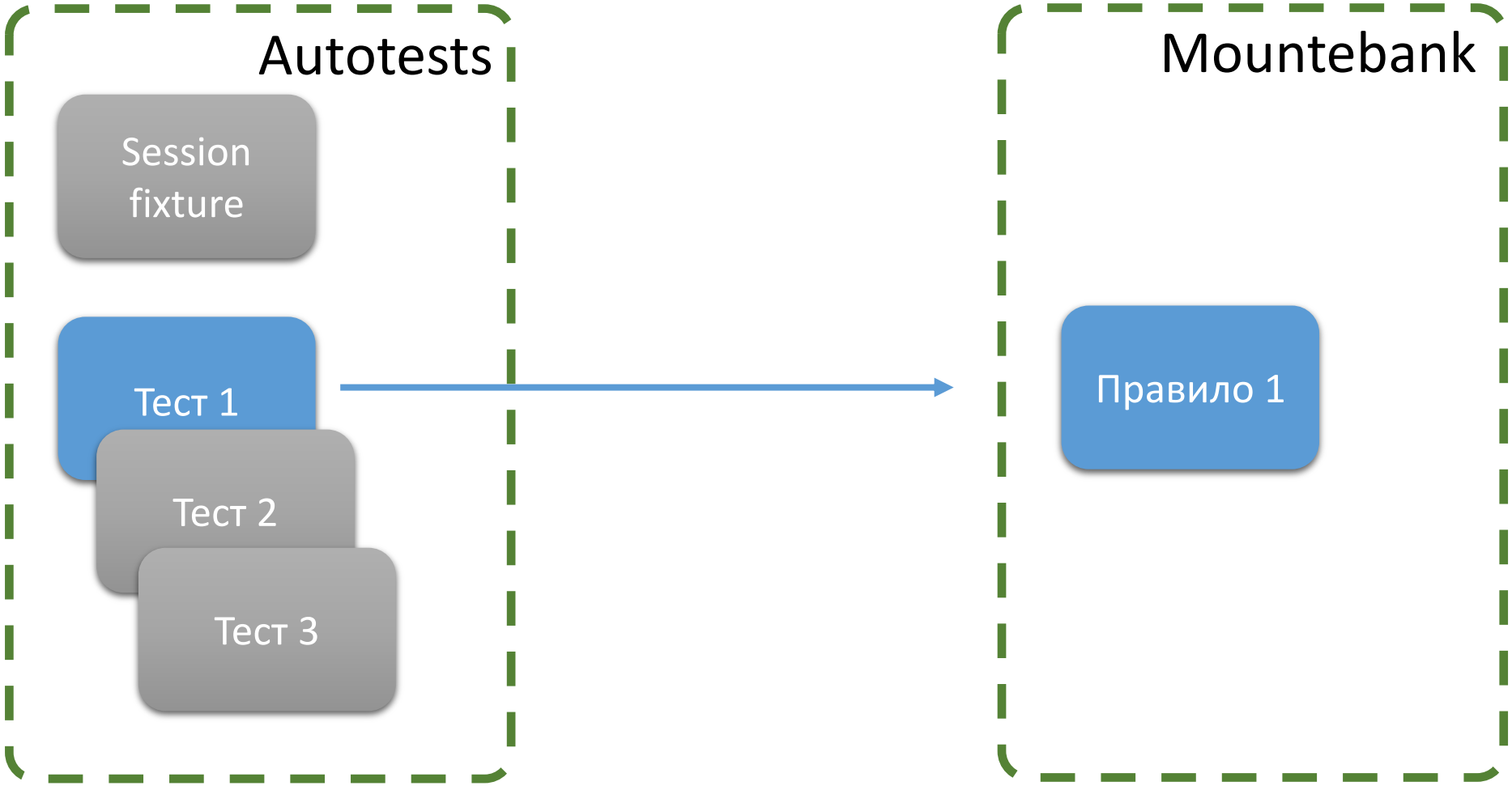
Не делайте моков «прозапас»



