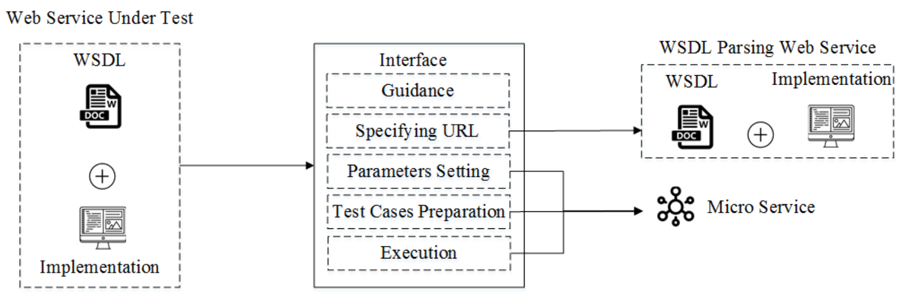
Table of content

DRTester

The prototype tool *DRTester* can test web services using dynamic random testing technique. We describe the implementation and configuration of the tool in detail.

Framework of DRTester



The above figure illustrates the *DRTester* framework, comprising four main parts, corresponding to the left of web service is the testing target; *interface* is the interface between the user and *DRTester*; the top right of web service is responsible to parse the address of the target web service's wsdl, and return information (such as methods and parameters) to *micro services* that are used to partition input domain, generate test cases, execute test case, and send information to *interface*.

We next examine each component in the framework individually.

Front-end interface

We developed a html page by using the Vue framework (https://cn.vuejs.org/), the source code of which can be obtained by visiting https://github.com/phantomDai/DRTester.git.

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

WSDL parsing service

We can obtain the necessary information by parsing WSDL 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 names and types of interested methods of web service under test, along with their parameters information (name and type). Besides, we also made this web service publicly accessible (https://github.com/phantomDai/parseesdlws.git).

Micro services

The back-end logic is composed of several Restful APIs (For more details, please visit linkage: https://github.com/phantomDai/drtAPI.git) and Java classes: The APIs are responsible for communicating HTTP messages to and from the front-end interface. The controller class is responsible for updating the test profile according to the test results, and for selecting test cases from the partitions. The selected test cases are wrapped in SOAP messages and sent to the web service under test through the proxy class, which also intercepts the test results.

Configuration of DRTester

This section descripts the configuration of the front-end interface, Micro services, and WSDL parsing service.

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 users can download the source code of which by visiting the linkage: https://github.com/phantomDai/DRTester.git. Next, users need to go the the root of the downloaded file, and creat a directory named "node_modules". The above configuration is sufficient for uses to execute the following command in DOS.

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

Finally, users need to find all *post* or *get* methods in *BaseTable.vue* and *Tabs.vue*, and change the values of *url* by replacing the address of IP with uses' IP. For instance, uses should replace the following value of *url* (on the line 266 of *BaseTable.vue*)

```
url: 'http://202.204.62.171:8082/api/parse/wsdl'
```

