# Paper Title: Dynamic Random Testing of Web Services: A Methodology and Evaluation

## Manuscript ID: TSC-2019-01-0031

Dear Editor-in-Chief,

Thank you for your email on May 8, 2019 regarding our paper titled “Dynamic Random Testing of Web Services: A Methodology and Evaluation” submitted to IEEE Transactions on Services Computing (Manuscript ID: TSC-2019-01-0031).

We are submitting a new version of the paper, in which we have made revisions to address and respond to each comment from the reviewers. Below are the detailed responses to the comments.

We look forward to hearing from you.

Yours sincerely,

Chang-ai Sun, Hepeng Dai, Guan Wang, Dave Towey, Kai-Yuan Cai, and Tsong Yueh Chen

# Response to comments of associate editor and reviewers

In the following, unless otherwise specified, all comments refer to the revised version of the paper.

### Associate editor’s comments

***E1C1: Having analyzed in detail the reviews and the manuscript, I have matured my*** ***recommendation to the EIC, which is that it undergoes a major revision.***

Response: Thank you for your recommendation.

Action: None.

***E1C2: I believe the reviewers provided useful comments for you to improve the manuscript. Should you choose to revise your manuscript, pay attention to address all reviewers' concerns, especially those on: (1) the novelty of this paper with respect to own previous publication; (2) the improvements to the writing of various Sections; (3) the repeatability of experiments; (4) the definition of failure rate in the 2nd research question and the revision of the answer to the 3rd research question; (5) more details about the mutants generated; (6) insight as to why CP testing was able to uncover faults; (7) the quality of the reported graphics.***

Response: Thank you for summarizing the comments from all the reviewers. All these comments have been responded and the corresponding revisions have been made accordingly. Please refer to our response to R3C2 for the first concern; responses to R1C2, R2C2, R2C3, R2C9, R2C10, R3C4, R3C5, R3C6, and R3C7 for the second concern; response to R2C5 for the third concern; responses to R2C7 and R2C8 for the fourth concern; response to R1C5 for the fifth concern; response R1C5 to the sixth concern, and response to R2C9 for the seventh concern.

Action: None.

### Reviewer 1’s comments

***R1C1: The paper is relevant to the services computing community. However, the specific representations for the SOA used in the paper (WSDL) etc sound a bit dated.***

Response: Thank you for the comment. Although Web services are somehow dated, they are still adopted for developing various applications. For instance, a representative Web service repository (https://github.com/ouniali/WSantipatterns) contains 226 realistic Web services in various domains. On the other hand, researchers from academia are still working on the performance improvement of Web services, and reporting their research results in top conferences or journals in this area. For instance, several work related to Web services has been recently published in TSC, such as: *S. Wang, Y. Ma, B. Cheng, et al. Multi-Dimensional QoS Prediction for Service Recommendations, IEEE TSC,* *2019, 12(1):47-57; P. Wang, X. Du. QoS-Aware Service Selection Using an Incentive Mechanism, IEEE TSC, 2019, 12(2):262-275.*

Action: None.

***R1C2: The probability distribution computation model described in section 3.2 is too detailed. Some of the proofs etc and moved into an appendix without taking away from the core message.***

Response: Thanks for the suggestion.

Action: In the revised version, we have followed the suggestion and moved the proofs in Section 3.2 to the appendix.

***R1C3: The technique also requires a user to provide category partition details along with an initial test profile. This could be an unreasonable expectation from a test practitioner.***

Response: Thank you for the comment. It is very common that certain inputs are expected from the tester when she/he exercises a testing technique. In the context of partition testing, the tester has to construct partitions based on the specification of software under test. Furthermore, partition details can be derived in different ways, such as equivalent class method or category partition method. Since our technique is based on partition testing, partition details are naturally expected to be provided from the user. This can be easily done by analysing the input parameters and constraints among them described in the specification of Web service under test. Once partition details of Web service under test are available, then it is not difficult to set an initial test profile. For simplicity, the tester can use the uniform probability distribution (i. e. *P1*=*P2*=…=*Pk*=*1/k*, where *k* denotes the number of partitions, and *Pi* (*i*=*1..k*) denotes the probability of the *i*th partition). As a result, our technique only requires a user to provide additional information on the initial test profile compared with original partition testing, and such additional information should be affordable.

Action: In the revised version, we have added a discussion on inputs required from the tester for our technique in Section 3.1.

A: Explain the reason or argue the reasonable of the proposed approach

[For discussion]

***R1C4: The applications used in the study are fairly small (~100 SLOC). While the authors say it is not possible to gain access to service request implementations, they do seem to have access to the implementation of the web services used in the experiments in order to create the faulty mutants.***

Response: We do understand the reviewer’s concern on the size of subject Web services used in the study. Definitely, it will be more convincing to include larger open-source subjects for evaluation. However, to the best of our knowledge, there are not such subjects in the field of Web services. Furthermore, our evaluation needs to access the source code of Web services in order to seed faults. However, the owner of realistic Web services is not willing to make the source code accessible since the implementation of a Web service involves commercial interests or technical secrets.

In order to overcome the unavailability of realistic open-source Web services, we decided to develop the subject applications based on the real-life specifications. In this way, we are able to access the source code of these Web services for evaluation.

Action: In the revised version, we have added a note to clearly state that subject Web services are developed in our laboratory based on the real-life specifications in first paragraph of Section 4.2.

***R1C5: Authors should provide more details about the actual mutants generated and provide insight as to why the specification based testing (CP testing) was able to uncover those faults.***

Response: Thanks for the suggestion. We agree that the suggested details are helpful to understand the experimental settings. Accordingly, we decided to include more details about the actual mutants generated, including the tool that was used to generate mutants, mutation operators that were employed to generate mutants, and how the generated mutants were finally selected for evaluation.

We also agree that it is necessary to provide insight as to why the specification-based testing is able to uncover those faults. In our study, we employed decision table (DT rather than CP) to construct partitions for each studied Web service, which is described in Section 4.4.1. DT provides a systematic and efficient way to partition input domain into disjoint subdomains and generate test cases, since DT considers all parameters and identifies invalid combination of parameters. In practice, each condition entry of a DT rule corresponds to a partition in which test cases cover some paths, accordingly, the faults in those paths has a chance of being detected.

Action: In the revised version, we have followed the suggestion to provide more details on the mutants generated in the first paragraph of Section 4.2, and provide inside as to why specification-based testing is able to reveal those faults in Section 4.4.1.

### Reviewer 2’s comments

***R2C1: The problem addressed in this manuscript is interesting. The proposed solution is appealing because of its simplicity and low applicability effort; moreover it improves significantly the performance of random testing and partition testing (commonly used in web services testing).***

Response: Thank you for the endorsement.

