Detail answers to reviewers' questions

A. Answer to R1Q1: Aims and contributions of this paper.

Taking a specific goal in mind is important. In our context, we try to explore which metric is the most accurate for scoring the effectiveness of a test suite. Scoring a test suite is a common scenario. For example, it can be used to assess the adequacy of the current testing or to compare the effectiveness of multiple test case generation tools. In this scenario, we will have many metrics at hand to evaluate (e.g., various coverage metrics and mutation score metrics). A natural question is: **which metric is the most accurate for depicting the effectiveness of test suites in detecting defects**? Our study focuses on the fundamental question of how to (accurately) evaluate the quality of the proxy metrics for test suite effectiveness (i.e., MTE). Fig. 1 shows the relationships among source code, test suites, MTEs, and metric evaluation. As can be seen, our standpoint is at the step marked in red: "metric evaluation". Our study proposes a methodological framework called ASSENT, which provides a standard paradigm and guideline for following up studies to consistently and accurately evaluate and compare the quality of MTEs.

Figure 1. Relationship diagram. The word in red is the key part.

There has been a great deal of research on metric evaluation. But we have researched a large number of related studies and found that the conclusions are often quite different. For example, in [41], there is a **weak** correlation between mutation detection ratio and real fault (i.e., mutation score is inaccurate) while in [34], there is a **strong** correlation between mutation detection ratio and real fault (i.e., mutation score is accurate). We can't help but ask "are some conclusions wrong?" Or "**are there some factors that lead to the different conclusions?"** If the latter is the case, can we sort out the key factors? This is the motivation of our study. Recall that the question we are trying to answer is "which metric is the most accurate?" Therefore, **the first step should be to define "accurate" i.e., the ground truth**. Let's take an analogous example: students are asked to each make a ruler, and we want to find the most accurate ruler among them. Naturally, each student will claim that "my ruler is the best". So the teacher must first pull out a "metre" ruler when assessing other rulers. This "metre" ruler is accepted by everyone for its accuracy. Then a specific item is selected and measured first by a "metre" ruler to get a standard length. Then all students are asked to measure this item. Finally, the ruler providing the value closest to the standard value is the most accurate ruler. Analogously, you are concerned with the selection of items (analogously in [39], they suggested that we should not use a randomly selected group of rabbits with the same weight as the items for length measurement). Indeed, this is a very important issue. But it is not the first thing. The first thing is the "metre" ruler (i.e., the ground truth). A large number of studies have ignored this issue (i.e., some "metre" rulers are not accepted by everyone for its accuracy). We point out this problem, and then we propose a framework. The framework contains three elements: ground truth ("metre" ruler), benchmark test suite(item), and agreement indicator (value difference). This constitutes the most important contribution of our study.

In the following, we further explain this problem from the viewpoint of measurement theory. According to measurement theory, a metric is a map from the test suite space to a numerical space. As shown in Fig. 2, in the ideal case, the metric should maintain

the partial order in the test suites space if the MTE *m* is a "valid" metric (i.e., satisfying the representation condition). Here, the partial order denotes the ground truth relation "is more effective in detecting defects" between test suites.