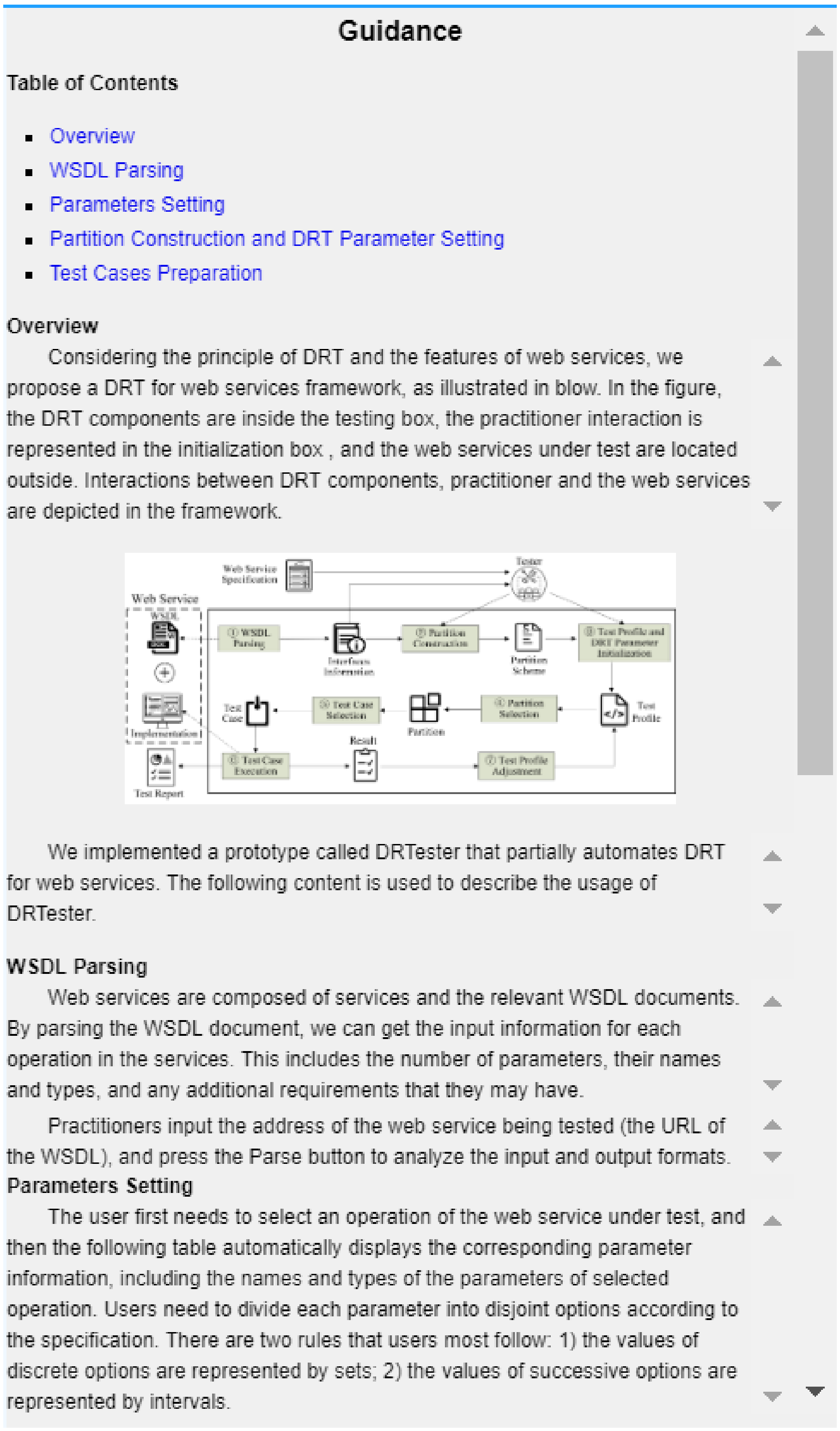
with

```
url: 'http://IP*:8082/api/parse/wsdl'
```

,where IP* is the users' IP address.

```
#  if you are not in China, you can just execute the
#  following command in DOS
npm install
npm run dev
#  if you are in China, you can just execute the following
#  command in DOS
cnpm install
npm run dev
```

After completing the above steps, users can enter http://localhost:8080 in their browser. Accordingly, the *Guidance* page is visible, which describes the steps users should follow when testing a web service.

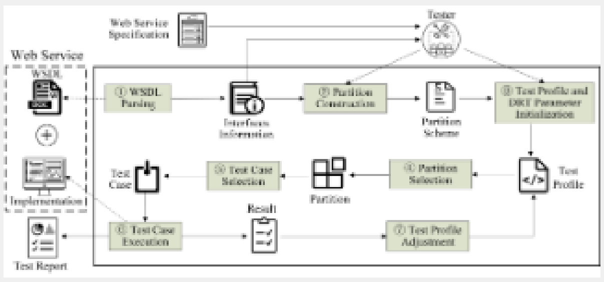


Guidance

Table of Contents

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.



We implemented a prototype called DRTester that partially automates DRT for web services. The following content is used to describe the usage of DRTester.

