Южно-Уральский государственный университет | Национальный исследовательский университет

Приложение для генерации мок-сервисов на основе RAML спецификации

Глеб Радченко, доцент кафедры СП, ВШ ЭКН ЮУрГУ (gleb.radchenko@susu.ru)
Андрей Черных, профессор CICESE Research Center (Ensenada, Mexico)
Никита Ашихмин, студент кафедры СП, ВШ ЭКН ЮУрГУ

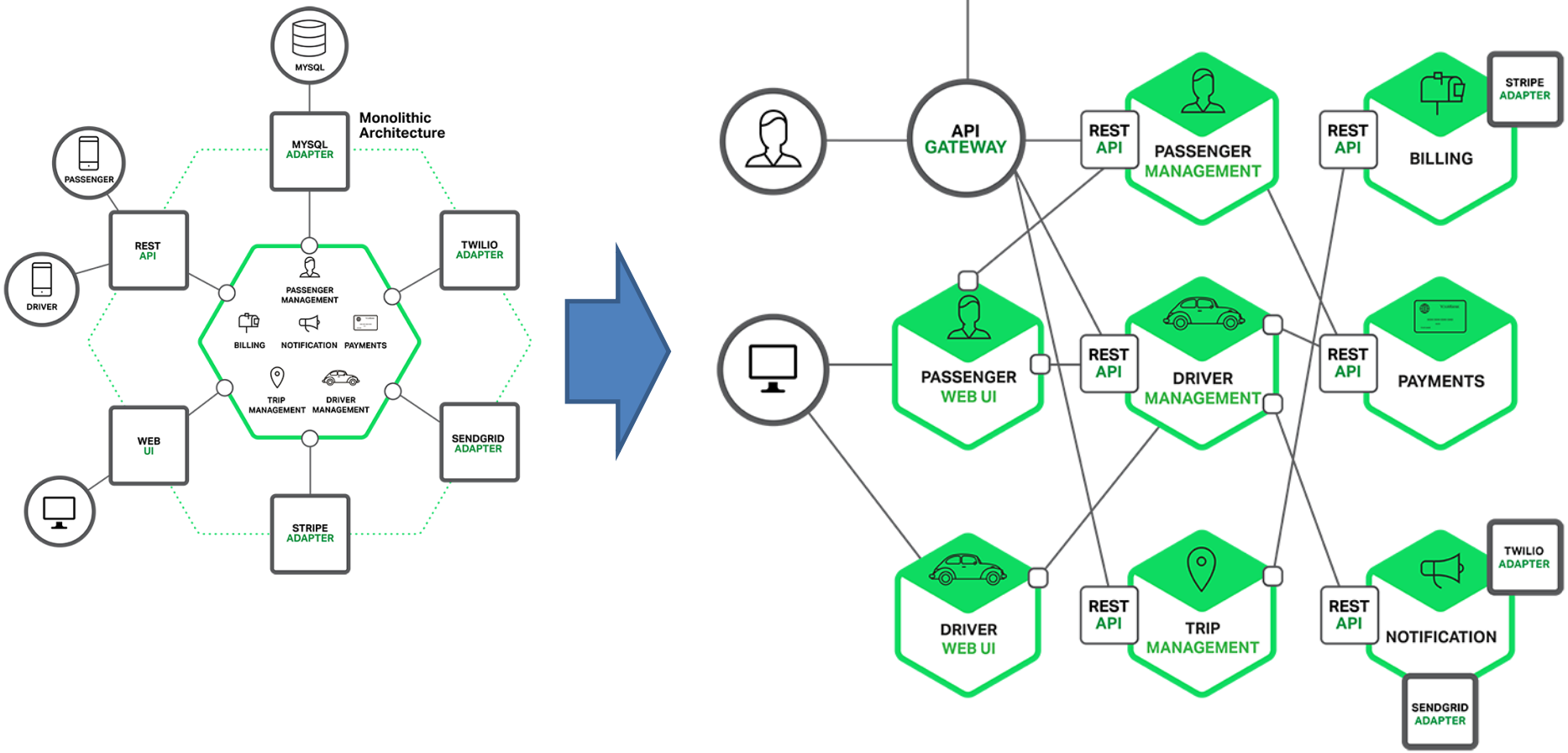
Микросервисы и все остальные

- **Микросервисная архитектура** облачных систем – это концептуальное развитие компонентно-ориентированного подхода.

