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. $P_1=P_2=\ldots=P_k=1/k$, where k denotes the number of partitions, and P_i ($i=1..k$) denotes the probability of the i^{th} partition). As a result, our technique only requires the user to

provide additional information on the initial test profile, and such additional information is 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.

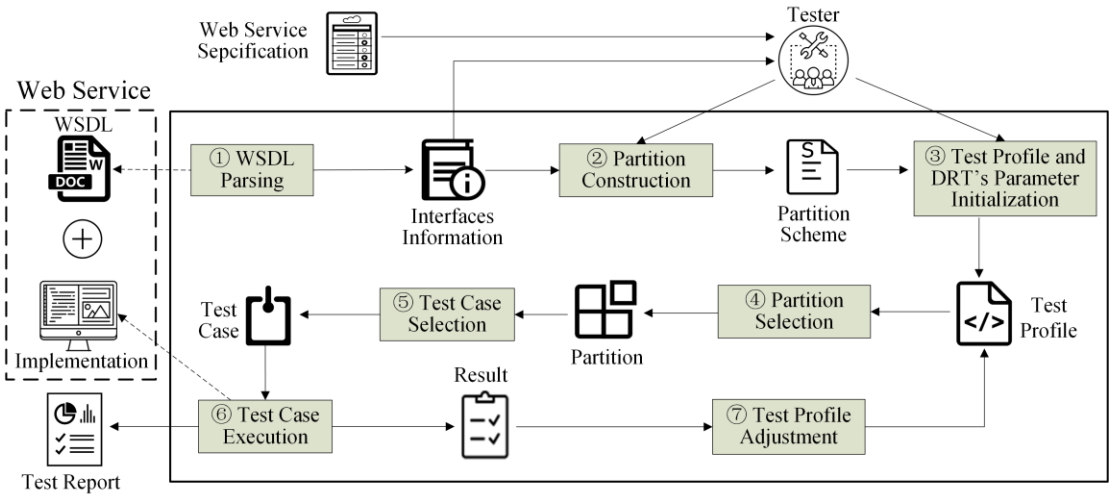
Action: None.

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: We feel sorry that human interactions are missing in the framework in Figure 1. The interaction mainly includes partition scheme construction and the settings of initial test profiles and DRT parameters.

Action: In the revised version, we have followed the suggestion and added the human interactions in Figure 1. Accordingly, we have changed the description of our framework in Section 3.1.

Intended details and updated framework are as follows:



Test Case Selection: DRT selects a test case from the 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. Indeed, the original description is very thin and very similar to the one in the conference. This is mainly due to the page limit. We decide to follow the suggestion to add more details of prototype. Accordingly, we have made substantial efforts to enhance the previous prototype reported in the conference paper. The enhancement includes the following: (1) a feature (for guideline) is implemented to guides the user about the usage of this prototype; (2) a feature (for configuration) is implemented for setting DRT parameters, partitions, and test case generations; (3) a feature (for execution) is implemented to indicate information about the execution process of WSUT and download the testing results.

Action: In revised version, we have updated the description of enhanced prototype in Section 3.3.