WSDL Parsing

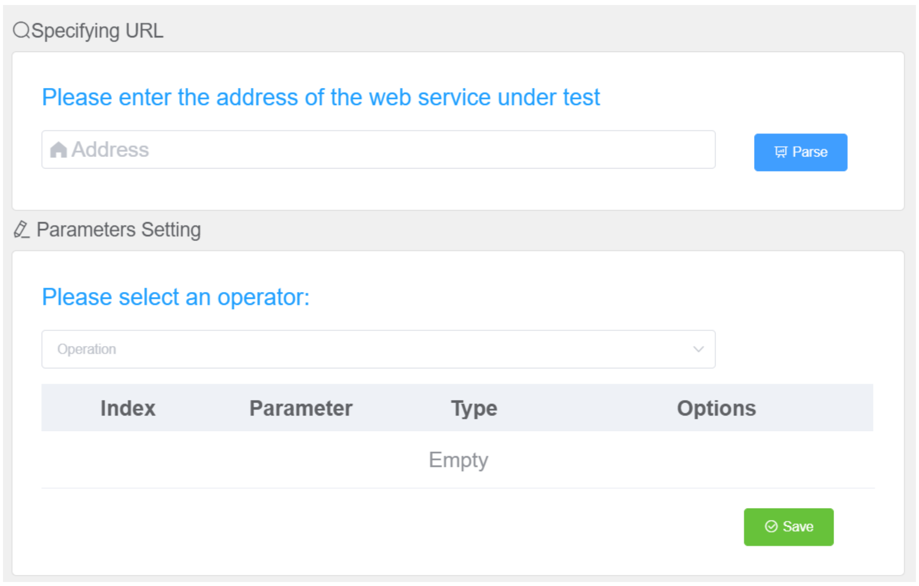
Web services are composed of services and the relevant WSDL documents. By parsing the WSDL document, we can get the input information for each operation in the services. This includes the number of parameters, their names and types, and any additional requirements that they may have.

Practitioners input the address of the web service being tested (the URL of the WSDL), and press the Parse button to analyze the input and output formats.

Parameters Setting

The user first needs to select an operation of the web service under test, and then the following table automatically displays the corresponding parameter information, including the names and types of the parameters of selected operation. Users need to divide each parameter into disjoint options according to the specification. There are two rules that users most follow: 1) the values of discrete options are represented by sets; 2) the values of successive options are represented by intervals.

Then user can parse WSDL, set parameters, partition input domain, generate test cases, and download test report in the following interfaces.



