

# Table of contents

## Table of contents

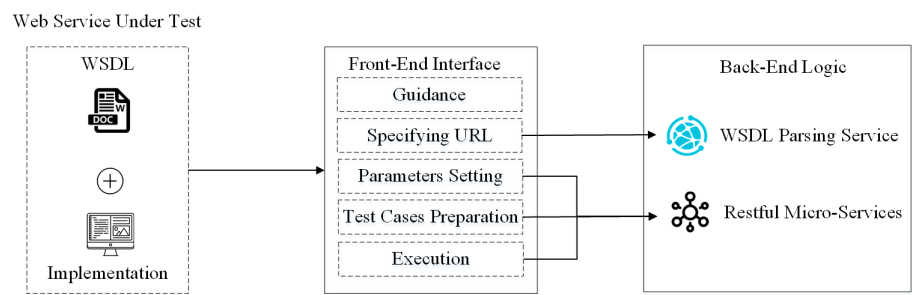
### DRTester

- Framework
  - Front-end interface
  - Back-end logic
    - WSDL parsing service
    - Restful Micro-Services
- Configuration
  - Configuration for front-end interface
  - Configuration for back-end logic
- An example
  - The specification of web service under test
  - Step 1: Specifying url and setting parameters
  - Step 2: Partition construction and parameter setting
  - Step 3: Test case preparation
  - Step 4: Test case execution

# DRTester

The prototype tool *DRTester* supports dynamic random testing technique for web service testing. We describe the implementation and usage of the tool in detail.

## Framework



The above figure illustrates the *DRTester* framework comprising four main components, corresponding to web service under test; the *interface* between the user and *DRTester*; WSDL parsing service is responsible for parsing WSDL file of web service under test; and Restful micro-services that are used to divide input domain, generate test cases, execute test case, and send information to the *interface*.

We next examine each component in the framework individually.

## Front-end interface

We developed three HTML pages by using the Vue framework (<https://cn.vuejs.org/>), their source codes are available by visiting <https://github.com/phantomDai/DRTester.git>.

This *interface* wraps the setting information in the HTTP messages, and sends them to several Restful micro-services that are not only responsible for communicating with this *interface* but also wrap the selected test cases in SOAP messages, and send them to the web service under test.

## Back-end logic

In this section, we describe the implementation of back-end logic, which comprises two parts: 1) a WSDL parsing web service; 2) several Restful micro-services.

### WSDL parsing service

We obtain the necessary information by parsing WSDL file of web service under test to generate test cases and automatically invoke interested methods of web service under test. Accordingly, a web service has been developed to acquire information about the interested methods of web service under test (such as names and types of return value), along with their parameters information (names and types). Besides, we also made this web service publicly accessible (<https://github.com/phantomDai/parseesdlws.git>).

### Restful Micro-Services

The Restful micro-services (For more details, please visit linkage: <https://github.com/phantomDai/drtAPI.git>) are responsible for communicating HTTP messages to and from the front-end interface. Besides, these micro-services need to update the test profile according to the test results, and select test cases from the partitions. The selected test cases then are wrapped in SOAP messages and sent to the web service under test through the proxy class.

## Configuration

This section describes the configuration of the front-end interface and back-end logic.

### Configuration for front-end interface

The users need to set up the local environment as follows:

1. download and install *node.js* (please visit linkage: <https://nodejs.org/en/>)
2. execute the following command in DOS (if not in China, please ignore this step):

```
npm install -g cnpm --  
registry=https://registry.npm.taobao.org
```

3. execute the following command in DOS:

```
npm install vue -g
```

4. execute the following command in DOS:

```
npm install vue-cli -g
```

After the front-end environment is configured, the user downloads the source codes by visiting the linkage: <https://github.com/phantomDai/DRTester.git>. Next, the user needs to go to the root directory of the downloaded file, and create a directory named "node\_modules". Then the user needs to execute the following command in DOS.

```
npm install (if in China, please execute command: cnpm install)
```

