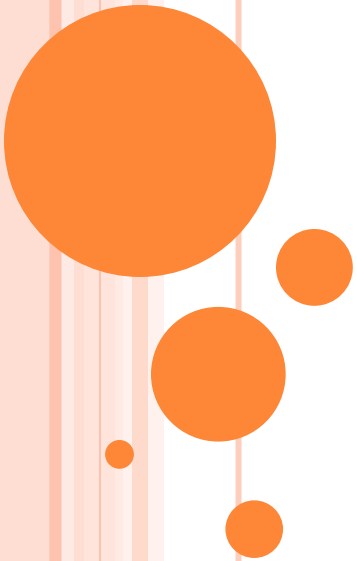


Deploying REST Applications

COMP 30220: Distributed Systems

Lecturer: Rem Collier

Email: rem.collier@ucd.ie

REST Contracts

- Contracts are a key feature of REST.
- A **contract** is a clear and unambiguous definition of the REST services.
  - What data entities exist.
  - What endpoints exist.
  - What actions can be performed on those endpoints.
  - What outcome to expect when performing an action on an endpoint.
- Without a contract, it becomes difficult to understand how to interact with a service.
- In SOAP, the contract is specified using WSDL

REST Contract Languages

- Google Discovery service format
  - https://developers.google.com/discovery/
- Mashery IO Documents
  - https://www.mashery.com/
- Apiery
  - https://apiary.io/
- Swagger
  - https://swagger.io/

Swagger.io

- Swagger is a definition format that allows developers to describe their RESTful APIs.
  - Swagger is implementation independent
  - Swagger definitions can be validated
  - Swagger is both human and machine readable
  - Swagger can be used to automatically generate code for a number of target environments.

- Swagger underpins the **Open API initiative**, which aims to create a single open standard for specifying RESTful APIs.
  - https://www.openapis.org/

Swagger.io

- Swagger was initially based on YAML (Yet Another Markup Language), but it can also be written in JSON.

```
openapi: 3.0.0
info:
  title: Sample API
  description: Optional multiline or single-line description in [CommonMark](http://commonmark.org/help/) or HTML.
  version: 0.1.9


servers:
  - url: http://api.example.com/v1
    description: Optional server description, e.g. Main (production) server
  - url: http://staging-api.example.com
    description: Optional server description, e.g. Internal staging server for testing
```

Swagger.io

- Swagger was initially based on YAML (Yet Another Markup Language), but it can also be written in JSON.

```
paths:
  /users:
    get:
      summary: Returns a list of users.
      description: Optional extended description in CommonMark or HTML.
      responses:
        '200':    # status code
          description: A JSON array of user names
          content:
            'application/json':
              schema:
                type: array
                items:
                  type: string
```

Swagger & HelloWorld

```
openapi: 3.0.0
info:
  title: Hello World API
  version: '0.1'
paths:
  /hello/{code}
    get:
      summary: Returns hello world in a language
      parameters:
        - name: code
          in: path
          required: true
          description: country code
          schema:
            type: string
      responses:
        '200':
          description: OK
          content:
            plain/text:
              schema:
                type: string
```

Swagger (JSON)

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "Hello World API",
    "version": "0.1“
  }, "paths": {
    "/hello/{code}": {
       …
    }
  }
}
```

Swagger (JSON)

```
"get": {
  "summary": "Returns hello world in a language",
  "parameters": [
    {
      "name": "code",
      "in": "path",
      "required": true,
      "description": "country code",
      "schema": { "type": "string‟ }
    }
  ],
  "responses": {
    "200": {
      "description": "OK",
      "content": { "plain/text": { "schema": { "type": "string‟ } } }
}
  }
}
```

Swagger: Code Generation

- Need to use older version of swagger (hello.yml):

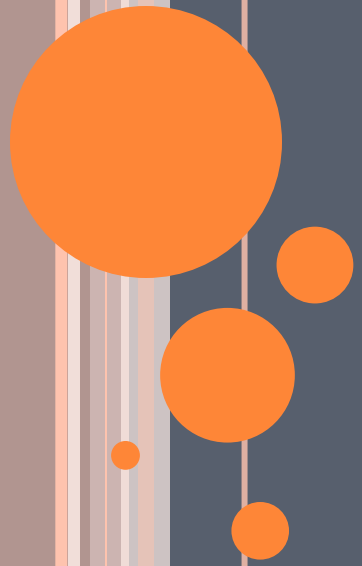
```
swagger: '2.0'
info:
  title: Hello World API
  version: '0.1'
paths:
  '/hello/{code}':
    get:
      summary: Returns hello world in a language
      parameters:
        - name: code
          in: path
          required: true
          description: country code
          type: string
      responses:
        200:
          description: OK
          schema:
            type: string
            description: some text
```

Swagger: Code Generation

- Download the code generator:
  - https://swagger.io/docs/swagger-tools/#installation-11
- Put the code generator and the swagger specification file in the same folder, and run:
  - `java -jar swagger-codegen-cli-2.2.1.jar generate -i hello.yml -l jaxrs`
- This will create a source folder with a full Jax-RS deployment.
- All you need to do is edit the relevant implementation:
  - See `\src\main\java\io\swagger\api\impl\HelloApiServiceImpl.java`

Swagger: Code Generation

