InterEvo-TR: Interactive Evolutionary Test Generation with Readability Assessment
— *Extended report* —

Pedro Delgado-Pérez[*], Aurora Ramírez[†], Kevin J. Valle-Gómez[*],
Inmaculada Medina-Bulo[*] and José Raúl Romero[†]

October 7, 2022

1 Experiment #1: Parameter study

1.1 Goals of the analysis

In this report, we want to study the influence of the interaction scheme on the approach results to select appropriate values for some of the interactive-related parameters. Namely, we evaluate the following three parameters:

- *Percentage_to_revise*: Percentage of test cases in the population the user is willing to revise at most.
- *Max_targets_interaction_moment*: Maximum number of targets the user is willing to address in one interaction moment.
- *Max_times*: Maximum number of times the user is willing to interact during the search.

More specifically, we aim at analyzing (1) the influence of the parameter *Percentage_to_revise* in the total number of test cases that the tester observes during the execution, (2) the influence of *Max_targets_interaction_moment* in the number of different targets and different methods addressed by the tester, and (3) the influence of *Max_times* in the previous two measures (tests and targets/methods seen). Also, this experiment allows us to observe whether the number of interactions set (*Max_times*) are actually consumed with different parameter settings.

Additionally, this study allows us to know whether using the preferred test cases as a source to generate other test cases in the evolution has an impact on the effectiveness of the generated tests, that is, whether the interactive approach achieves a comparable coverage level than the fully automated version.

1.2 Experimental procedure

To achieve these goals, we designed a simulated process of interaction in which the readability score is computed from some readability factors (e.g., character count and number of identifiers) [1] measured over the test cases. A simulated interaction provides a controlled environment to analyze algorithm behavior, and is frequently found in other interactive search-based

[*]University of Cádiz, Spain. Email: {pedro.delgado, kevin.valle, inmaculada.medina}@uca.es
[†]University of Córdoba, Spain. Email: {aramirez, jrromero}@uco.es

software engineering studies before conducting experiments with human participants [2, 3]. In this way, each test case selected for revision in the interactions is assigned a readability score (mean of readability factors) without the need of human intervention.

We run the simulated version of InterEvo-TR using predefined values for the following parameters: population (50), coverage criteria (Line, Branch and Weak Mutation), number of interaction moments (6), *Revise_after_percentage_coverage* (50%), *Enable_secondary_objective_after* (10) and *P_preference_selection* (20%). Also, the default value for the three parameters under study was 8% for *Percentage_to_revise*, 3 in the case of *Max_targets_interaction_moment* and 10 for *Max_times*. The values for other parameters were, however, specific for each class under test: the search budget for a class was computed as the average number of generations in 30 executions of EvoSuite when setting the search to 2 minutes; consequently, *Revise_frequency* was independently adjusted for each class so that 6 interaction moments were enabled (i.e., dividing the resulting number of generations into 6 equal parts).

The range of values used for the interactive-related parameters under study were:

- *Percentage_to_revise*: 6% (3 tests), 8% (4 tests) and 10% (5 tests).
- *Max_targets_interaction_moment*: 2, 3 and 4 targets addressed (in individual interactions) in the same interaction moment.
- *Max_times*: maximum of 10 and 15 individual interactions.

We run InterEvo-TR 30 times for each of the 10 classes with each parameter setting. At the end, we computed the average number of test cases revised in the interactions, the average number of goals and methods addressed, the average number of interactions consumed and the average coverage level achieved. In order to analyze the coverage level, we collected the coverage achieved by both EvoSuite and InterEvo-TR (both using DynaMOSA) with the same seeds.

1.3 Results

Table 1 shows the results of the parameter setting analysis. Starting with *Percentage_to_revise*, it is clear from the results that the higher the percentage, the more test cases will be revised (#TC). However, the number of additional test cases is far from being proportional to the increase in the value of the parameter. Taking *XMLUtil* as an example, 18.9 test cases were selected with *Percentage_to_revise*=6%. This represents 63% of the maximum of 30 test cases that could potentially have been selected (i.e., 3 test cases for each of the 10 interactions). However, when the value is increased to 8% and 10%, 20.9 out of 40 potential test cases (52%) and 22.4 out of 50 potential test cases (45%) were selected, respectively. This drop in the percentage reflects that it is not always easy for the algorithm to find new minimizations for the interactions. In general, we neither observe a substantial increase in the number of shown test cases in the more complex classes.