- Микросервис – это независимый программный компонент, который:
  - разрабатывается, разворачивается и исполняется независимо от других компонентов системы;
  - обеспечивает прозрачный веб-доступ к своим ресурсам посредством стандартизованного интерфейса (например, REST);
  - отвечает за реализацию отдельной бизнес-задачи.

- Применяется ведущими провайдерами облачных ресурсов: Amazon, eBay, Netfix

Микровервисы



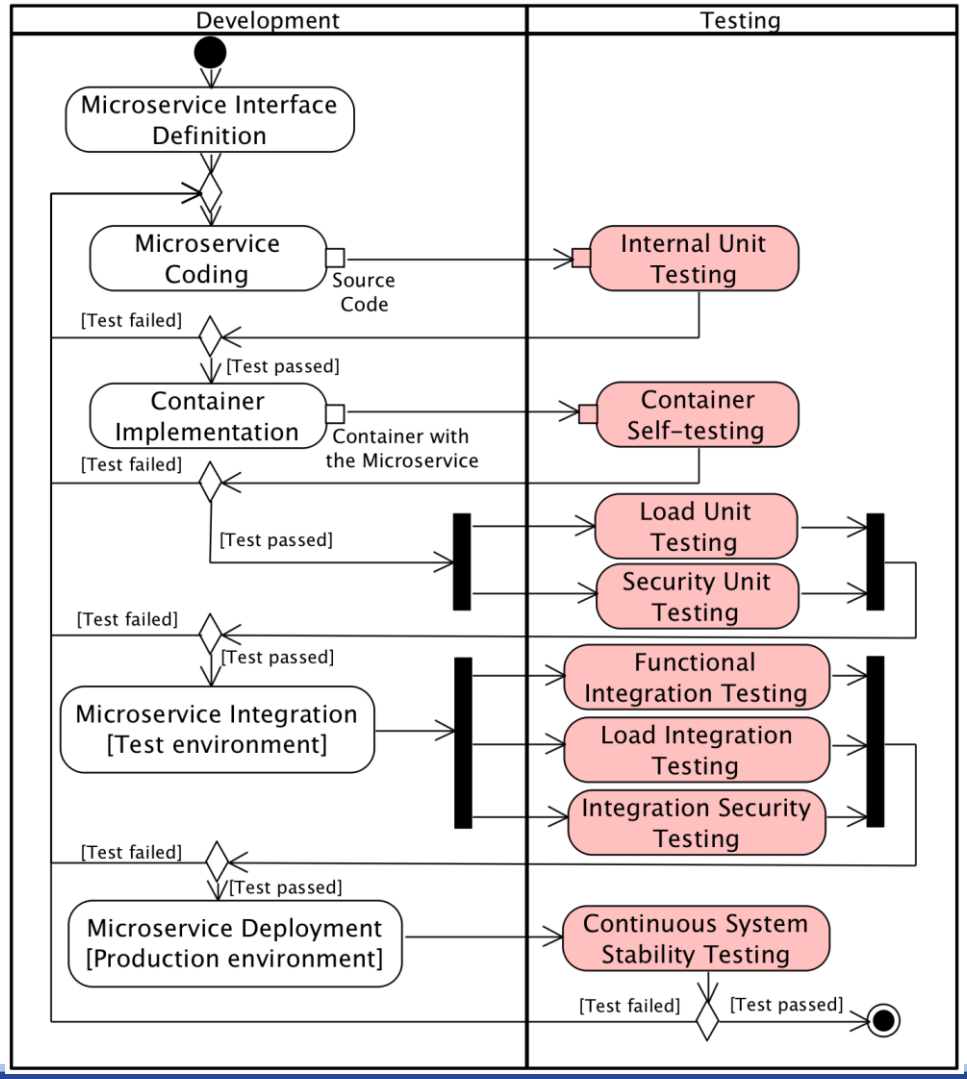
Особенности и ограничения микросервисов

- Для имплементации микросервисного подхода необходима вычислительная среда, обеспечивающая автоматическое **развертывание**, **масштабирование** и **взаимодействие** микросервисов.
- Разработка микросервисных систем невозможна без **непрерывной интеграции** разработанных и/или модифицированных микросервисов в существующую инфраструктуру.
- Что влечет необходимость в постоянном **автоматизированном тестировании** как отдельных микросервисов, так и совокупности системы в целом.