```
public class HelloApiServiceImpl extends HelloApiService {
    @Override
    public Response helloCodeGet(String code,
         SecurityContext securityContext) throws NotFoundException {
        // do some magic!
        return Response.ok().entity(
            new ApiResponseMessage(
                ApiResponseMessage.OK,
                "magic!“
            )).build();
    }
}
```



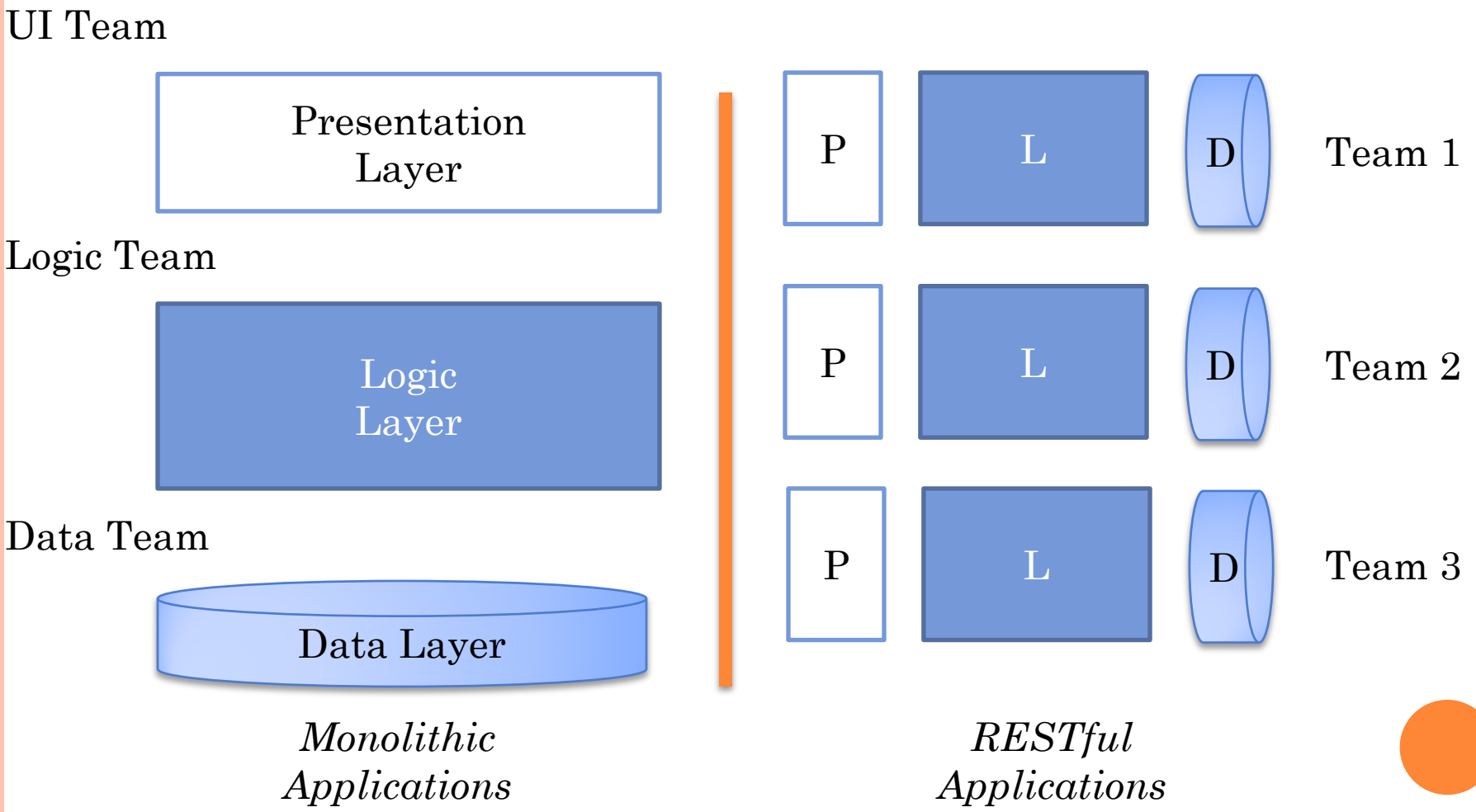
REST and Architectural Design

The Power of Rest

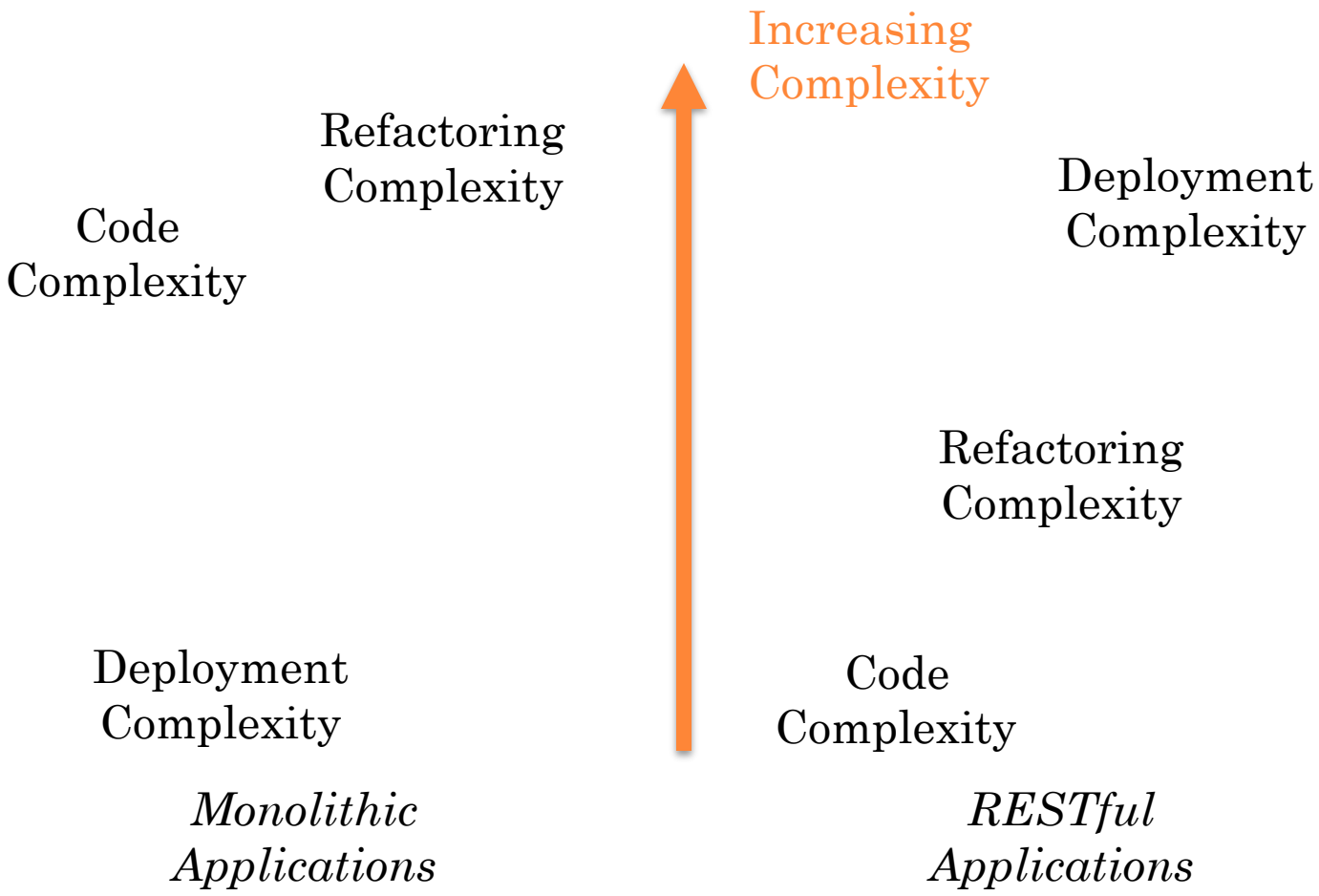
- REST is simpler to understand (it is HTTP)
  - No need to understand WSDL, SOAP, IDL, …
- REST is independent of data format
  - You can choose what format to use, or even support multiple formats JSON, XML, …

    http://localhost:8080/students?format=json
- REST services are loosely coupled
  - the interface is separated from both the implementation ***and*** the deployment.
- REST services are typically small
  - They promote a decentralised architecture

Trends in Development



Trends in Development



Micro Services

- Microservices are:

  "an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API"

  Martin Fowler, 2014

- Microservices facilitate:
  - **Strong Module Boundaries**: Clear lines demarking what is an what is not part of a service.
  - **Independent Deployment**: Simple Services are easier to deploy, and since they are autonomous, are less likely to cause system failures when they go wrong
  - **Technology Diversity**: Enables mixing of programming languages, frameworks, data storage techniques.

Micro Services

- Microservices are:

  "an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API"

  Martin Fowler, 2014

- Microservices result in:
  - **Distribution**: Distributed systems are harder to program, since remote calls are slow and always at risk of failure.
  - **Eventual Consistency**: Strong consistency becomes impossible to maintain. Instead we must live with eventual consistency.
  - **Operational Complexity**: More services means more work for the operations team. Embracing of automation and good monitoring services are essential for success.

Further Reading

- Lewis, J. and Fowler, M., Microservices: a definition of this new architectural term, *URL: https://martinfowler.com/articles/microservices.html*