Finally, the user needs to find all of the *post* and *get* methods in *BaseTable.vue* and *Tabs.vue*, and change the value of *url* (that is a variable in *BaseTable.vue* and *Tabs.vue*) to replace the IP address with user's IP. For instance, the user replaces the current value of *url* (on the line 266 of *BaseTable.vue*)

```
url: 'http://202.204.62.171:8082/api/parse/wsd1' (current  
IP address)
```

with

```
url: 'http://xxx.xxx.xxx.xxx:8080/api/parse/wsd1' (user's  
IP address)
```

Finally, the user executes the following command in DOS and inputs "<http://localhost:8080>" in her browser.

```
npm run dev
```

*Guidance* is the first page of front-end interface (as shown in following figure), where we describe the steps and rules the tester should follow when testing a web service.

## Guidance

### Table of Contents

- Overview
- WSDL Parsing
- Parameters Setting
- Partition Construction and DRT Parameter Setting
- Test Cases Preparation

### Overview

Considering the principle of DRT and the features of web services, we propose a DRT for web services framework, as illustrated in blow. In the figure, the DRT components are inside the testing box, the practitioner interaction is represented in the initialization box , and the web services under test are located outside. Interactions between DRT components, practitioner and the web services are depicted in the framework.

The second page of front-end interface is *Configuration* (as shown in following figures), where the user needs to provide some information to partition input domain and generate test cases.

Q Specifying URL

Please enter the address of the web service under test

Address

Parse

P Parameters Setting

Please select an operator:

Operation

Index	Parameter	Type	Options
Empty			

Save

Partition Construction and Parameter Setting

Please input option combinations for partition construction and set parameters for DRT:

Partition	Option Combination	Test Profile	Adjusting Factor
partition	<input type="text" value="choices"/>	<input type="text" value="profile"/>	<input type="text" value="value"/>

+ Add

− Delete

✔ Save

Test Cases Preparation

Please select a method to generate a test suite:

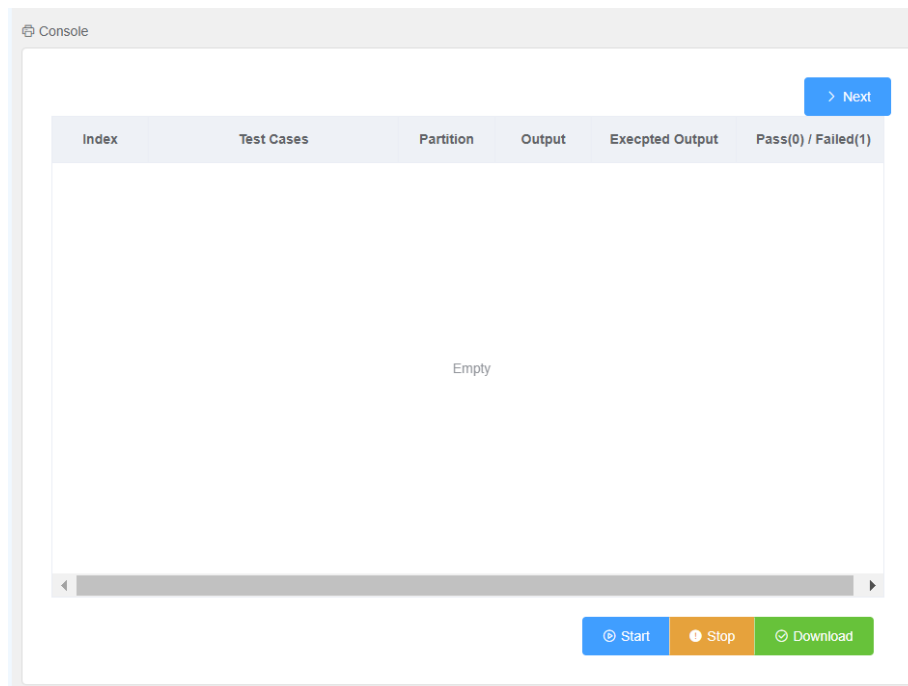
☒ Randomly Generate Test Suite

☐ Upload Test Suite File

Please set the number of test cases to be generated:

Please upload an XML file that contains test cases:

The third page of front-end interface is *Execution* (as shown in the following figures), where the user controls the execution of test cases and downloads test report.