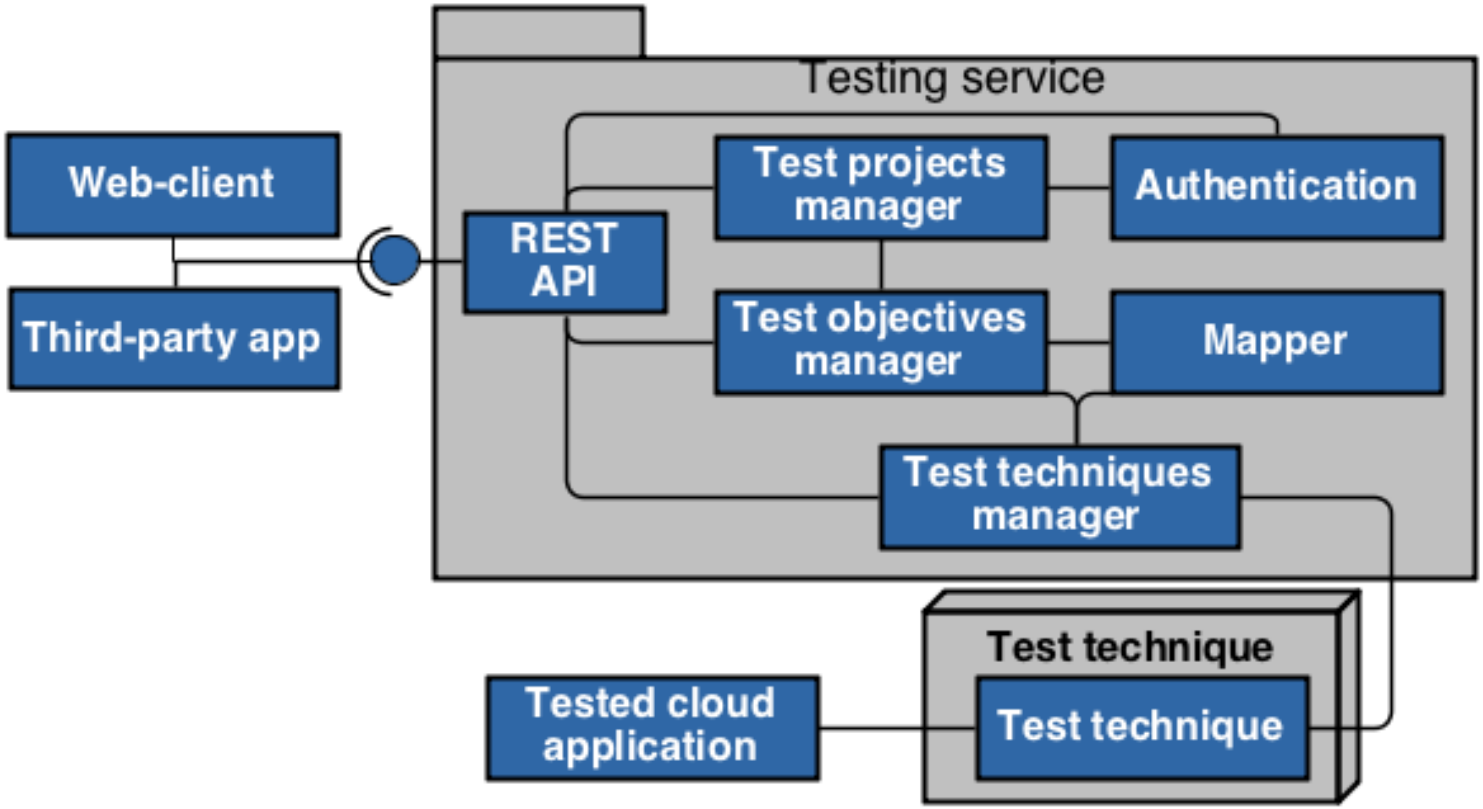
Виды тестирования

- ***Модульные тесты*** обычно проверяют один вызов функции или метода.
- ***Сервисные тесты*** предназначены для проверки интерфейса отдельных сервисов.
- ***Сквозные тесты*** проверяют корректность работы всей системы.

Тестирование микросервисов



Платформа автоматизированного тестирования микросервисов



Цель

- **Цель работы -** описать архитектуру и реализацию приложения, которое обеспечивало бы автоматическую генерацию мок-сервисов на основе RAML спецификации. Мок сервисы должны поставляться в виде докер образа с мок-сервисом внутри.

REST и RAML

- **REST** - один из наиболее распространенных подходов к реализации программного интерфейса сервиса
- **RESTful API Modeling Language** (**RAML**) это язык, основанный на YAML для описания REST сервисов. Он предоставляет задать, адреса ресурсов, возможные параметры запросов и формат ответов.

Пример RAML спецификации

```
/employees:
  get:
    queryParameters:
      department:
    responses:
      200:
        body:
          application/json:
             example: |
                ...
  post:
  delete:
  /{employee}:
   get:
     responses:
       200:
         body:
          application/json:
            schema: |
                 ...
```

Мок-сервис

- ***Мок-сервис*** – сервис, который имитирует поведение реального сервиса, возвращая http ответы, на определенные http запросы, по заранее заданным правилам.
- Мок-сервисы применяются в следующих случаях:
  - *Разработка* - в начале разработки мок-сервис может имитировать поведение сервисов, которые не были еще реализованы, тем самым решая проблему «курицы или яйца».
  - *Тестирование* – мок-сервисы позволяют тестировать каждый сервис изолированно от других. Это позволяет существенно сократить время и ресурсы, требуемые для тестирования.