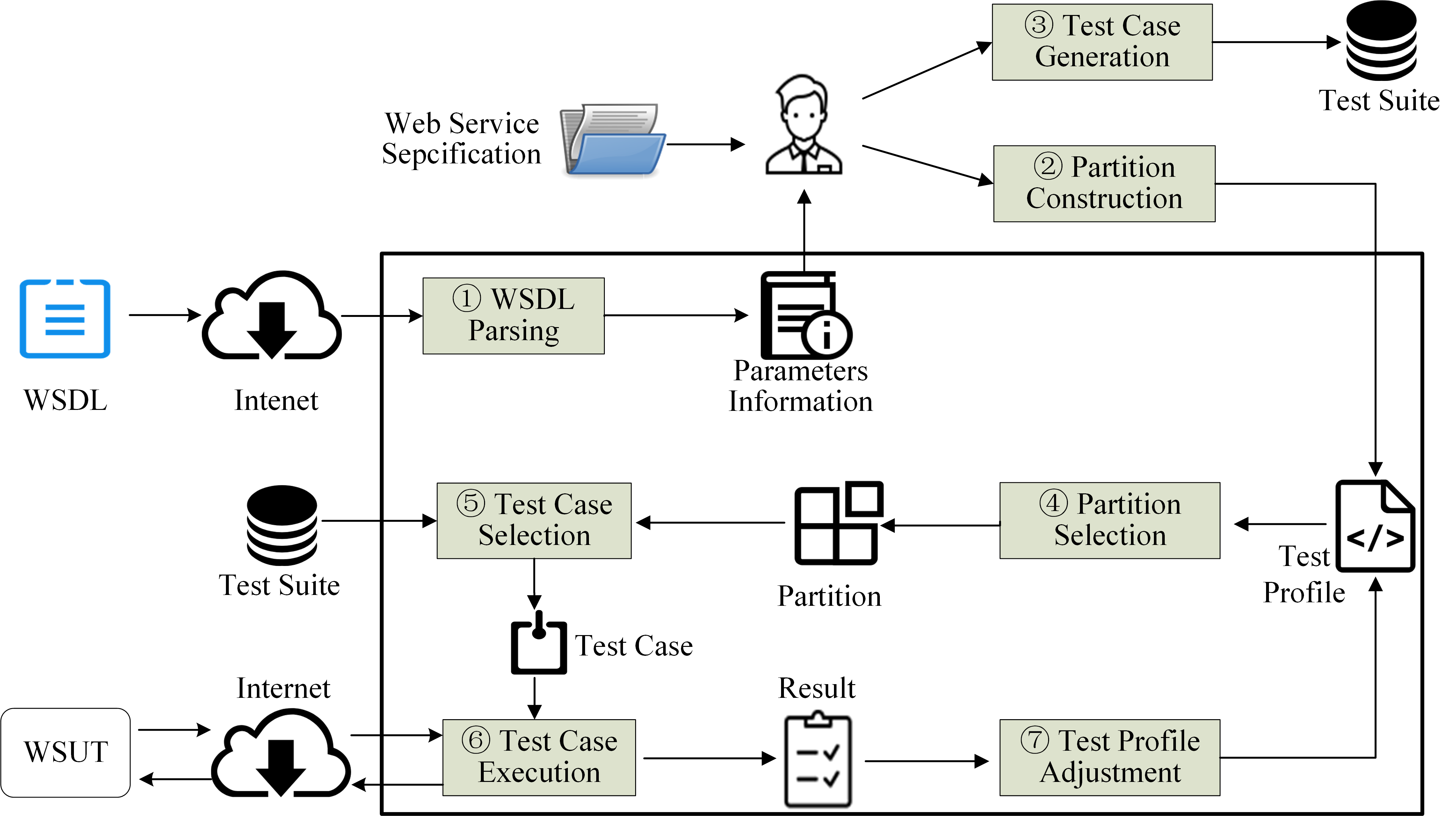
Action: None.

***R2C2: In the model in Figure 1, the human interaction is not represented, although it is important not only for DRT parameters configuration, but also for partitions construction as specified in Section 3.3 “The tool provides two options for the partitions and test suites: either to manually specify the partitions (and test cases); or to upload the predefined partitions and test suites”.***

Response: Thanks for the comment. Our method DO needs user interaction, such as DRT parameter initialization, partition construction. Therefore, we have followed the reviewer’s comment, and added the user interaction in our framework illustrated in Figure 1.

Action: In the revised version, we have added the human interaction in Figure 1. The user interaction mainly includes parameter initialization, partition construction, and test cases uploading (that is the user does not choose randomly generate test cases). Accordingly, we have changed the part of description about our framework in the first paragraph of Section 3.1 and added a discussion about the component called “Test Case Selection”.

Intended details and updated framework are as follows:



Test Case Selection: DRT selects a test case from selected partition s\_i according to a uniform distribution.

***R2C3: The first contribution reported by authors in Section 3.1 is “a DRT framework that addresses key issues for testing web services, and a prototype that partly automates the framework”. From the description in section 3.1, the focus is more on positive features than on issues, in fact the WSDL document allows to automate part of the testing process, that is a common practice in web services testing. The prototype description in Section 3.3 is very thin and too similar to the one in the conference paper. Further details are desirable.***

Response: Thanks for the comment. We agree that we should add more description of prototype and improve the quality of our prototype from three aspects, which include parameters setting, partition construction and results exhibition.

We have developed a new prototype called DRTester. Compared to the previous prototype, DRTester has an interface called Handbook that guides user to use this prototype and “Configuration” interface where user can follow the handbook to set parameters, construct partition, and generate test cases. Besides, DRTester also provides “Execution” interface where the user can obtain information about the execution process of WSUT and download the testing results.

Accordingly, we have added more details about DRTester in Section 3.3.

Action: In revised version, we propose a totally different prototype that is easier to use and more flexible. Accordingly, we have added more details to describe our prototype.

Intended details are as follows:

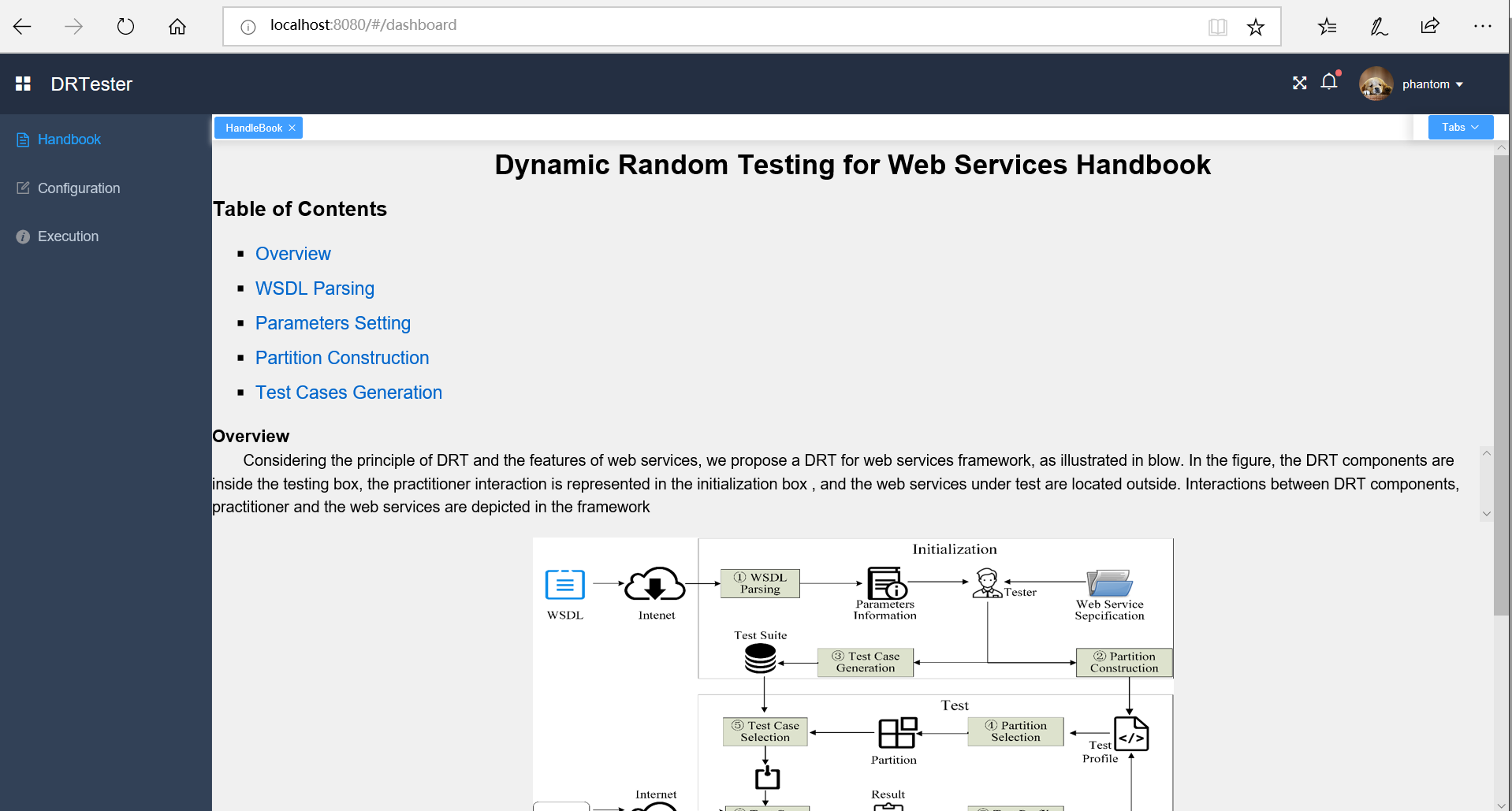
Up to now, I just have finished the web pages of DRTester and the most of the backend logic. I need extra time to implement our tool (about a week).

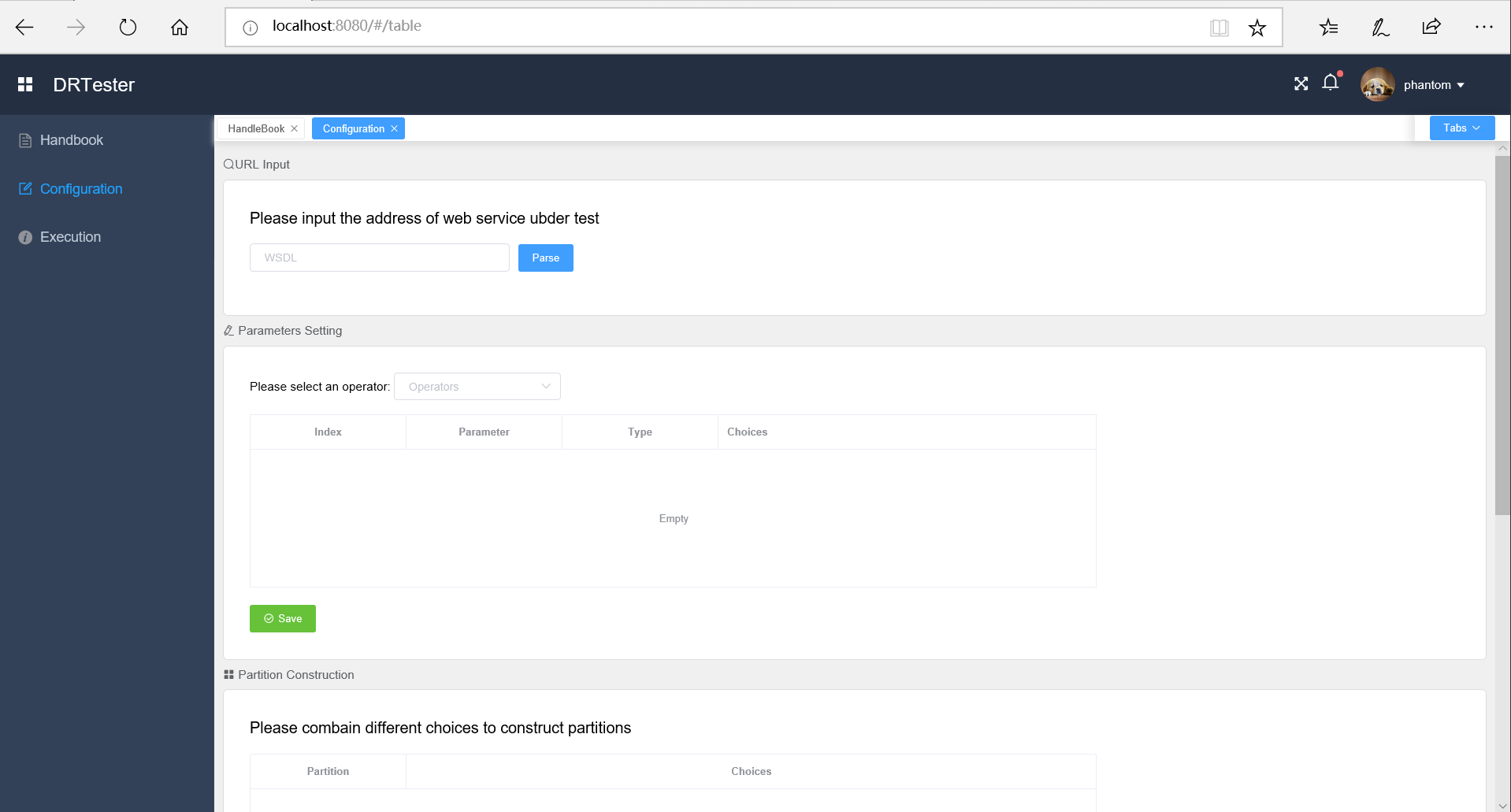
The below pictures present the all interfaces, including Handbook, Configuration, and Execution. We next discuss the individual web pages.