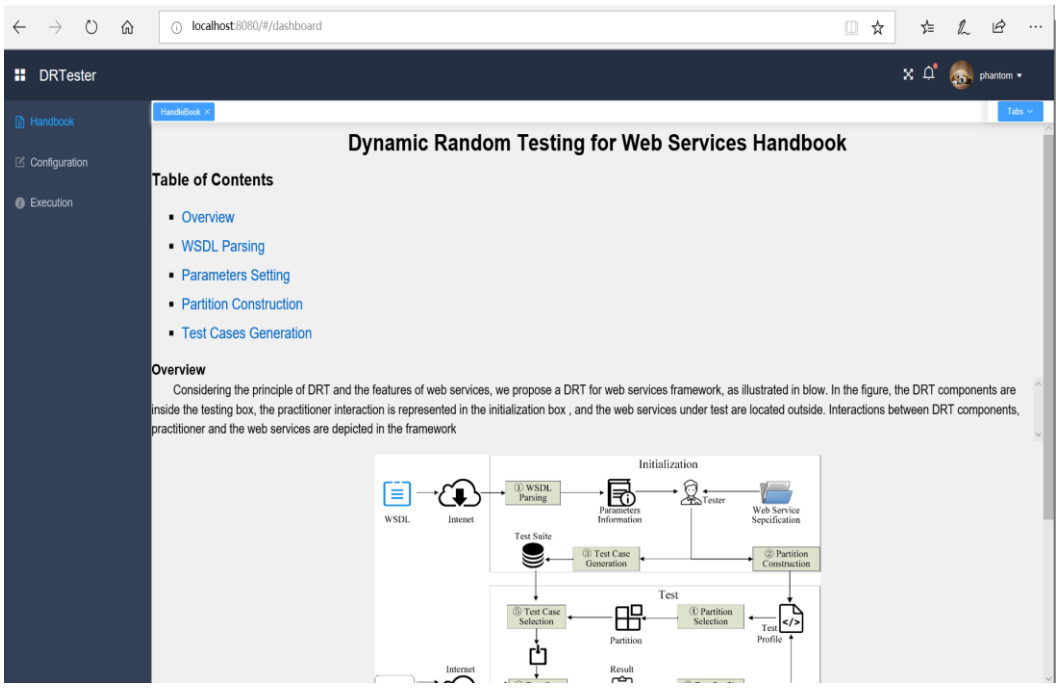
Intended details are as follows [这些界面设计太粗糙，无法使用!!!]:

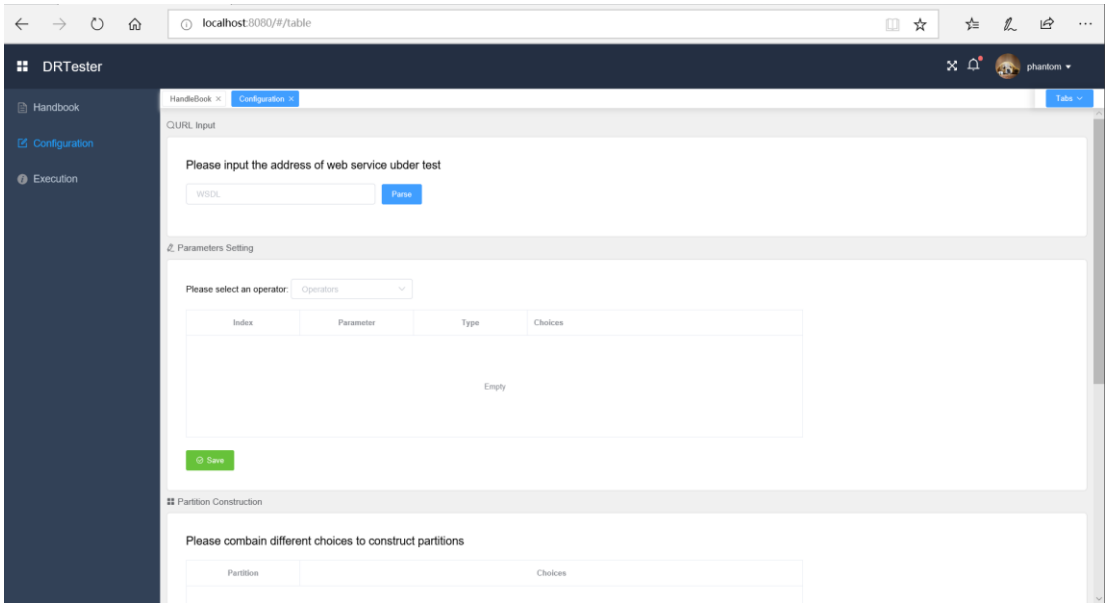
(1) Guideline. This feature describes the steps the tester should follow when testing a web service.