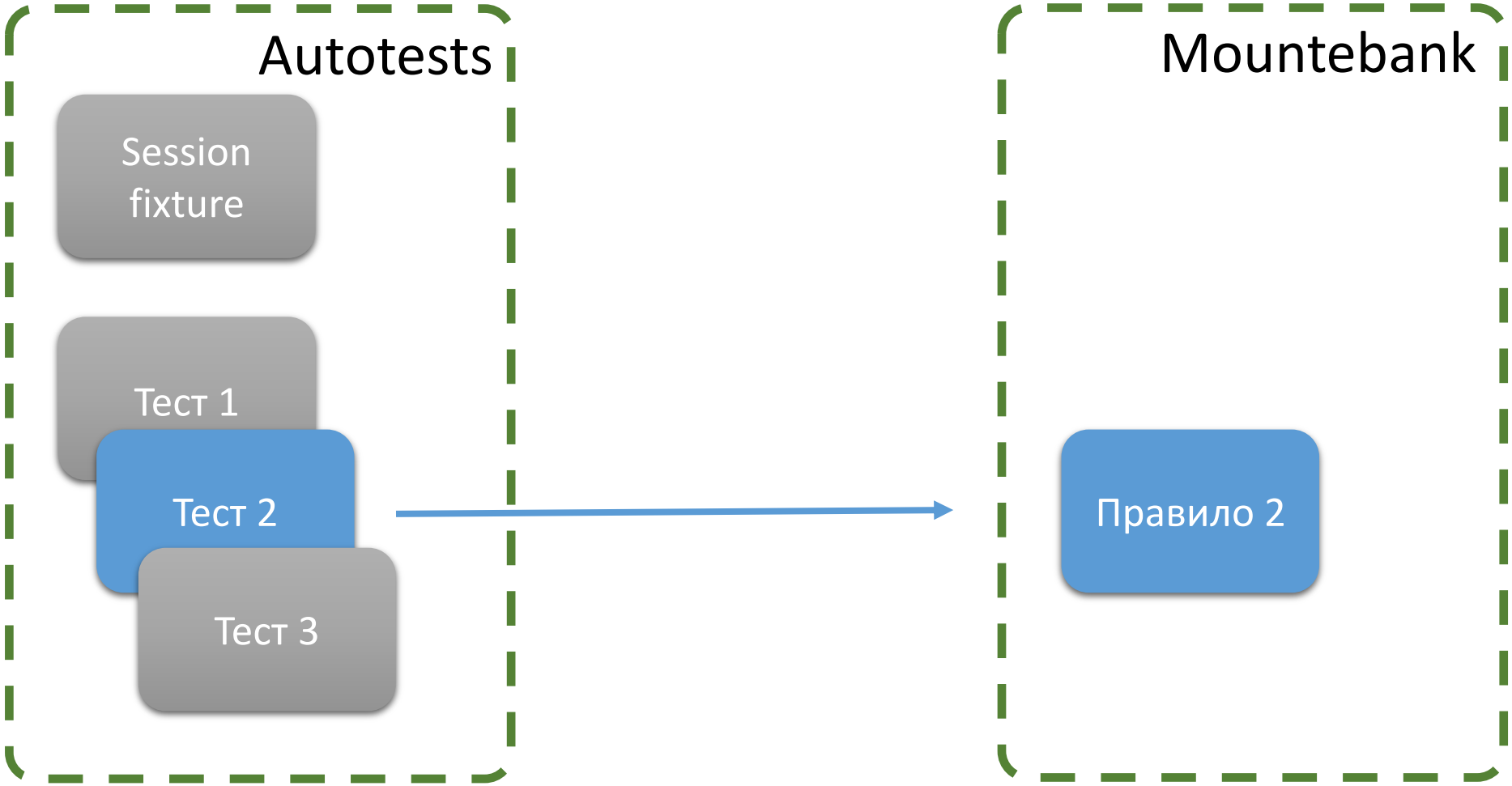
NO



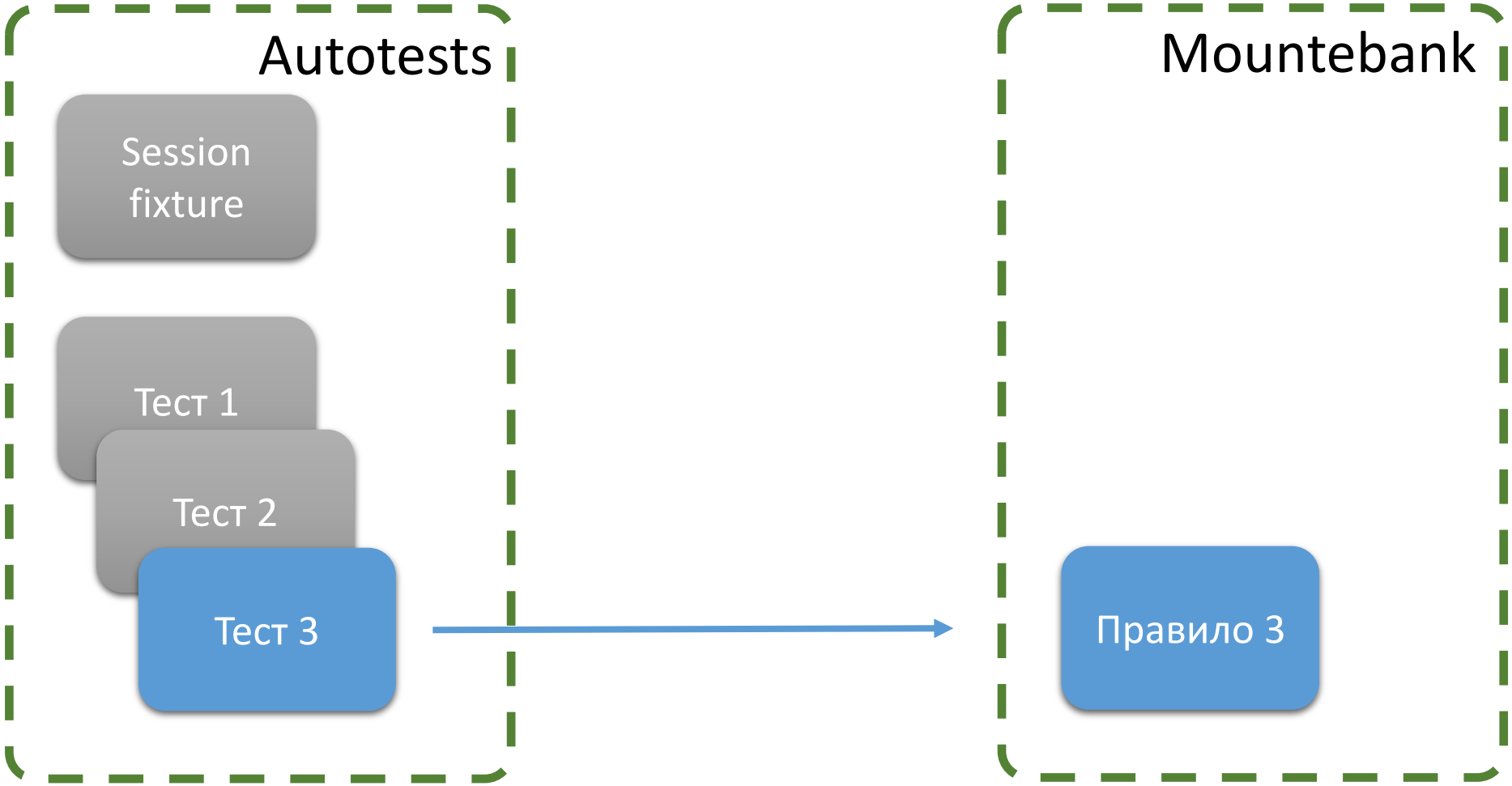
Не делайте моков «прозапас»



Не делайте моков «прозапас»



Не делайте моков «прозапас»



Не делайте моков «прозапас»

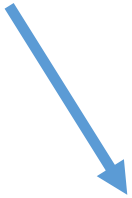
```python
@pytest.fixture
def setup_user(external_service):
    # user model
    u = User(name='Andrey', balance=500)
    external_service.update(user=u)

def test_user(setup_user):
    # ...test logic
```

YES

Мок = объект

```
@pytest.fixture(scope='session')
def external_service():
    mb = Mountebank(host='localhost')
    mock = mb.add_imposter(external_service_config)
    return mock
```



```
{
  "responses": [{
    "proxy": {
      "to": "http://external-server.com",
      "mode": "proxyTransparent"
    }
  }]
}
```

Мок = объект

Адаптеры есть для большинства популярных языков:

C#, Clojure, Delphi, F#, Go, Java, JavaScript, Perl, PHP, Python, Ruby, Shell, TypeScript

http://www.mbtest.org/docs/clientLibraries

Language	Name	Author
C#	MbDotNet	Matthew Herman
Clojure	Charlatan	Matthew Daley
Delphi	mountebank-delphi	Jamie Geddes
F#	MbDotNet.FSharp	Matthew Herman
Go	GoBank	Erkan
	mbgo	Senseye
Java	javabank	James Thomas
JavaScript	mountebank-helper	Alex
Perl	Test::Mountebank	Dagfinn Reiersøl
PHP	Juggler	Andrejs Mironovs
	mountebank-api-php	Demyanovsky Ruslan
	mountebank-php	Abraham Vallez
Python	mountepy	Michał Bultrowicz
	py-mountebank	Kevin Qiu
	mountebank-python	Alex Holyoke
	mbtest	Simon Brunning
Ruby	mountebank-gem	Michael Cheng
Shell	mountebank-sh	Sergi Bech Robleda
TypeScript	node-mountebank	Ron van der Wijngaard