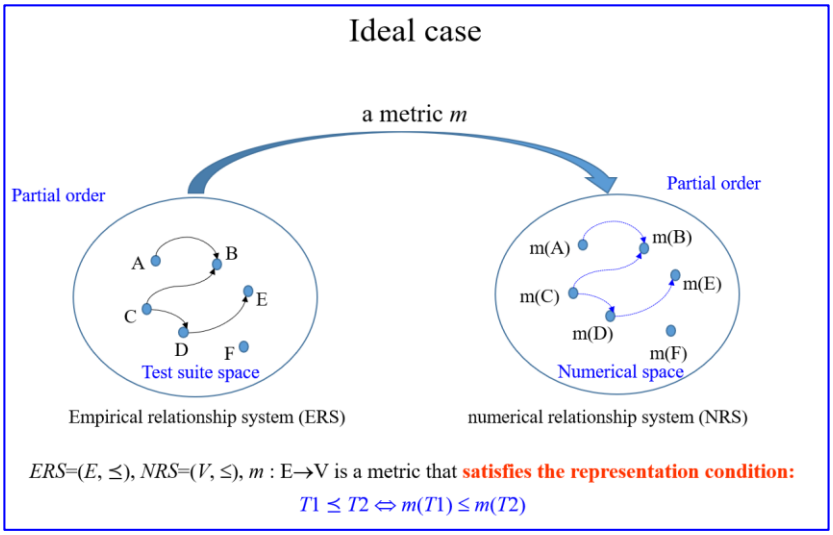


Figure 2. Ideal case: a valid metric should be a metric satisfying the representation condition

In practice, however, the ideal case does not exist (as shown in Fig. 3). The reasons are mainly two-fold. On the one hand, the MTE *m* may provide a total order in the numerical space. On the other hand, the MTE *m* may twist the original order (e.g., B->C).

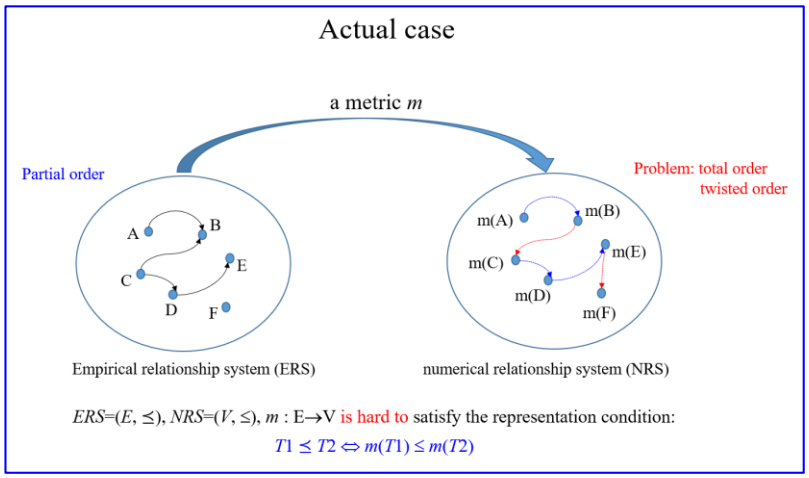


Figure 3. Actual case: a metric is hard to be "valid" due to the occurrence of total order and twisted order in NRS

Then, what should we do if we still want to apply the MTE to practice? As shown in Fig. 4, the best we can do is to evaluate to what extent the MTE is valid, rather than whether the MTE is valid or not. In nature, as shown in Fig. 5, this requires an evaluation to what extent the ground truth relations in test suite space is preserved in the numerical relations in numerical space. The more relations preserved, the higher probability of the MTE being a "valid" metric (i.e., the agreement). Of a number of available MTEs, the MTE with the highest agreement would be the most "accurate" metric. This is the rationale behind the evaluation of test suite effectiveness metrics from the viewpoint of measurement theory.