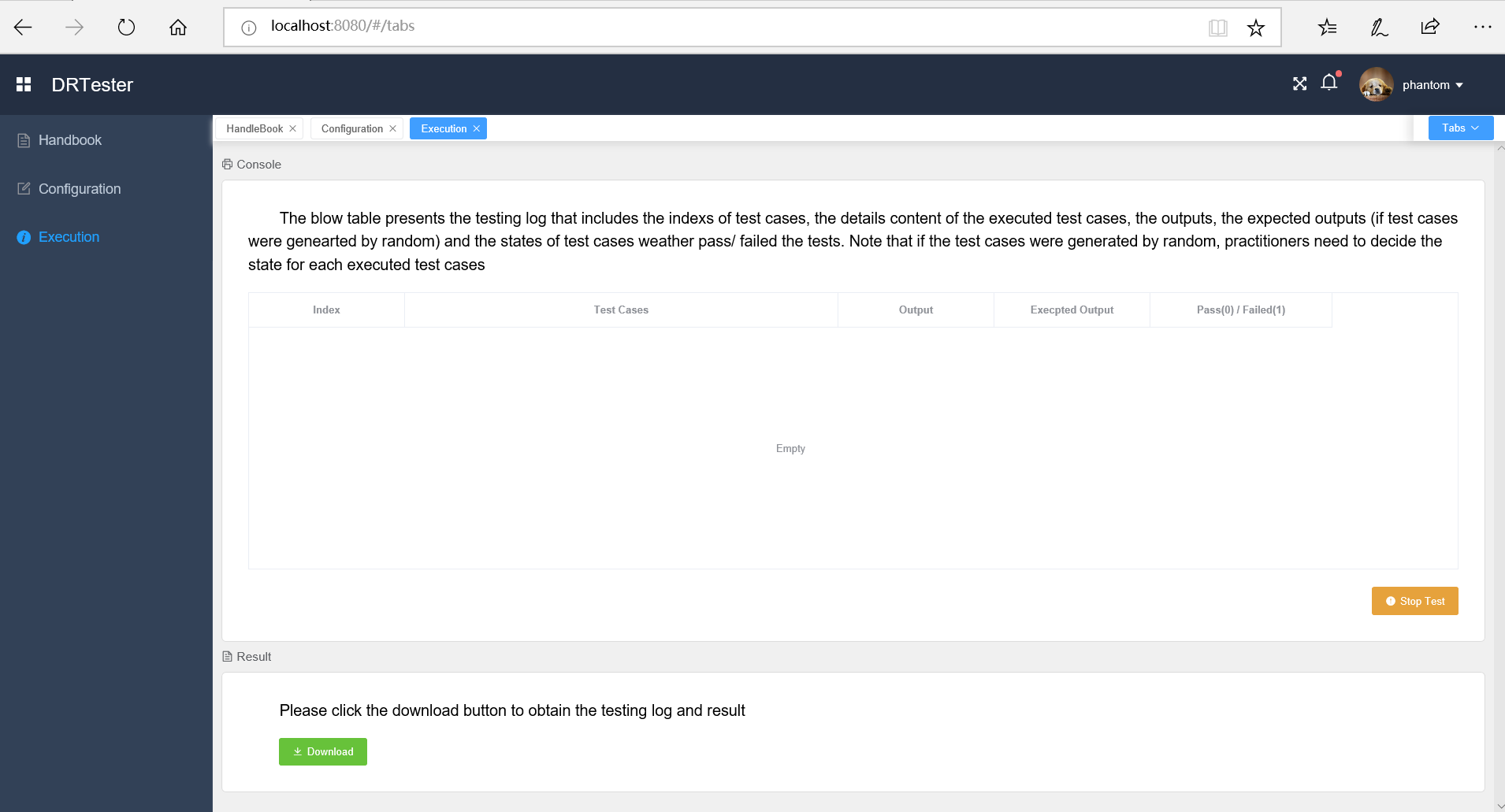
1. Handbook. This document describes the steps we follow to test a web service. We first introduce the framework of DRT for testing web services based on the principle of DRT and features of web services. Then, according to the presented framework, we detail the configuration and operation of each step.
2. Configuration. This web page implements the interaction of DRTeser with practitioners in order to obtain all information for testing web services, including the address of web service under test (WSDL), the information of parameters and partitions, and test cases. Detailed configuration steps are shown below:

* URL inputting: DRTester integrates some of the features of MT4WS (*C-A. Sun et al. MT4WS: an automated metamorphic testing system for web services*, *IJHPCN 9(1/2): 104-115, 2016.*), so as to obtain all parameters and their types. Those information presents the component of “Parameters setting”.
* Parameters setting: In this step, practitioners are responsible to choose a function of web service under test and partition each parameter into distinct choices that include all different kinds of values that are possible for the parameter.
* Partition construction: Practitioners construct partitions though combining different choices that come from distinct parameters.
* Test case generation: DRTester obtains test cases though different method, including randomly generating test cases based on the information of parameters setting and uploading test cases by practitioners.

1. Execution. This web page presents the details of testing, including the index of test case, test data, output, excepted results, and testing result (i.e. pass or fail). Note that if practitioners choose to randomly generate test cases, the practitioners have to check the testing result one by one. When all tests have completed, the test report is summarized and output in a file that could be download by practitioners.







***R2C4: The experimentation is conducted generating mutants of three web services. The authors remove equivalent mutants, and mutants that can be detected with less than 20 randomly generated test cases. This latter criterion is strongly influenced by luck: a mean of the number of test cases generated through multiple repetitions is more robust.***

Response: Thanks for the comment. We totally agree with the reviewer’s standpoint. In our previous version, we did not describe in detail the selection process of used mutants. In fact, for each mutant we used 50 random seeds to generate different test suites, and calculated the average number of test cases needed to kill the mutant, which could be confirmed in our test scripts (<https://github.com/phantomDai/evidenceDRT>).

Action: In the revised version, we have clearly mentioned the repeated times (50) to select used mutants in Section 4.2.

***R2C5: Some defections make the repeatability of experiments impossible, more details are required:***

***- (1) The test profile initialization is not explicitly reported, although the authors, in the Section 4.4.2, indicate that “a feasible method is to use a uniform probability distribution as the initial testing profile. On the other hand, testers may also use past experience to guide a different probability distribution as the initial profile”.***

***- (2) The authors set the partitions by making use of decision table, obtaining two partition schemes for each application. The decision tables used in the experiments for the partitioning are not reported.***

***- (3) The applications are well described, but there is no reference to where they are taken (open source repository, private repository, …).***

Response: (1) If practitioners have no past experience to initialize test profile, they can use the uniform probability distribution (that is p\_1 = p\_2 = … = p\_m, where m is the number of partitions) as the initial test profile. In our other paper titled Adaptive Partition Testing, we compared the equal and proportional (p\_i = k\_i / k, where k is the total number of test cases in the test suite and k\_i is number of test cases in the partition c\_i) initial test profile in terms of F, F-2, and T-measure. The results shown that there was no significant different between these two types of initial probability profiles, that means there is no a strong correlation between the probability profile type and DRT’s performance. The reason is that DRT can automatically update the test profile during the test, increase the selection probabilities of partitions with larger failure rates, and decrease the selection probabilities of partitions with smaller failure rates. In a word, practitioners can use any method that is easy to use to construct partitions, and then use the uniform probabilities distribution as the initial test profile.

Otherwise, practitioners have the experience, they can set larger selection probabilities to the partitions with higher failure rates and accordingly set a smaller selection probabilities to the partitions with lower failure rates.

(2) Respond to the second point: We agree that reporting partition information of studied web services can make our experiment setting more explicit. In our previous version, there was not enough space to report detailed partition information. In the revised version, since we have followed another reviewer’s suggestion, and moved the proof of Section 3.2 to appendix, there is enough space to report detailed partition information.

(3) Respond to the third point: Sorry for the confusion, the subject Web services were developed in our laboratory based on the real-life specifications.