Научите моки работать с тестовыми моделями

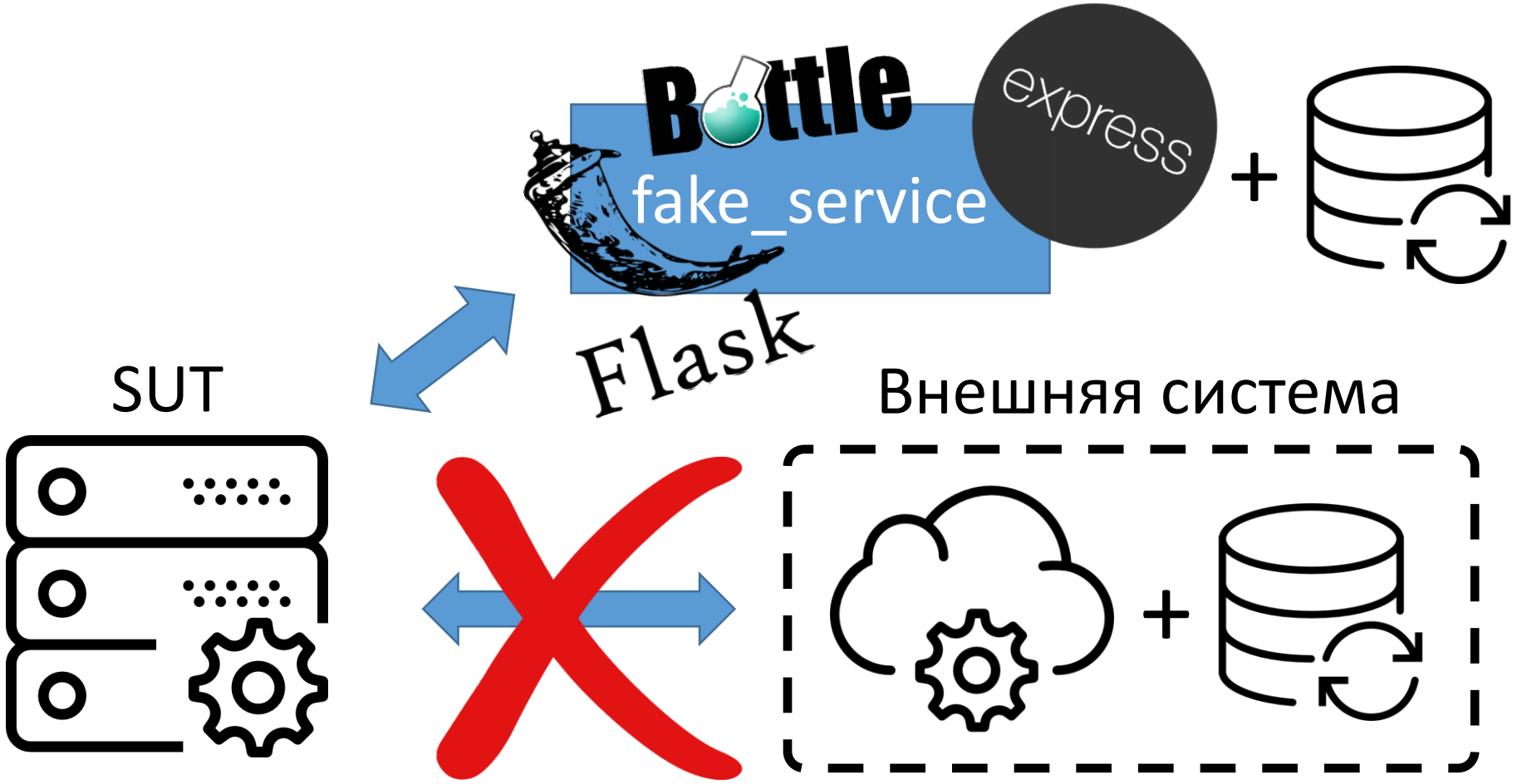
```python
@pytest.fixture
def setup_user(external_service):
    # user model
    u = User(name='Andrey', balance=500)
    external_service.update(user=u)

def test_user(setup_user):
    # ...test logic
```