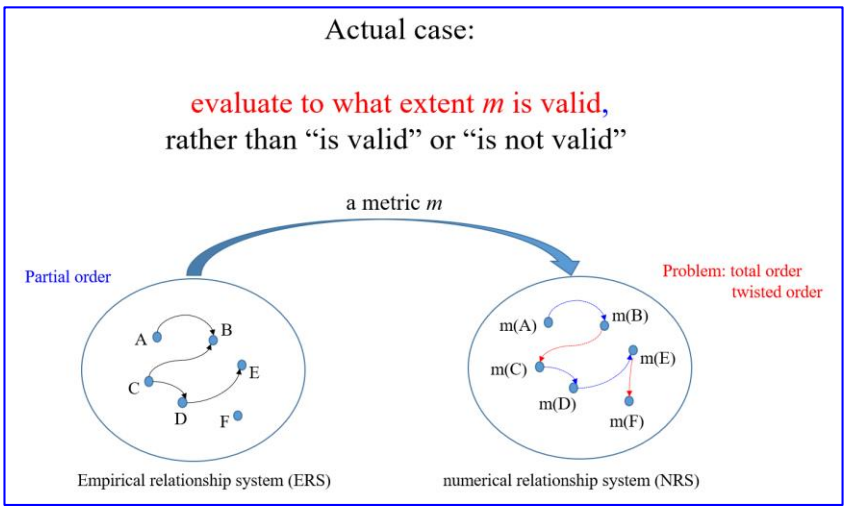


Figure 4. Actual case: we should evaluate to what extent a metric is "valid"

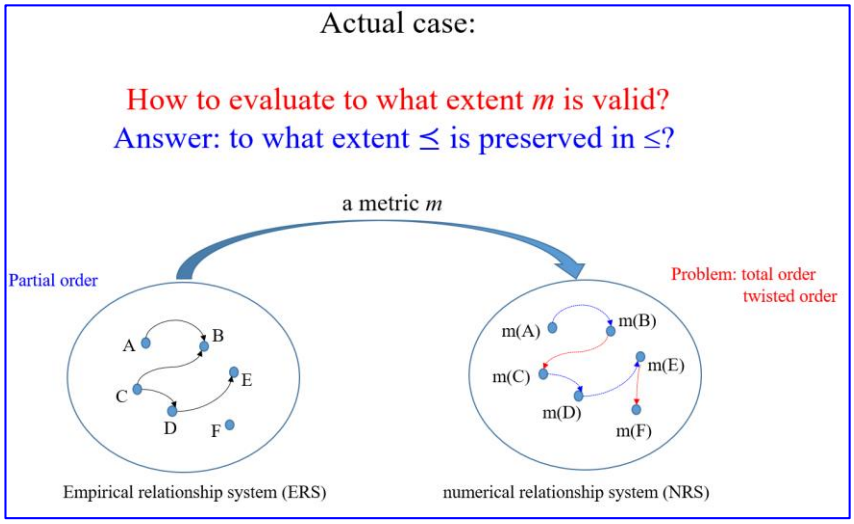


Figure 5. Actual case: we should quantify the agreement between the ground truth and MTE relations

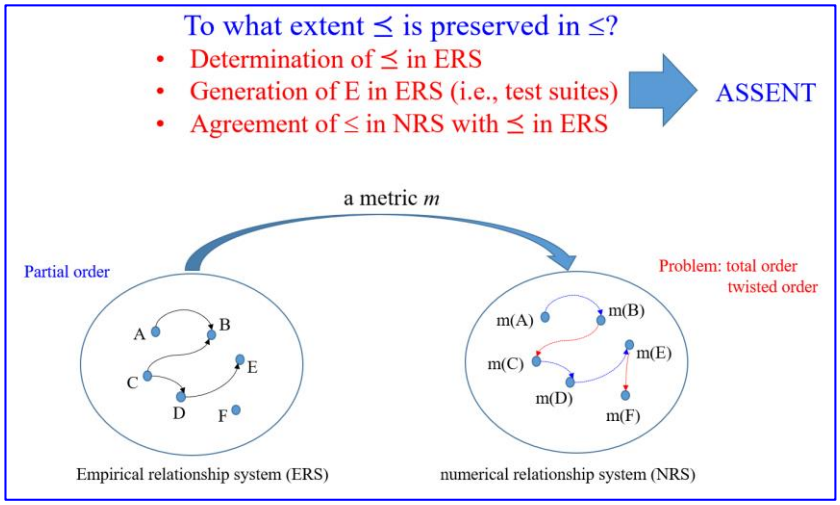


Figure 6. The elements of the framework ASSENT for test suite effectiveness metric evaluation

To enable the above evaluation, we can see (from Fig. 6) that three elements are involved. The first element is how to define the

ground truth relation, the second element is how to generate test suites for evaluation, and the third element is how to assess the degree of agreement between the ground truth relations and the numerical relations produced by the MTE. The main contribution of our study is that we encode these elements in a methodological framework called ASSENT. With ASSENT, the following up studies are able to consistently and accurately evaluate and compare the quality of MTEs, regardless of whether the existing MTEs or new MTEs are considered.