Action (1) In the revised version, we have reported test profile initialization for each studied Web service in Section 4.4.2.

Intended details are as follows:

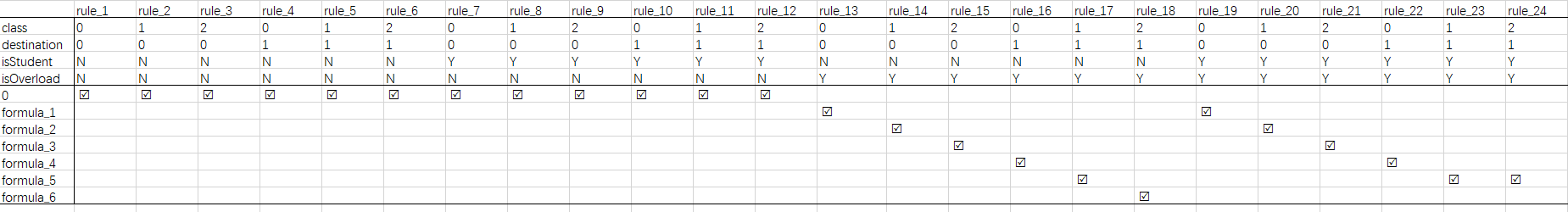
Because test cases may be generated randomly during the test process, a feasible method is to use a uniform probability distribution as the initial test profile. On the other hand, testers may also use past experience to guide a different probability distribution as the initial profile.

In our other paper titled Adaptive Partition Testing, we compared the equal and proportional (p\_i = k\_i / k, where k is the total number of test cases in the test suite and k\_i is number of test cases in the partition c\_i) initial test profile in terms of F, F-2, and T-measure. The results shown that there was no significant different between these two types of initial probability profiles, that means there is no a strong correlation between the probability profile type and DRT’s performance. The reason is that DRT can automatically update the test profile during the test, increase the selection probabilities of partitions with larger failure rates, and decrease the selection probabilities of partitions with smaller failure rates. In this study, we used a uniform probability distribution as the initial test profile, and the below table presents the details of initial test profiles of all studied web services.

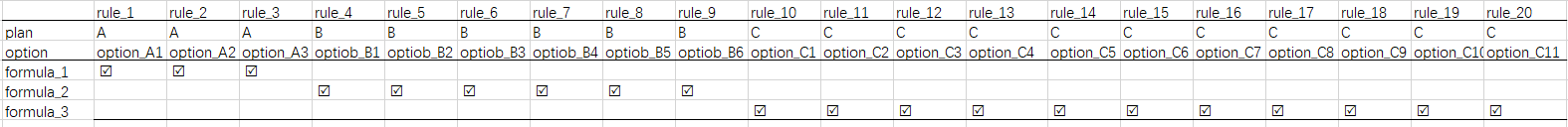
|  |  |  |
| --- | --- | --- |
| Web service | Number of partitions | Initial test profile |
| ACMS | 24 |  |
| 7 |  |
| CUBS | 20 |  |
| 3 |  |
| PBS | 18 |  |
| 3 |  |

(2) In the revised version, we have reported detailed partition information for all Web services in Section 4.4.1.

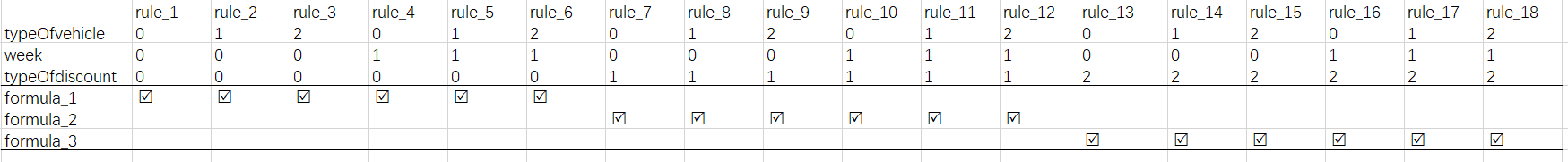
In the last paragraph of Section 4.4.1, intended details are as follows:

A DT rule is composed of a CE and its corresponding AE. With DT, it is possible to obtain partition schemes with different granularities. For fine-grain partition schemes, each CE of a DT rule corresponds to a partition; while for coarse-grained schemes, a partition corresponds to the union of a group of partitions of which all CE of DT rules have the same AE. Accordingly, we obtained two partition schemes for each subject web service: ACMS had 24 and 7 partitions; CUBS, 20 and 3; and PBS, 18 and 3, respectively. The below figures are the fine-grain DTs of three studied web services.

(A) The fine-grain DT of ACMS



(B) The fine-grain DT of CUBS



(C) The fine-grain DT of PBS

The explanation of Figure A: In order to check the cost of additional baggage, we identified the related conditions that affect the result of calculating the fee of additional baggage, along with the options for each condition (the details are presented in Table A1 where we use some simple characters to make presentation easier). The lower-left of Figure A shows all possible actions that is all formulas for calculating the fee of additional baggage. The formulas are presented in Table A2, where the “0” presents that the passenger does not have additional baggage. The main difference of between other formulas is the values of free baggage that are confirmed by the destination, domestic and isStudent (Table 2).

Table A1 the conditions and corresponding options

|  |  |
| --- | --- |
| Conditions | Options |
| class | 0: First class; 1: Business class; 2: economy class |
| isStudent | N: the passenger is a student; Y: the passenger is not a student |
| isOverload | N: there is nor additional baggage; Y: there is additional baggage |
| Destination | 0: domestic; 1: international |

Table A2 the formulas to calculate the fee of additional baggage

|  |  |
| --- | --- |
| Formulas | Presentation |
| 0 | 0 |
| Formula\_1 |  |
| Formula\_2 |  |
| Formula\_3 |  |
| Formula\_4 |  |
| Formula\_5 |  |
| Formula\_6 |  |

The explanation of Figure B: In order to help customers to know the bills of cell-phone, we identified the related conditions that affect the result of calculating the bills, along with the options for each condition (the details are presented in Table B1 in which we use some simple characters to make presentation easier and we use option\_XY to present the option of Plan X, where , and ). The lower-left of Figure A shows all possible actions that is all formulas for calculating the bill. The formulas are presented in Table B2. The main difference of between formulas is the fee of a one-minute call.

Table B1 the conditions and corresponding options

|  |  |
| --- | --- |
| Conditions | Options |
| Plan | A: plan A; B: plan B; C: plan C |
| Option | option\_A1, option\_A2, option\_A3; option\_B1, option\_B2, option\_B3, option\_B4, option\_B5, option\_B6; option\_C1, option\_C2, option\_C3, option\_C4, option\_C5, option\_C6, option\_C7, option\_C8, option\_C9, option\_C10, option\_C11 |

Table B2 the formulas to calculate the bills.

|  |  |
| --- | --- |
| Formulas | Presentation |
| Formula\_1 |  |
| Formula\_2 |  |
| Formula\_3 |  |

The explanation of Figure C: In order to calculate the parking fee, we identified the related conditions that affect the result of calculating the parking fee, along with the options for each condition (the details are presented in Table C1 in which we use some simple characters to make presentation easier). The lower-left of Figure A shows all possible actions that is all formulas for calculating the bill. The formulas are presented in Table C2 where the *baseFee* is calculated based on the type of vehicle, day of week, and the total time of parking car. The main difference of between formulas is the value of discount.