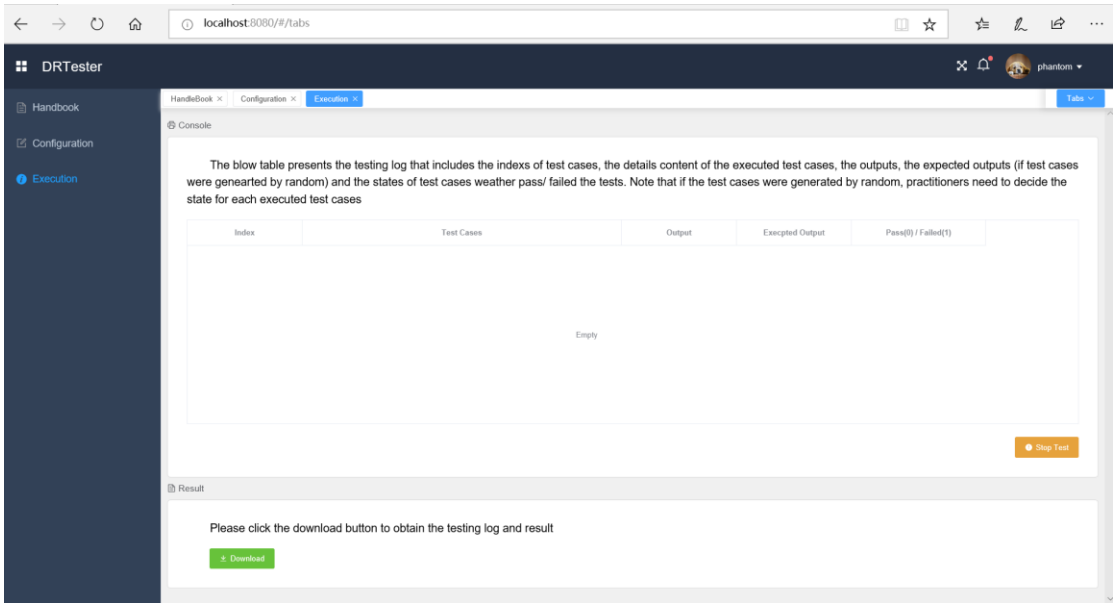
(2) Configuration. This feature implements the interaction with testers in order to obtain or set the information related to web services testing, including the address of web service under test, DRT parameters and partitions, and test case generations. Detailed configuration steps are as follows:
   - Inputting and parsing URL: We integrate the feature of WSDL parsing previously provided by MT4WS (*C-A. Sun et al. MT4WS: an automated metamorphic testing system for web services, IJHPCN 9(1/2): 104-115, 2016.*). In this way, all parameters and their types of the WSUT's WSDL are automatically obtained.
   - Parameters setting: The tester is responsible for selecting operations of the current WSUT to be tested, and partitioning each parameter into disjoint choices.
   - Partition setting: The tester is responsible for specifying partitions though combining choices associated with each parameter.
   - Test case generation: The tester is responsible for specifying the mode of test case generation, namely, either randomly generating test cases based on the parameters, or uploading test cases generated using other techniques.

(3) Execution. This feature demonstrates the summary of testing results, including the information of test cases executed (number, input, expected output, belonging partition), and testing result (i.e. pass or fail). If test cases are randomly generated,

the tester has to check the testing result individually. Otherwise, when all tests have been completed, a test report is provided in a file that can be downloaded.







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. In the previous version, the detailed selection process of mutants was not clearly described. Actually, for each mutant, we generated test suites with 50 random seeds and then calculated the average number of test cases that are required to kill this mutant. In other words, our treatment DID consider the multiple repetitions to avoid the randomness. This selection process was manifested in our test scripts available on https://github.com/phantomDai/evidenceDRT.

Action: In the revised version, we have provided more details on the mutant selection for experiments in Section 4.2.

Intended changes are as follows:

[…]

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".