Moving on to the second parameter under evaluation, *Max_targets_interaction_moment*, we can observe that the number of different targets addressed (#DT) increases as the value of this parameter increases in most cases. This means that, with 2 individual interactions per interaction moment, the possibility to repeat the same targets is higher than with 3 or 4 interactions per moment. As a remark, the number of different targets was lower in the most simple classes (e.g., around 5 in *Label* and *ATM*) than in the most complex ones (around 7-9 in *CoordSystems* and *DateUtil*); this is an aspect to consider when selecting the parameter setting if we desire to increase the diversity of targets seen in the interactive process. The influence

Table 1: Influence of the parameter setting in the approach results

Class	*Perc_to_revise*		*Max_targ_int_moment*			*Max_times*				
	V.	**#TC**	**V.**	**#DT**	**#DM**	**V.**	**#IC**	**#TC**	**#DT**	**#DM**
Label	6%	21.1	2	5.1	2.2	10	10	22.9	5.3	3.6
(*jipa*)	8%	22.9	3	5.3	3.6	15	15	30.6	8.1	3.8
	10%	25.4	4	7.3	4.6					
ATM	6%	17.3	2	4.7	3.4	10	10	17.9	5.1	3.9
(*tutorial*)	8%	17.9	3	5.1	3.9	15	12.9	24.8	5.7	3.9
	10%	18.2	4	5.5	4.0					
JSJshopNode	6%	17.3	2	9.4	2.9	10	10	17.9	9.7	3.1
(*shop*)	8%	17.9	3	9.7	3.1	15	15	23.8	14.2	3.2
	10%	18.2	4	9.3	4.2					
XMLUtil	6%	18.9	2	6.0	4.0	10	10	20.9	6.9	4.6
(*checkstyle*)	8%	20.9	3	6.9	4.6	15	15	26.5	9.1	5.2
	10%	22.4	4	6.9	5.1					
bcWord	6%	19.4	2	6.8	2.1	10	10	21.7	6.9	3.4
(*battlecry*)	8%	21.7	3	6.9	3.4	15	15	26.7	10.5	3.8
	10%	23.4	4	6.9	4.5					
ArrayIntList	6%	17.6	2	8.0	4.2	10	10	19.3	8.3	4.6
(*commons p.*)	8%	19.3	3	8.3	4.6	15	15	25.5	11.5	5.3
	10%	20.3	4	8.3	5.4					
Player	6%	18.9	2	8.4	5.3	10	10	20.8	8.4	6.1
(*gangup*)	8%	20.8	3	8.4	6.1	15	13.5	23.5	11.8	7.7
	10%	20.9	4	8.8	6.4					
User	6%	20.8	2	7.1	2.0	10	10	22.4	6.8	3.1
(*lhamacaw*)	8%	22.4	3	6.8	3.1	15	15	28.5	9.5	3.3
	10%	23.3	4	7.8	4.2					
CoordSystem.	6%	21,7	2	8.4	4.7	10	10	24.3	9.1	6.0
(*jopenchart*)	8%	24.3	3	9.1	6.0	15	15	29.7	13.0	6.3
	10%	25.8	4	9.6	6.7					
DateUtil	6%	19.1	2	6.9	4.2	10	10	20.3	7.2	4.9
(*caloriecount*)	8%	20.3	3	7.2	4.9	15	15	26.9	10.7	6.0
	10%	20.8	4	8.2	5.9					

of *Max_targets_interaction_moment* is even more evident with regards to the number of different methods seen (#DM). This result shows the desired effect of the implemented strategy in InterEvo-TR to diversify the methods involved in the interactions.

Regarding the number of interactions, in all cases the 10 interactions were actually consumed (#IC). The only exception was the class *Player*, where in a very few executions —with some of the parameter settings— the algorithm could not find not-yet-seen test cases for the covered targets and did not reach the 10 interactions. When the maximum was increased to 15, all the interactions were also consumed, except for the class *ATM* (only 12.9 interactions out of 15 interactions took place on average) and *Player* again (13.5 out of 15 interactions). In general, this result shows the feasibility of the approach (i.e., the interactions could be performed) and that there is still room for further interactions in case the tester desires a greater involvement in most cases. This increase in the maximum number of interactions, from 10 to 15, shows diverse but logical influence in the measures: a greater number of test cases to revise, and more targets and methods addressed during the execution. However, we should not forget that this increase always comes at the cost of a higher revision burden. Although the growth in the number of different targets is significant in most cases, it is not always accompanied by a similar increase in the number of different methods seen, so many of those targets are related to the same methods. For instance, in *User*, the number of different targets increases from 6.8 to 9.5, but the number of different methods remains almost the same (from 3.1 to 3.3). The total number of test cases seen also increases, but the average of test cases seen in the last 5 interactions is lower than in the ten first interactions. Focusing again on *User* as an example, the mean of test cases seen in