Specifying URL

Please enter the address of the web service under test

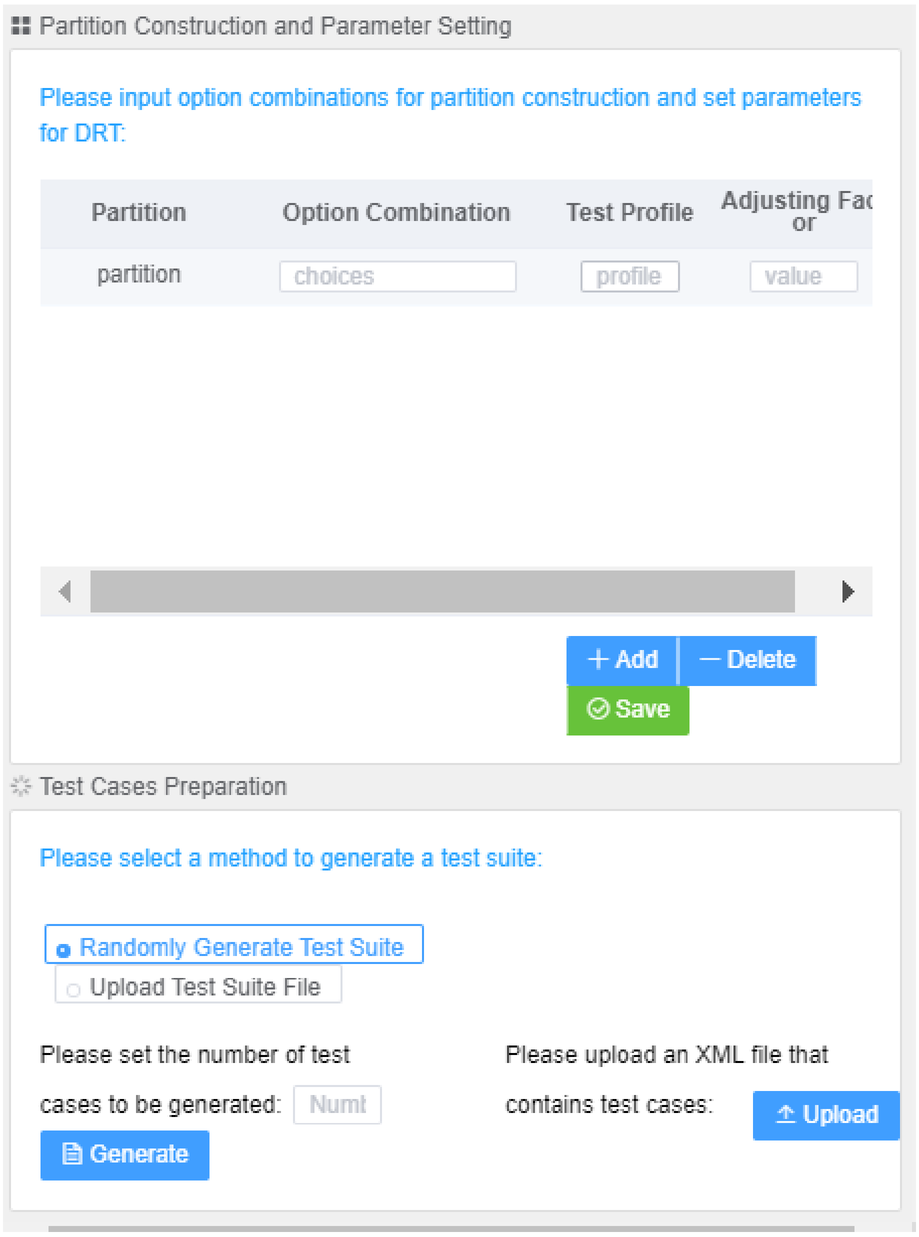
Address

Parameters Setting

Please select an operator:

Operation

Index	Parameter	Type	Options
		Empty	



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	choices	profile	value

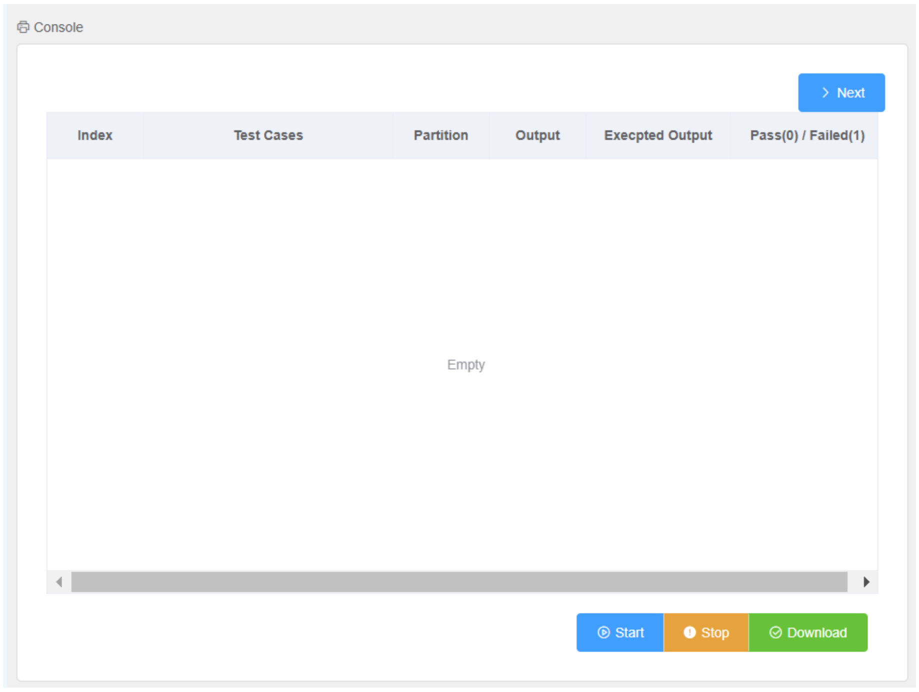
Test Cases Preparation

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: Numb

Please upload an XML file that contains test cases:



Index	Test Cases	Partition	Output	Execpted Output	Pass(0) / Failed(1)

Empty

Before running this web service, the user needs to configure the following environment.

1. Tomcat 9.06 (The installer for Tomcat 9.06 is provided in this repository)

The default port of this web service is ***8085***. If changing the port of this web service, the user needs to change the the value of parameter *endpoint* in *ParseWSDL* script that can be available in linkage: https://github.com/phantomDai/drtAPI.git.