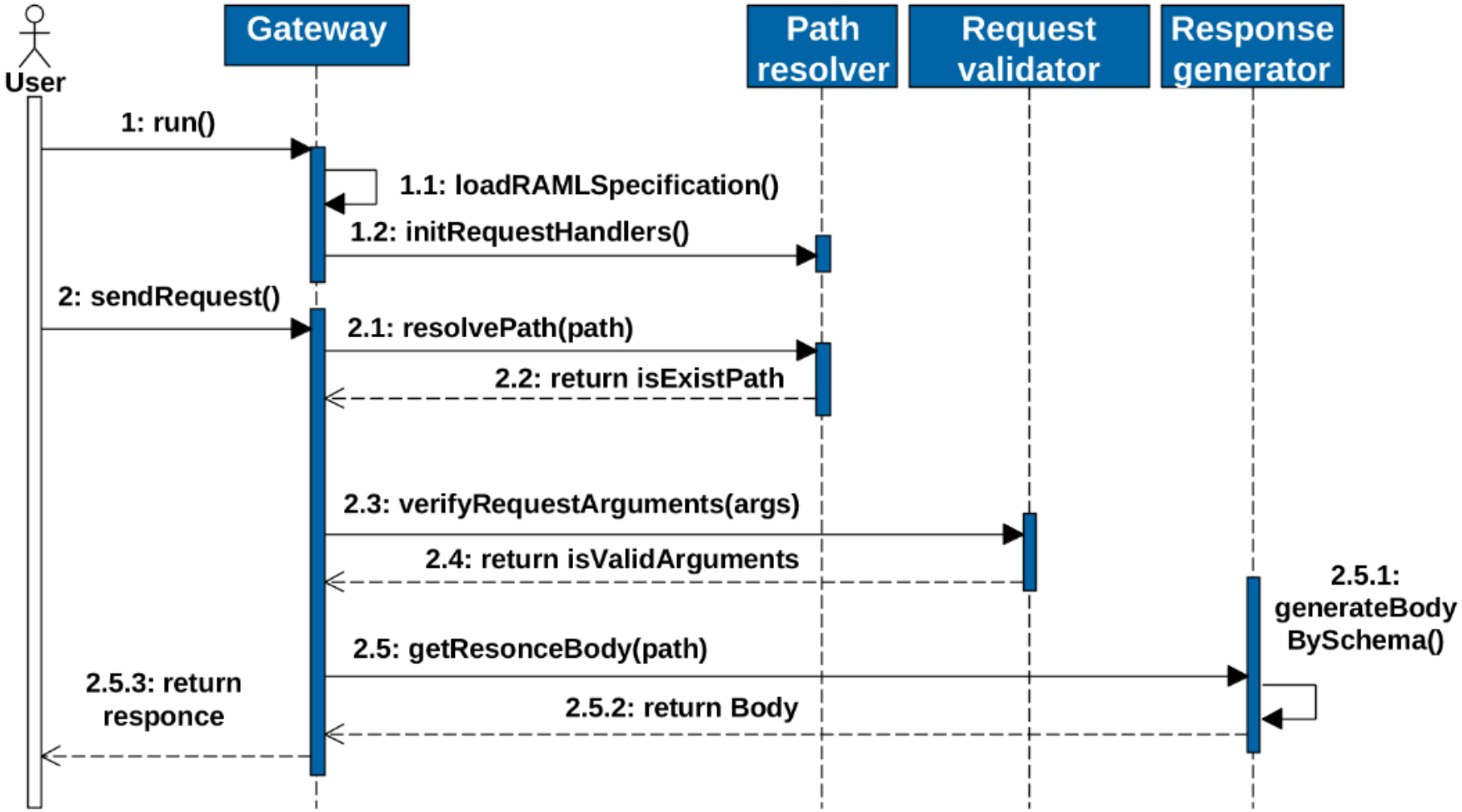
Генератор мок-сервисов

- **Mock service generator** – это независимое консольное приложение, с которым взаимодействует пользователь. Оно создает мок-сервис, упакованный в Docker image на основе RAML конфигурации пользователя

```
FROM python:3-onbuild
EXPOSE 5000
ENV PYTHONPATH .
CMD ["python", "./server/main.py"]
```

Dockerfile для генерации docker-image на основе файлов мок-сервиса

Взаимодействие пользователя с сервисом-генератором



Request Validator

- **Request Validator** – компонент, проверяющий корректность http запросов на основе спецификации, указанной в RAML файле.

```
/{documentId}
  uriParameters:
    id:
      description: document identification number
      type: string
      minLength: 20
      pattern: ^[a-zA-Z]{2}\-[0-9a-zA-Z]{3}\-\d{2}$
```

Пример спецификации http запроса

Response generator

- ***Response generator*** – компонент, который генерирует ответ на основе JSON Schema, если она указана. В противном случае, компонент возвращает пример ответа, описанного в RAML файле.
- При формировании ответа, используется библиотека Mimesis (https://github.com/lk-geimfari/mimesis), обеспечивающая наполнение фиктивными данными ответных сообщений от REST-сервиса

Пример работы системы

request:

```
GET http://127.0.0.1:5000/employees/1
```

response:

```
{
"fullName":"John Smith",
"department": "wood",
"email": "smith-toronto-1765@gmail.com"
}
```

Результаты

- Предложена архитектура приложения для генерации мок-сервисов.
- Оно обеспечивает создание мок-сервисов на основе спецификации RAML в виде докер-контейнеров.
- Исходный код приложения доступен на нашей странице GitHub

https://github.com/veor12/raml-mock-service