Response: We admit that the test profile initialization was not clearly stated. Test profile initialization can be done by different ways depending on the tester's knowledge on software under test (SUT). In the experiments, we used the uniform probability distribution as the initial test profile.

Action: In the revised version, we have clearly reported test profile initialization used in our experiments in Section 4.4.2.

Intended details are as follows:
Add the following sentence in the end of Section 4.4.2 "In our experiments, we used the uniform probability distribution as the initial test profile, and the initial test profiles of each web services are summarized in Table XXX, where <S_i, P_i> represents the probability of partition S_i is P_i . ".

Table XX Summary of Initial Test Profiles of Subject Web Services

Web service	Number of partitions	Initial test profile
ACMS	24	$\{< s_1, \frac{1}{24} >, < s_2, \frac{1}{24} >, \ldots, < s_{24}, \frac{1}{24} >\}$
	7	$\{< s_1, \frac{1}{7} >, < s_2, \frac{1}{7} >, \ldots, < s_7, \frac{1}{7} >\}$
CUBS	20	$\{< s_1, \frac{1}{20} >, < s_2, \frac{1}{20} >, \ldots, < s_{20}, \frac{1}{20} >\}$
	3	$\{< s_1, \frac{1}{3} >, < s_2, \frac{1}{3} >, < s_3, \frac{1}{3} >\}$
PBS	18	$\{< s_1, \frac{1}{18} >, < s_2, \frac{1}{18} >, \ldots, < s_{18}, \frac{1}{18} >\}$
	3	$\{< s_1, \frac{1}{3} >, < s_2, \frac{1}{3} >, < s_3, \frac{1}{3} >\}$

R2C6:- (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.

Response: We agree that it is necessary to describe the decision tables that are used to set the partitions. These decision tables were not reported in the previous version mainly due to the page limit. We report the decision tables used in our experiments accordingly.

Action: In the revised version, we have clearly reported the partitions via decision tables for each Web services before the last paragraph of Section 4.4.1.

Intended details are as follows:
Add the following sentences before the last paragraph of Section 4.4.1: "Accordingly, Tables XX1-XXX shows the decision tables for ACMS, CUBS, and PBS, respectively. In Table XX1, . In Table XX2, …. "

Table XX1 Decision Table of ACMS

Table XX2 Decision Table of CUBS

	rule_1	rule_2	rule_3	rule_4	rule_5	rule_6	rule_7	rule_8	rule_9	rule_10	rule_11	rule_12	rule_13	rule_14	rule_15	rule_16	rule_17	rule_18
typeOfvehicle	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
week	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1
typeOfdiscount	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2
formula_1	☑	☑	☑	☑	☑	☑												
formula_2							☑	☑	☑	☑	☑	☑						
formula_3													☑	☑	☑	☑	☑	☑

Table XX4 Decision Table of PBS

	rule_1	rule_2	rule_3	rule_4	rule_5	rule_6	rule_7	rule_8	rule_9	rule_10	rule_11	rule_12	rule_13	rule_14	rule_15	rule_16	rule_17	rule_18	rule_19	rule_20
plan	A	A	A	B	B	B	B	B	B	C	C	C	C	C	C	C	C	C	C	C
option	option_A1	option_A2	option_A3	optiob_B1	optiob_B2	optiob_B3	optiob_B4	optiob_B5	optiob_B6	option_C1	option_C2	option_C3	option_C4	option_C5	option_C6	option_C7	option_C8	option_C9	option_C1(	option_C11
formula_1	☑	☑	☑																	
formula_2				☑	☑	☑	☑	☑	☑											
formula_3										☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

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	$(w - 45) * price_0 * 1.5\%$
Formula_2	$(w - 35) * price_0 * 1.5\%$
Formula_3	$(w - 45) * price_0 * 1.5\%$
Formula_4	$(w - 47) * price_0 * 1.5\%$
Formula_5	$(w - 37) * price_0 * 1.5\%$
Formula_6	$(w - 27) * price_0 * 1.5\%$