To make things simple, you can just download the ***drtAPI.jar*** from the linkage above, and execute the following command in the directory that include ***drtAPI.jar*** to release all Micro services.

```
java -jar ./drtAPI.jar
```

An example of testing web service

We show an example of testing web service using our prototype tool.

Environment

- Windows 10
- Tomcat 9.06
- JDK 1.8.0_161
- The address of the WSDL of web service under test: http://202.204.62.171:8081/services/acms?wsdl

The specification of web service under test

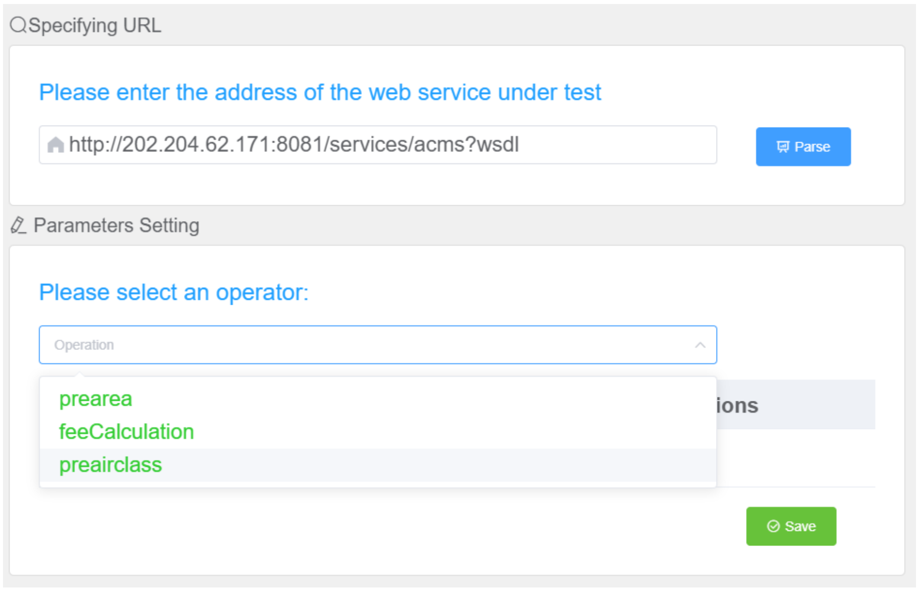
Aviation consignment management service (ACMS) (that is available by visiting the linkage: https://github.com/phantomDai/drt4ws) helps airline companies check the allowance (weight) of free baggage, and the cost of additional baggage. Based on the destination, flights are categorised as either domestic 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.

Testing steps

The details of testing steps are as follows.

Step 1: Specifying url and setting parameters

Users first need to enter the address of the WSDL of web service under test (WSUT), and click "Parse" button, and then a method of WSUT can be selected in the following drop-down menu (as shown in following figures).


Specifying URL

Please enter the address of the web service under test

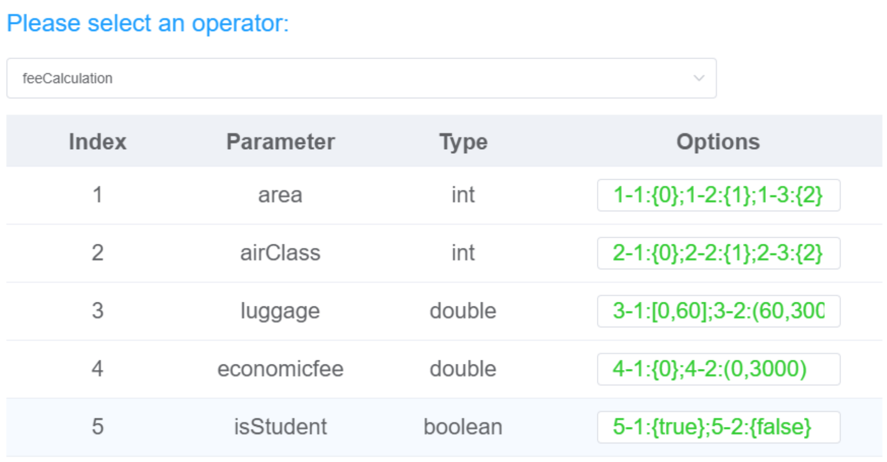
http://202.204.62.171:8081/services/acms?wsdl

Parameters Setting

Please select an operator:

Operation

Users must partition each parameter into disjoint options, and describe them according to predefined rules that are introduced in ***Guidance*** page (as shown in the following figure).