Table C1 the conditions and corresponding options

|  |  |
| --- | --- |
| Conditions | Options |
| typeOfVehicle | 0: motorbike; 1: 2-door coupe; 2: others |
| week | 0: weekend; 1: workday |
| typeOfdiscount | 0: discount voucher; 1: estimation holds on actual parking hours;2: estimation does not hold on the actual parking hours |

Table C2 the formulas to calculate the bills.

|  |  |
| --- | --- |
| Formulas | Presentation |
| Formula\_1 |  |
| Formula\_2 |  |
| Formula\_3 |  |

(3) In the revised version, we have added a sentence to clearly explain where they are taken in the first sentence of the first paragraph of Section 4.2.

Intended details are as follows:

We selected three web services as the subject programs for our study: Aviation Consignment Management Service (ACMS), China Unicom billing service (CUBS), and Parking billing service (PBS). Note that the owner of realistic Web services is not willing to make the source code accessible since the implementation of a Web service involves commercial interests or technical secrets. Therefore, we developed those web services by ourselves based on the real-life specification.

***R2C6: In the first Research Question, DRT is evaluated by comparing the effectiveness (in terms of F-, F2- and T-measure) with that of RT and RPT, also if DRT is described in references as an improvement of RT and PT. An additional comparison with a more competitive technique is more*** ***significant to appreciate effectiveness of DRT. Some techniques that improve RT and PT are reported by authors in Section 6.2.***

Response: Thanks for the suggestion. As we description in Section 6.2, related work mainly includes two kind of techniques: adaptive random testing (ART) and adaptive testing (AT). ART is aiming to improve the fault-detecting efficiency of RT. AT takes advantage of RT and PT, but may require a very long execution time in practice, which is a driving force to develop DRT. There are some techniques that improve the performance of DRT depended on different intuition. Our method just employed original DRT technique, hence we have followed the reviewer’s suggestion and added AT as one of baselines.

Action: In the revised version, we have added AT as one of baselines.

A: Whether a benchmark technique mentioned in Section 6.2 should be included for comparison? [For discussion] (AT 似乎是可以的!)

In [7], Cai et al. first proposed DRT and used AT-GA, AT-GA-200, and RT as baselines to evaluate the effectiveness of DRT. Therefore, I think that we can also use AT-GA and AT-GA-200 as our baselines.

If we decide to use AT as a baseline, I will start experimenting immediately.

***R2C7: In the second Research Question (Section 5.2), the definition of failure rate is not clear: it is defined as the ratio between k and ki, where k is the number of test cases until revealing a fault and ki is the total number of test cases in si, that could be infinite. For instance, if a parameter can take all real values between 1 and 10, the number of inputs is*** ***countable*** ***infinite. This formulation of failure rate is more proper of test case “selection” algorithm. In this case, a solution is to consider ki as the total number of test cases performed to reveal an error and k equal to 1. Failure rate should be defined as #number of failure/#number of executed tests.***

Response: Thanks for the comment. We agree with the reviewer’s all standpoints. We have revisited the method of calculating partition failure rates, and discovered that we used the formula 1 / ki (ki is the number of executed test cases until revealing a fault) to calculate the failure rate of a partition. As we described in Section 4.2, all studied web services contain the parameters that have countable infinite valid values, hence any partition of studied web services has infinite test cases. In order to get approximate partition failure rates, we used the formula 1/ ki, which can be conformed in our test scripts (<https://github.com/phantomDai/evidenceDRT>).

Action: In the revised version, we have rewritten the definition of failure rate in the second sentence of the second paragraph of Section 5.2.

A: Re-execute the experiment

Intended details are as follows:

Before starting the test, the failure rate \theta\_i of partition s\_i was needed. From Table 2-6, we can observe that the values of some parameters (such as the weight of baggage, the minutes of calls, and parking hours) are countable infinite that means we can obtain infinite test cases in a partition. In such case, the formula 1 / k\_i, where k\_i is the total number of test cases performed to reveal a fault, to get the approximate failure rate of the partition s\_i.

***R2C8: The purpose of the Third Research Question is to validate that DRT requires linear time to generate test case through empirical examination of the actual test case generation and execution. Instead, in Section 5.3 the focus is on the comparison among the three techniques, without any reference to the temporal complexity. For this reason, the answer of the Research Question must be revised.***

Response: Thanks for the suggestion. Our description in Section 4.1 does not match the content of our investigation in Section 5.3. Compared to RT and PT, DRT increases the time overhead, since DRT uses the results of the test to control the test process. We wish to compare the time overhead in terms of actual test case generation and execution with RT and PT, without reference to time complexity.

Action: In the revised version, we have changed the third research question to match the content of Section 5.3.

Intended details are as follows:

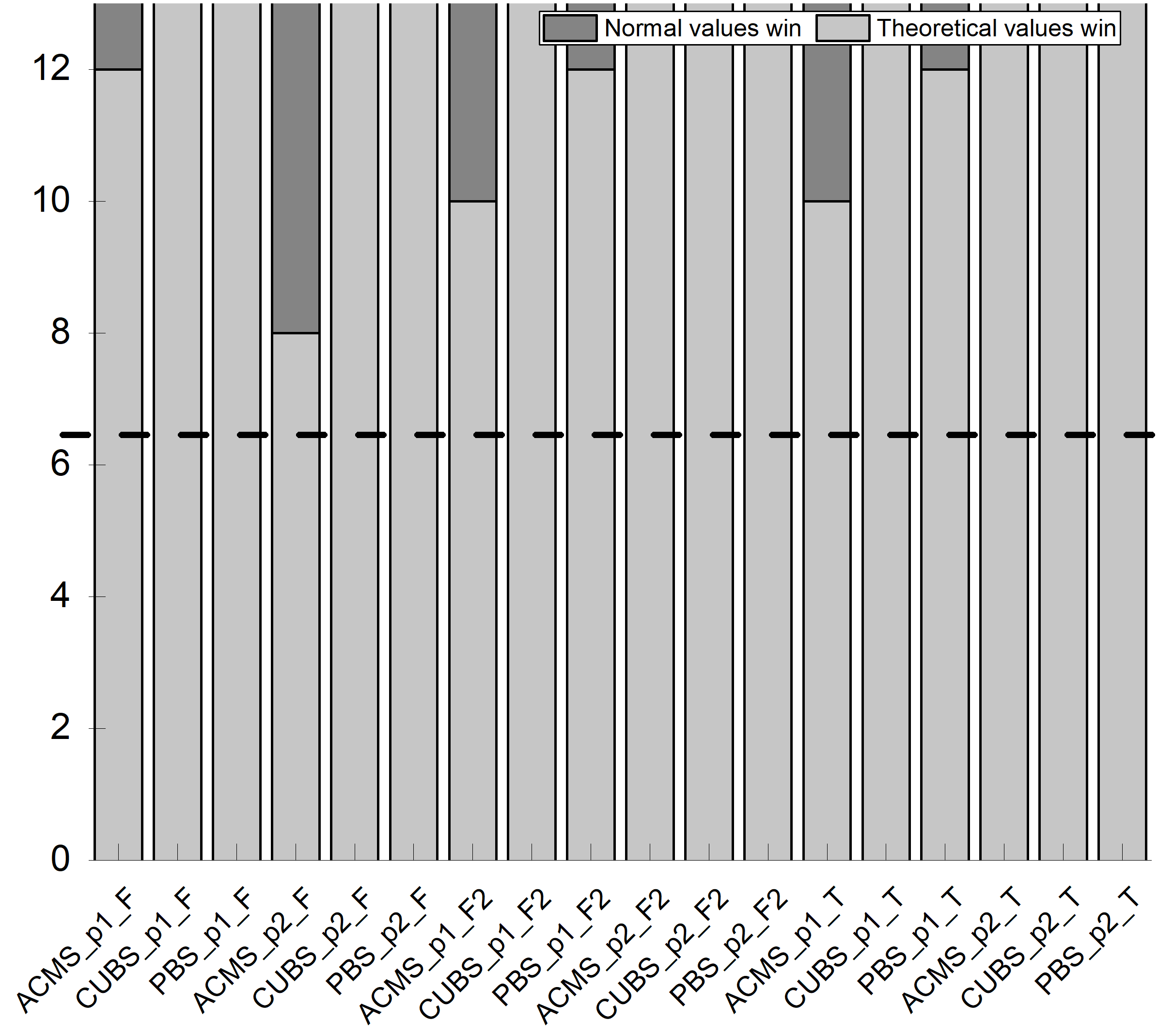
To investigate how much computational cost will DRT incur in actual testing task, we record the time consumptions on the test case generation process of all testing strategies and web services. Then we discuss the time overhead of DRT by comparing their computational costs.