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 i^Y option of Plan X, where $X \in \{A, B, C\}$, and $Y \in \{R|Y \neq 0\}$). 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	$\text{option} + (c - freeCall) * 0.25 + (d - freeData) * 0.0003$
Formula_2	$\text{option} + (c - freeCall) * 0.20 + (d - freeData) * 0.0003$
Formula_3	$\text{option} + (c - freeCall) * 0.15 + (d - freeData) * 0.0003$

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	$baseFee * 50\%$
Formula_2	$baseFee * (1 - 40\%)$
Formula_3	$baseFee * (1 + 20\%)$

R2C7:- (3) The applications are well described, but there is no reference to where they are taken (open source repository, private repository, …).

Response: Sorry for this omission. The subject Web services were developed in our laboratory based on the real-life specifications.

Action: In the revised version, we have clearly explained where subject applications are from in the beginning of Section 4.2.

Intended details are as follows:
Add the following sentence after the first sentence in Section 4.2: "Note that these web services were developed by ourselves based on the real-life specifications."

R2C8: 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 described in Section 6.2, there are two kinds of related techniques, namely adaptive random testing (ART) and adaptive testing (AT). ART improves the fault detection efficiency of RT through evenly spread test suites in the input domain. AT improves the fault detection efficiency of RT through introducing feedback to the process of RT. We decide to include AT as an additional benchmark technique due to their similar rationale.

Action: In the revised version, we have added AT as an additional benchmark technique in Section 4.3.1 and Section 5.

For discussion: Although we have completed additional experiments with AT. I am still not sure whether this is should added or not. Please suggest.

R2C9: 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: Sorry for the confusing description. We agree that failure rate should be defined as #number of failure/#number of executed tests. However, it is impossible to

follow this definition in our experiments, because all studied web services have countable infinite test cases. In order to get approximate failure rates of each partition, the solution suggested by the reviewer works, that is, the failure rate θ_i of partition s_i is calculated by 1/ k_i (k_i is the total number of test cases performed to reveal a fault). We have checked the results presented in the previous version and confirmed that this solution was actually used. Please refer to the test scripts which are available on https://github.com/phantomDai/evidenceDRT for more evidence.

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.

Intended details are as follows:
Before starting the test, the failure rate θ_i of partition s_i is needed. From Table 2-6, we observe that the values of some parameters (such as the weight of baggage, the minutes of calls, and parking hours) are countable infinite, meaning that result in infinite test cases in a partition. In this situation, we employ $1/k_i$ (k_i is the total number of test cases performed to reveal a fault) to get the approximate failure rate θ_i of s_i.

R2C10: ***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 comment. Indeed, the third research question in Section 4.1 does not match the answer in Section 5.3. Originally, we would like to evaluate the fault detection efficiency of DRT in terms of time cost. Especially, DRT introduces the selection of partitions and test cases in a partition compared with RT and PT. Thus, we are interested in comparing fault detection efficiency of DRT, RT and PT in term of their time cost. We decide to rephrase the third research question in Section 4.1 to fit our original intent.

Action: In the revised version, we have rephrased the third research question in Section 4.1 and the title of Section 5.3 to make the evaluation more reasonable and consistent.

Intended details are as follows:
Changes of Section 4.1:
"RQ3 How efficient is DRT at detecting web service faults in terms of time cost when comparing with baseline techniques?
DRT introduces the selection of partitions and test cases in a partition compared with RT and PT. Naturally, one may be interested in knowing the fault detection efficiency of DRT in terms of time cost. To answer this question, we compare the time cost of DRT with that of baseline techniques."