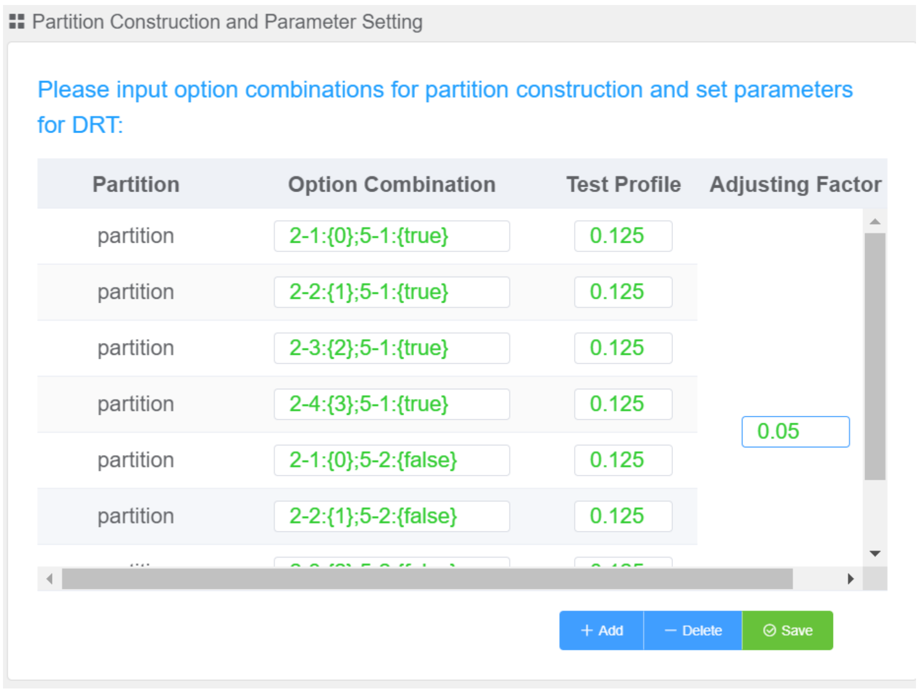

Please select an operator:

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

Users must partition input domain by combining options with different parameters of selected method (as shown in following figure).

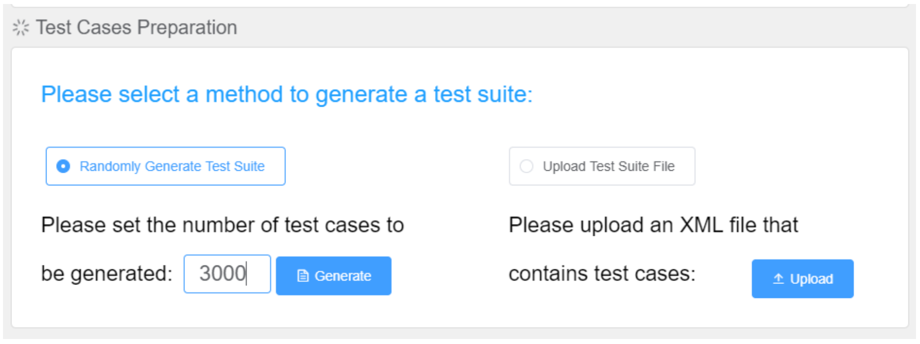

Partition Construction and Parameter Setting

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

Partition	Option Combination	Test Profile	Adjusting Factor
partition	2-1:{0};5-1:{true}	0.125	0.05
partition	2-2:{1};5-1:{true}	0.125	
partition	2-3:{2};5-1:{true}	0.125	
partition	2-4:{3};5-1:{true}	0.125	
partition	2-1:{0};5-2:{false}	0.125	
partition	2-2:{1};5-2:{false}	0.125	

Step 3: Test case preparation

We provide two methods to generate test cases: 1) Randomly generate test cases; 2) Upload Json file that include test cases. Note that there are rules about the format of the uploaded Json file, which are described in ***Guidance*** page.


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: 3000

Please upload an XML file that contains test cases:

Step 4: Test case execution

After performing all the steps above, all necessary stuff for testing are ready. Users first click *start* button, then the table in the middle of page shows the information of testing. If test cases are generated using random strategy, users must decide whether the last test case detected a fault.