B. Answer to R2Q1: discussion of recommendations and implications.

Recommendations. We mainly provide two recommendations:

- The first is, if one tries to evaluate a test suite effectiveness metric, he can use ASSENT as the guideline. First, define the ground truth. Then, generate test suite pairs for comparison based on the ground truth. Finally, calculate the agreement indicator to evaluate the metric. The benefits are two-fold. On the one hand, it is easy to compare with other related work. On the other hand, it is clear for readers to understand the work and find the potential risks.
- The other recommendation is that when one tries to quote or refute some conclusions (e.g., one wants to quote that there is a strong correlation between mutation detection ratio and real fault), he cannot simply pick the conclusions of some literature and quote them directly. He must pay careful attention to whether the three elements that produce the conclusion are applicable in your scenario. If not applicable, for example, some ground truths cannot be widely recognized in his scenario, he should not quote that conclusion.

Implications. Our study has important implications for both researchers and practitioners. The detailed implications are listed as follows:

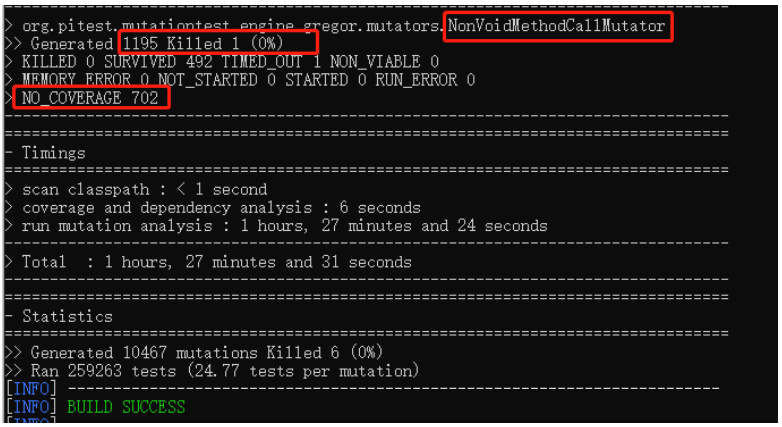
- We provide a framework that can help researchers to investigate the test effectiveness metrics effectively. Meanwhile, it is easy to compare all the related work by our framework.
- We analyze the possible threat caused by representative ground truths. On the one hand, the threat of "real fault" is analyzed in Section 3.1.1. On the other hand, the threat of "mutant" is analyzed in RQ2. When the researchers are going to cite the corresponding conclusions, to eliminate the threat, not only they must pay attention to whether the three elements are consistent with the cited paper or not. But also, they should explain how the pitfalls we point out will influence their work.
- For the practitioners, we help them to rebelieve that using MS or SMS is still the least risky metric among the MTEs. Using other metrics, there will be a higher risk of ignoring the enhancement of test suite effectiveness.