***R2C9: In some cases, the results graphics could be scaled better, because the difference between the various elements is not very appreciable in the printing of paper.***

Response: Thanks for the suggestion. Indeed, Figures 3~9 were a bit crowded in order to save the space. In particular, Figure 6 includes many elements, and some points are overlapping with those elements, which may make it difficult to distinguish. To overcome this drawback, we decide to use stacking bar chart to present the significant difference between DRT with theoretical parameters and DRT with normal parameters (i.e. non-theoretical parameters).

Action: In the revised version, we have followed the suggestion to redraw all figures in the paper to make them appreciable.

The below graphic is the updated Figure 6.



***R2C10: The article is well written and it is easy to read. An incorrect tables layout is evident: Table 5 is printed after Table 2 and before Tables 3, 4 and 6.***

Response: Thanks for the suggestion.

Action: In the revised version, we have followed the suggestion to rearrange the layout of Tables 2~6.

Reviewer 3’s comments

***R3C1: The paper investigates the quality of service-based applications and proposes a dynamic random testing (DRT) technique for web services that improves existing methods. The authors present the technique, a framework for its usage and a prototype implementation. An empirical study including three different case studies is also included to show the effectiveness of the proposed approach.***

Response: Thanks for the endorsement.

Action: None.

***R3C2: The paper deals with an interesting topic and the proposed method extends an idea already proposed in [1]. I have a few comments about the current version of the paper. The authors should explicitly state in the Introduction the novelty of this paper with respect to their previous publication. The method and the prototype seem to be already present in [1] together with some empirical studies. Besides the original idea has already been presented in [7]. Even if a list is provided in the cover letter, a clear statement about the differences is necessary here.***

Response: Thanks for the comment. Indeed, it is necessary to add a clear statement about the extensions in the manuscript, although we described a list of substantial extensions to the previous work in the cover letter.

Action: In the revised version, we have followed the suggestion to add clear statements about the differences between this study and the previous studies (i.e. [1] and [7]) in the third last paragraph of Section 1.

The intended changes are as follow:

This paper extends our previous study [1] in five significant ways. First, this paper thoroughly examines the challenges and practical situations related to testing Web services in Section 2.2. It also extensively discusses the limitations of RT, Partition Testing (PT), and Random Partition Testing (RPT), when they are used for testing Web services in the third 3 to 7 paragraph of Section 1. Second, the previous paper provided a coarse-grained framework for DRT of Web services. PT was not studied in the conference paper. In contrast, the current submission provides a comprehensive solution based on partitioning (Section 4.4.1). Third, this paper provides guidelines for setting the DRT parameters, based on a theoretical analysis (Section 3.2). Such guidelines, which were not covered in the conference paper, are crucial to enhance the practical application of DRT. Fourth, to evaluate the fault detection effectiveness of the proposed approach (DRT), in terms of the F-measure and T-measure, the conference paper used only two small Web services (ATM Service and Warehouse Service), comparing the performance with RT. This paper, in contrast, contains a more comprehensive evaluation that not only evaluates the fault detection effectiveness of the proposed approach in terms of the F-measure, F2-measure, and T-measure (Section 5.1), but also evaluates its execution overheads in terms of F-time, F2-time, and T-time (Section 5.3). Besides, we used three real-life Web services as subjects, and compares the fault-detection effectiveness and execution overhead of the proposed approach with those of both RT, RPT, and AT. Furthermore, we made use of statistical analyses verifying the significance of the empirical evaluations and comparisons (Sections 5.1 and 5.3), which were not covered in [1]. We also analysed the relationship between the number of partitions and the optimal control parameter settings for DRT, evaluating the usefulness of guidance provided by the theoretical analysis (Section 5.2), which was not covered in [1]. Fifth, in Section 6, we substantially extended the literature compared the [1].

The paper [7] is an abstract, which is the first paper proposed DRT. In [7], Cai et al. mainly introduced the principle of DRT, and evaluated the fault-detecting effectiveness of DRT through case studies. This paper made use of the high fault-detecting effectiveness of DRT and proposed a framework for testing web services based on the principle of DRT and the features of web services. Besides, we provided guidelines for setting the DRT parameters, based on a theoretical analysis, and conducted empirical study to evaluate the usefulness of guidance.

***R3C3: The cover letter describes a set of improvements concerning the writing (points (i), (ii) and (vi)) and some other major extensions concerning the presentation of the framework, the definition of guidelines about parameters settings in DRT and a more thorough empirical evaluation. From my point of view, the most consistent improvement is the evaluation part. The other ones need more clarification, as described below***

Response: Thanks for endorsing our empirical study. We also agree that others suggestions or comments about our paper, and the details of responses are presented in R3C4, R3C5, R3C6, and R3C7.

Action: None.

***R3C4: Section 3 describes the application of DRT to web services. The novel part described in Section 3.2 needs some rewriting. I understand the importance of parameters setting and the need of mathematical treatment, but as it is now it is not easily understandable. A high-level description of the procedure and of the findings is necessary, together with a description of the followed procedure. The mathematical demonstrations and theorem should be moved to an Appendix. At present these details disrupt the reading flow and at the end of Section 3.2, it is not clear how to practically set the parameters.***

Response: Thanks for the suggestion. We agree with the reviewer that removing the proof will increase the readability of the paper.