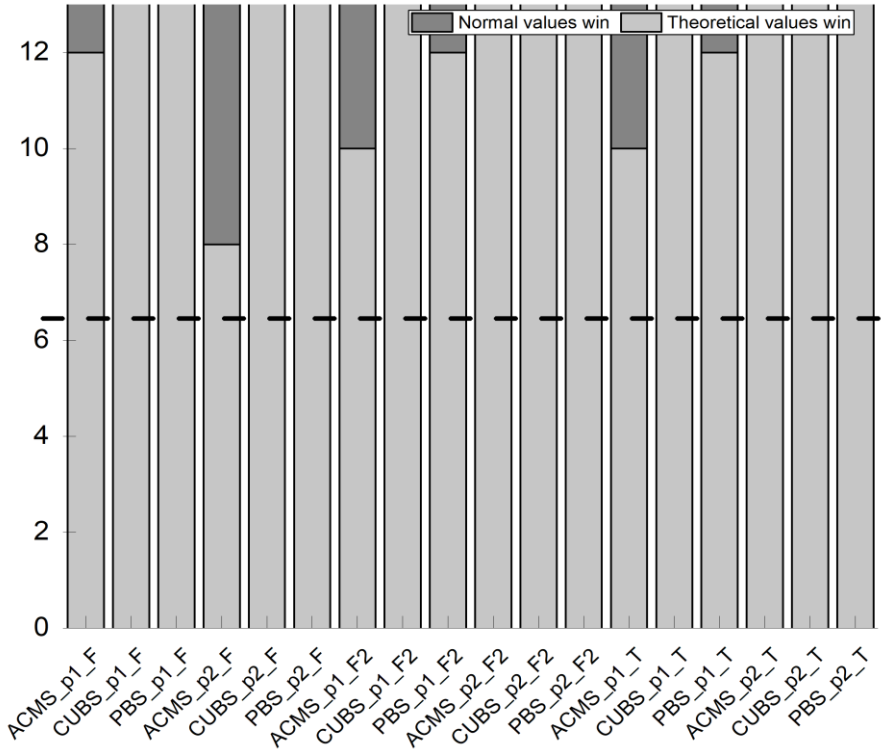
Changes of Section 5.3:
"Section 5.3 RQ3: Fault detection efficiency"

R2C11: ***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 comment. In the previous version, Figures 3~9 were a bit crowded in order to save the space. In particular, some points are overlapping in Figure 6, which make it difficult to distinguish them. To improve the presentation, we decide to take the following actions: (1) Figures 3~9 are extended with more details and scaled better; (2) A stacking bar chart is added to summarize the fitness of experimental results with theoretical analysis; (3) Table 9~11 are moved to appendix.

Action: In the revised version, we have followed the review's suggestion to redraw all figures to improve the presentation. Significant changes happened to Section 5.

The following will be added for summarizing the fitness of experimental results with theoretical analysis:



R2C12: ***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. We agree that it is necessary to add a statement in the Introduction to explicitly state the novelty of this paper with respect to the previous publication. We decide to move those substantial extensions to the previous work provided in the cover letter to here.

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

Replace "we examine key issues … in terms of fault detection efficiency" with the following:

We examine key issues of such an adaptation, and accordingly propose a framework for web services testing through the combination of the principle of DRT proposed in [7] and the features of web services. To validate the fault detection effectiveness and efficiency of the proposed DRT in the context of SOA, we conduct a comprehensive empirical study. Besides, we further explore impact factors of the proposed DRT and provide guidelines for setting DRT parameters based on a theoretical analysis. Finally, we compare the performance of the proposed DRT with other baseline techniques.

Replace "The contributions of this work include" with the following:

This paper extends our previous work [1] in the following aspects. First, this paper extensively examines the challenges and practical situations related to testing Web services (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 (Section 1). Second, the previous work [1] provided a coarse-grained framework for DRT of Web services and PT was not studied. In contrast, this paper provides a comprehensive solution based on partitioning (Section 4.4.1). Third, this paper provides guidelines for setting DRT parameters, based on a theoretical analysis (Section 3.2). Such guidelines are crucial to enhance the practical application of DRT, which were not covered in [1]. Fourth, the previous work [1] only evaluated the fault detection effectiveness and efficiency of the proposed approach (DRT) in terms of F-measure and T-measure, and only two small Web services (ATM Service and Warehouse Service) used to evaluate and compare its performance with RT. This paper, in contrast, provides a 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 efficiency in terms of F-time, F2-time, and T-time (Section 5.3). Besides, we use three real-life Web services as subjects, and compared the fault-detection effectiveness and efficiency of the proposed approach with those of RT, RPT, and AT. Furthermore, we made use of statistical analyses to validate the significance of the empirical evaluations and comparisons (Sections 5.1 and 5.3), which were not covered in [1]. We also examine the relationship between the number of partitions and the optimal control parameter settings for DRT, evaluating the usefulness of guidelines provided by the theoretical analysis (Section 5.2), which was not covered in [1]. Fifth, we substantially extended the literature compared the [1] (Section 6). The contributions of this work, together with the previous work [1], include:

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 the comments. We have made more clarifications as suggested. For more details, please refer to our responses to 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. Indeed, a high-level description of the procedure and the findings is helpful to improve the readability. Accordingly, we provide a general clue for better understand how the findings are achieved, and at the same, we move the proof to the appendix in order to avoid disrupting the reading flow. We examine each element in the parameter setting and provide further guidelines for them individually.

Action: In the revised version, we have added a high-level description of the findings and moved the proofs of lemma and theory to the appendix. We have also provided detailed guidelines for parameter setting.

Intended high-level description of the parameter settings:
In order to achieve the best performance, the probability of partition s_M that has the maximum failure rate being selected is expected to increase. To achieve this goal, the probability increase of s_M being selected for the next round should be larger than that of other partitions. The further analysis of sufficient condition can result in an interval of ε.

Intended practical guidelines of parameter setting:
For a given partition scheme, we can get the number of partitions, namely m. To get θ_{min}, we need to first figure out failure rates of all partitions and then get their minimum. The failure rate of each partition can be obtained through two ways: (1) The failure rate can be directly defined as #number of failure/#number of executed tests in case the test history of web service under test is accessible; (2) The failure rate can be approximately calculated by $1/k_i$ (k_i is the total number of test cases performed to reveal a fault).

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 comment. Indeed, the original description is very thin and very similar to the one in the conference [1]. This is mainly due to the page limit. We decide to follow the suggestion to add more details of the prototype and further improve the prototype. Accordingly, we have made substantial efforts to enhance the previous prototype reported in the conference paper. The enhancement includes the following: (1) a feature (for guideline) is implemented to guides the user about the usage of this prototype; (2) a feature (for configuration) is implemented for setting DRT parameters, partitions, and test case generations; (3) a feature (for execution) is implemented to indicate information about the execution process of WSUT and download the testing results.

Action: In revised version, we have updated the description of enhanced prototype in Section 3.3. [Note: The reviewer also reminds the description of the tool in the experiments.]

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. Originally, we described research questions in the *Empirical Study* section and answers in the *Experimental Results* section, in order to make the presentation more structural. We agree that moving the research questions to the Introduction may help improve the logic of the study. Accordingly, we decide to embed research questions in the Introduction as goals to motivate the work in this paper. Please also refer to our response to R3C2.

Action: In the revised version, we have followed the suggestion to restructure the Introduction, and added research motivation before presenting the contributions in Section 1.

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. Adding the subsection to summarise the results and possible limitation would improve the presentation. Our approach is a kind of black-box testing technique which is an enhanced version of random and partition testing techniques. Considering the fact that random testing and partition testing can be widely applied in practice, we believe that the proposed approach is able to be applied to larger subject applications. The current evaluation only involves small-size web services due to the fact that our evaluation needs to access the source code of web services in order to seed fault, while our approach itself has not such limitation.

Action: In the revised version, we have followed the suggestion to add a new subsection (namely Section 5.4) to summarize the results and possible limitations.

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

5.4 Summary

Summarize observation from empirical study:….

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.

Limitation observed from the empirical study:

(1) improvement over random testing …cost…partitions…manual way;

(2) DRT application's limitation … pay addition efforts … parameter settings, (theoretic analysis), and test profile settings…