C. Answer to R3Q2: Benchmark and automatically generated suites.

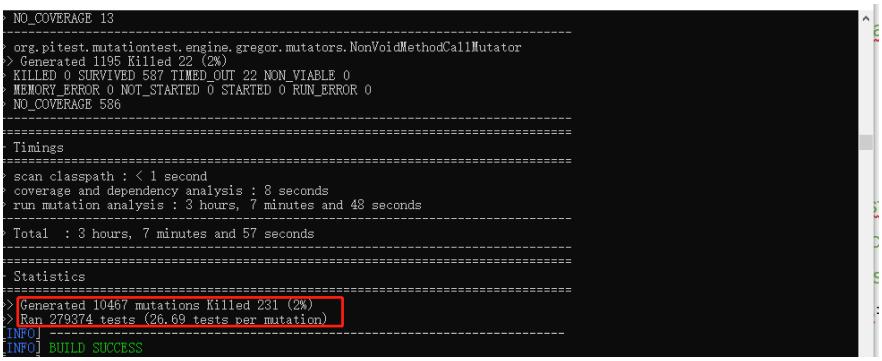
Benchmark is a key aspect of the problem. In the previous manuscript, we use other projects in apache.commons as a supplement in the discussion. Meanwhile, another mutation testing tool (PIT) is used. However, as reviewer B said, "the paper is trying too hard: on the upside it makes many contributions, but since space is limited some of them are presented too concisely." After careful reflection, we decided that, in the current version, we should still focus on clarifying the core issues of the test suite effectiveness metric evaluation. However, this suggestion is constructive. **During these days, we have tried the tools for automatically generating test suites.**

We first **applied *Evosuite* with the default configuration** (one of the state-of-the-art tools for automatically generating test cases) to one project *pool* (org.apache.commons.pool2). After that, we ran the resulting test suite against the mutants generated by PIT. We found that, although it achieved an acceptable coverage, **the mutation score was very low**: as shown in the following picture, only 6 out of 10467 mutations were killed. At the beginning, we suspected that we generated a large number of test cases that cannot cover mutants. But, after examination, we found that coverage might not be the cause. For example, quite a few (i.e.,

1195 - 702 = 493) of the mutants generated by the mutator “NonVoidMethodCallMutator” were covered while the mutation score was very low.



Clearly, it is inappropriate to conduct our experiments on projects with too few killed mutants. In order to address this problem, **we modified the configurations for *Evosuite*. Specifically, we set the criterion as “MUTATION” and used other default configurations when generating test cases.** Consequently, **the mutation score has been improved. However, it is still too low**:



Of course, this may be due to our experiments on specific projects. To this concern, we used the project *spark* listed in the website of *Evosuite* (https://www.evosuite.org/experimental-data/github-subjects/) for a try:

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
88	64%	1993/3135	12%	1612/13766

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
spark	21	46%	492/1067	18%	506/2867
spark.embeddedserver	4	29%	6/21	17%	11/64
spark.embeddedserver.jetty	6	58%	94/163	45%	286/632
spark.embeddedserver.jetty.websocket	5	67%	31/46	86%	96/111
spark.globalstate	1	100%	5/5	100%	11/11
spark.http.matching	11	61%	198/326	30%	252/854
spark.resource	8	74%	243/330	0%	2/2542
spark.route	5	96%	163/169	5%	40/825
spark.routematch	1	100%	14/14	0%	0/15
spark.serialization	5	93%	37/40	8%	7/86
spark.servlet	2	71%	46/65	45%	133/298
spark.ssl	1	100%	18/18	86%	48/56
spark.staticfiles	4	73%	123/169	25%	83/332
spark.utils	10	74%	316/425	5%	137/2801
spark.utils.urldecoding	4	75%	207/277	0%	0/2272

However, as indicated by the mutation score (i.e., MS = 0.12), the generated test suite still has a low strength (it is worth mentioning that the coverage is relatively high). Up to now, we are unable to obtain automatically generate test suites with a high strength.

Due to the time limitation, we only conducted the experiment on one project *pool* (org.apache.commons.pool2). In the future, we will do an extended work, including the investigation of multiple test case generation techniques with multiple mutation testing tools. By *EvoSuite*, we set the criterion as "MUTATION" and used the other default configurations. Meanwhile, we delete the surviving mutants and use the remaining mutants for evaluation. This makes the current test suite locally strong. In other words, we use only those mutants that have been considered by the test cases. This is a common approach when we do not have access to a sufficient test suite. Table 1 reports for 4 metrics the average OP. We can see that SMS is the best.

Table 1. The average OP of several metrics under alternative ground truth

	OP			
Project	RMS	COS	SMS	CMS
Pool	0.923	0.912	0.965	0.959