Action: In the revised version, we have moved the proofs and other demonstration to the Appendix, and added detailed descriptions on the usage of our guidelines for setting parameter.

A: Add a high-level description of the parameter settings

In order to satisfy Formula 5, we first analysis the relation between the p\_i^{n + 1} and p\_i^{n}, then we obtained the following theorem by theoretical derivation (the details of process can be found in Appendix).

Theorem 1. For failure rate \theta\_{min} = min{\theta\_1,…,\theta\_m}, \theta\_M > \theta\_{min}, if 0 < \theta\_{min}< 1/2, the following condition is sufficient to guarantee that p\_M^{n + 1} > p\_M^{n}:

(Note that it is not easy to type this formula here, hence I used a picture to present the formula.)

B: Practical guidelines of parameter settings

Note that if the failure rate is too high, any reasonable test strategy will detect faults easily. DRT aims to improve on both RT and PT, which is suitable to use in the situation where the faults are hard to be detected. Therefore, we assume 0 < θ\_{min} < 1/2 in Theorem 1. On the other hand, to obtain a upper bound (E\_{upper}) of E, we further assume that θ\_M > θ\_{min}, and1/2< θ\_M < 1 (The specific inferential process is presented in Appendix), which cannot be satisfied in some situation. However, When the value of θ\_M becomes smaller, the value of E\_{upper} should also become smaller, that is

where E\_{lower} is the lower bound of E. In practice, we can just set



***R3C5: Section 3.3 describing the prototype should be expanded. Now it looks similar to the description in [1], while I would expect a more detailed description here together with the possibility to experiment with the tool for the sake of replicability.***

Response: Thanks for the suggestion. Originally, we provided a very limited description of the prototype due to the page limitation. We agree that we should expand our prototype and add more detailed description of prototype.

We have developed a new prototype called DRTester. Compared to the previous prototype, DRTester has an interface called Handbook that guides user to use this prototype and has a “Configuration” interface where user can follow the handbook to set parameters, construct partition, and generate test cases. Besides, DRTester also provides an “Execution” interface where the user can obtain information about the execution process of WSUT and download the testing results.

Accordingly, we have added more details about DRTester in Section 3.3.

Action: In revised version, we propose a totally different prototype that is easier to use and more flexible. Accordingly, we have added more details to describe our prototype.

***R3C6: Section 4 reports the empirical studies conducted to evaluate the performance of the proposed method. A set of research questions are described and then answers in the results section.*** ***The authors may consider the possibility to*** ***anticipate the research questions as a way to motivate the paper in the Introduction or in a section describing the research approach followed in this paper.***

Response: Thanks for the suggestion. We agree with you that it is better to make research questions as a way to motivate the paper in the Introduction.

Action: In the revised version, we have followed the suggestion and restructured the Introduction. In detail, we have added the below content in the third paragraph of Section 1.

Dynamic random testing (DRT) proposed by Cai [7] introduces software cybernetics to the software testing, attempting to update test profile ({<p\_1, s\_1>, <p\_2, s\_2>, … , <p\_m, s\_m>}, where p\_m is the selection probability of partition s\_m, and m is the total number of partitions) during the test process: If a test case from a partition s\_i reveals a fault, the corresponding p\_i will be increased by a constant \epsilon; otherwise, decreased by \epsilon. This paper proposed a practical framework based on the principle of DRT and the features of web services. The following fundamental questions will be addressed through a series of experiments.

1. How effective is DRT at detecting web service faults?
2. How do the number of partitions and the DRT parameter ε impact on the failure detection effectiveness and efficiency of DRT?
3. What is the actual test case generation overhead when using the DRT strategy?

Accordingly, we have also changed the way to describe our questions in Section 4.1. The below content replaces previous questions.

In our experiments, we investigate the effectiveness and test case generation overhead of DRT, and the impact of the relationship between the number of partitions and parameters on the failure detection efficiency and effectiveness of DRT, which help to answer the proposed fundamental questions.

1. To investigate whether DRT can achieve better failure detection effectiveness over other strategies in actual testing for web services, we conduct three experiment to compare the effectiveness of RT and RPT in terms of F-, F2-, and T-measure.
2. To investigate the impact of the relationship between the number of partitions and parameters on the failure detection efficiency and effectiveness of DRT, we conduct three experiments where we compare the performance between DRT with theoretically optimum parameter value and selected 13 parameters values in terms of F-, F2-, and T-measure.
3. To investigate how much computational cost will DRT incur in actual testing task, we record the time consumptions on the test case generation process of all testing strategies and web services. Then we discuss the time overhead of DRT by comparing their computational costs.

***R3C7: The empirical study itself is quite interesting. I would suggest adding a subsection summarising the results and possible limitations discovered during the experimentation. Besides, since the three different case studies have more or less the same dimensions in terms of LOC, it would be nice understanding the scalability of the proposed approach.***

Response: Thanks for the suggestion. We agree with the reviewer that adding the subsection (Section 5.4) to summarise the results and possible limitation would improve the quality of our paper.

As for the scalability of our proposed approach, in our previous study (titled Adaptive Partition Testing) and recent work (H. Pei, K. Y. Cai, et al. Dynamic Random Testing: Technique and Experimental Evaluation, IEEE TR, 2019, 1-21), the experiment results has shown that DRT outperformed RT and RPT in terms of T-measure on several real and large programs (LOC > 5K), such as grep, gzip, and make.

We have confidence that my method would have a good performance on large-scale web services, even if we can't verify this now: To the best of our knowledge, there are not such subjects in the field of Web services. Furthermore, our evaluation needs to access the source code of Web services in order to seed faults. However, the owner of realistic Web services is not willing to make the source code accessible since the implementation of a Web service involves commercial interests or technical secrets.

Action: In the revised version, we have added a new subsection (namely Section 5.4) to provide a summary of the results and the possible limitations.

Intended results and limitations are as follows: (in Section 5.4)

5.4 Summary

5.4.1 Results

Web service testing can be more challenging than traditional software testing, due to the unique features of SOA, such as lock of access to service implementation and lock of control. Adaptive software testing aims to control the process of test, and several adaptive testing techniques has been proposed [7-9]. DRT is one of adaptive testing technique, which has been found to present an improvement over RT and RPT. In this paper we proposed a framework to test web services based on the principle of DRT and the features of web services, and developed a prototype called DRTester that partially automates our method. In order to make DRT easier to use, we introduce the parameter setting method by theoretical analysis. Furthermore, we conducted empirical studies where DRT is used to test three real-life web services. The results shown that DRT outperformed RT and RPT in terms of F-, F2, and T-measure with a lower test case selection overhead, and DRT with theoretical optimum value is confirmed to be more effective.

5.4.2 Limitations

Several investigations and evaluation have been reported to improve and verify the failure detection effectiveness of DRT. However, there still exist some deficiencies. DRT is proposed based on the observation, that is the fault-revealing inputs tend to cluster into “continuous regions”, which also highlighted the existence of the so-called Pareto rule. In practical, there are some cases where the faults are not cluster into together, that means test cases that can detect the faults are more evenly spread and far separated from each other. In such scenarios, the failure detection effectiveness of DRT is not verified.