Ограничения Mountebank’a

Ограничения Mountebank’a

1. Внешние системы с сохранением данных

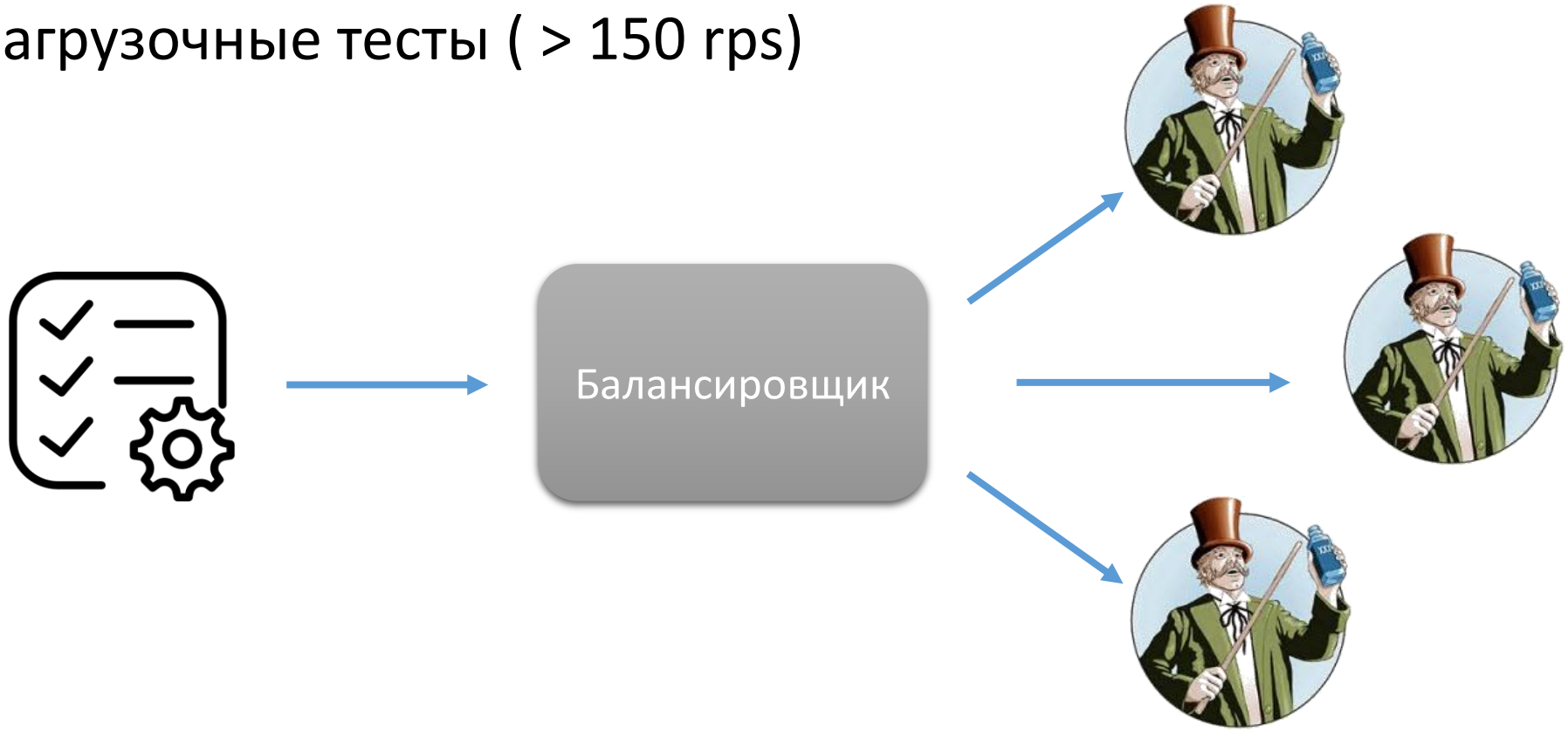


Ограничения Mountebank’a

1. Внешние системы с сохранением данных
2. WebSocket’ы

Ограничения Mountebank’a

1. Внешние системы с сохранением данных
2. WebSocket’ы
3. Нагрузочные тесты (> 150 rps)



Итоги

1. Решили проблему недоступности внешних систем
2. Научились эмулировать любое поведение внешних систем
3. Получили возможность верифицировать общение с внешними системами
4. Эффективно автоматизировали всё вышеперечисленное!

Спасибо за внимание!

Глазков Андрей
Paysystem.tech
andrewglazkov@gmail.com
@glazz87