## Configuration for back-end logic

The user needs to set up the local environment as follows:

1. Tomcat 9.06 (that is available in the repository: <https://github.com/phantomDai/parseesdlws.git>)
2. JDK 1.8.0\_161 (that is available in the linkage: <https://www.oracle.com/technetwork/java/javase/downloads/index.html>)
3. IntelliJ IDEA (that is available in the linkage: <http://www.jetbrains.com/>)

the source codes of WSDL parsing service are available by visiting the linkage: <https://github.com/phantomDai/parseesdlws.git>.

The default port of this service is 8085. If changing the port of this web service, the user needs to change the the value of variable *endpoint* in *ParseWSDL* script that is available in linkage: <https://github.com/phantomDai/drtAPI.git>. For instance, the user replaces the current value of *endpoint*

```
private static String endpoint =  
"http://202.204.62.171:8085/services/parser?wsdl" (default  
port)
```

with

```
private static String endpoint =  
"http://202.204.62.171:xxxx/services/parser?wsdl" (user's  
port)
```

As for Restful micro-services, the user needs to set up the local environment as follows:

1. Maven (that is available by visiting the linkage: <http://maven.apache.org/>)

Then, the user executes the following command in the root directory of drtAPI file that is available in linkage: <https://github.com/phantomDai/drtAPI.git>.

```
mvn clean package -Dmaven.test.skip=true
```

Finally, the user goes to the "target" directory and executes the following command in DOS.

```
java -jar ./drt-0.01-SNAPSHOT.jar
```

Congratulations! you have finished the configurations of environment.

## An example

We use an example to demonstrate web service testing using our prototype tool.

## The specification of web service under test

Aviation consignment management service (ACMS) (that is available by visiting the linkage: <https://github.com/phantomDai/drt4ws>) helps airline companies check the allowance (weight) of free baggage, and the cost of additional baggage. Based on the destination, flights are categorised as either domestic or international. For international flights, the baggage allowance is greater if the passenger is a student (30kg), otherwise it is 20kg. Each aircraft offers three cabins classes from which to choose (economy, business, and first), with passengers in different classes having different allowances.

## Step 1: Specifying url and setting parameters

The user first needs to enter the address of the WSDL of web service under test, and click the "Parse" button, and then select a interested method of web service under test in the following drop-down menu (as shown in following figure).

QSpecifying URL

Please enter the address of the web service under test

Parameters Setting

Please select an operator:

Operation

prearea  
feeCalculation  
preairclass

The user divides each parameter into disjoint options, and describes them according to predefined rules that are introduced in *Guidance* page (as shown in the following figure). Once the "Save" button is clicked, parameters and corresponding options are sent to some Restful micro-service.

Please select an operator:

feeCalculation

Index	Parameter	Type	Options
1	area	int	1-1:{0};1-2:{1};1-3:{2}
2	airClass	int	2-1:{0};2-2:{1};2-3:{2}
3	luggage	double	3-1:[0,60];3-2:(60,300]
4	economicfee	double	4-1:{0};4-2:(0,3000]
5	isStudent	boolean	5-1:{true};5-2:{false}

## Step 2: Partition construction and parameter setting

The user divides input domain by combining options with different parameters of selected method (as shown in following figure). Besides, the user needs to set the selecting probability for each partition, and the value of probability adjusting factor. After clicking the "Save" button, all provided information will be sent to some Restful micro-service, which is responsible for initializing test profile and setting the value of *epsilon*.





test report is also supported by clicking the "Download" button.

Console

Next

Index	Test Cases	Partition	Output	Exepected Output	Pass(0) / Failed(1)
1	{"area":1,"airClass":0,"luggage":2527.7,"economicfee":0.0,"isStudent":true}	0	0	NA	0
2	{"area":3,"airClass":1,"luggage":2313.0,"economicfee":0.0,"isStudent":false}	5	0	NA	0

Start

Stop

Download