each of the first 10 interactions was 2.24, but only 1.22 in the last 5 interactions.

Table 2: Coverage achieved (average of 30 executions)

Class	***Coverage***	
	EvoSuite	**InterEvo-TR**
Label	98.4%	98.4%
ATM	99.1%	99.1%
JSJshopNode	85.7%	85.8%
XMLUtil	73.7%	74.2%
bcWord	97.8%	97.8%
ArrayIntList	87.2%	87.2%
Player	98.6%	98.6%
User	98.3%	98.3%
CoordSystem.	98.3%	98.0%
DateUtil	93.4%	93.4%

Finally, Table 2 presents the percentage of targets covered by EvoSuite and InterEvo-TR (using the default values for *Percentage_to_revise*, *Max_targets_interaction_moment* and *Max_times*). As it can be seen, the coverage is the same in seven classes, and slightly superior for EvoSuite or InterEvo-TR in one and two classes, respectively. It seems that the introduction of the preference archive as a new source may unintentionally alter the test generation process sometimes —positively or negatively. However, given these results, we can conclude that InterEvo-TR does not reduce the effectiveness of the generated test suites with the selected parameter setting and the current configuration for the production of new test cases in each generation.

1.4 Discussion

This report shows the influence of the value of three interaction-related parameters in the number of test cases and different targets/methods shown to the tester. In this experiment, we have included *Max_times* as a parameter under study. However, even though the results suggest that more interactions could be easily performed (either increasing the interaction frequency or the maximum of interactions), setting these parameters actually depends on other factors, such as the fatigue or the level of engagement, and the knowledge and expertise of the tester. It also depends on the perceived complexity of the class under test: a more complex class seems to require more interactions to be able to observe a greater diversity of scenarios. In the most complex classes, the number of different methods and targets addressed by the interactions was higher than in the rest of the classes in general. This suggests that more interactions may still be needed to address some unseen targets/methods or to see new evolved test cases for some of those already addressed.

For the experiment with participants —using *ArrayIntList*, a class of medium complexity— we opt for the values in the middle of the range, that is, *Percentage_to_revise*=8% and *Max_targets_interaction_mom*
In this class, the average number of different targets (8.3) does not increase with a higher value for the latter parameter and, being 4.6 the average number of different methods, participants will have more chances of addressing each of those methods at least twice during the interactions (set to a maximum of 10 to avoid fatigue in the participants).

2 Experiment #2: Test comparisons

As part of Experiment #2, participants were asked to evaluate test cases generated by InterEvo-TR and EvoSuite, as way to analyze whether they perceive differences. Section 2.1 describes the methodological steps to carry out the comparison, including the process to choose the test cases for evaluation. The results of the experiments are included in Section 2.2.

2.1 Methodology

The goal of this part of the experiment is to compare whether participants perceive differences in the readability of test cases generated by our approach, InterEvo-TR, and the original tool, EvoSuite. Given that InterEvo-TR is an interactive tool, requiring a direct interaction with the user, they cannot perform a "blind" execution of each tool. Therefore, the comparison is based on a posterior evaluation of some of the test cases appearing in the test suites produced by each tool. More specifically, the three following comparison scenarios were designed:

- Comparison #1 - Interactive vs Interactive: For a given method of the class under test, the participant evaluates two test cases seen by another participant in his/her execution of InterEvo-TR.
- Comparison #2 - Interactive vs Automatic: For a given method of the class under test, the participant compares two test cases: one was positively evaluated by another participant using InterEvo-TR, whereas the other was automatically generated by EvoSuite.
- Comparison #3 - Automatic vs Automatic: For a given method of the class under test, the participant evaluates two test cases generated by EvoSuite.

The last comparison scenario was included as a control mechanism, so that we can check whether the participants infer they are evaluating results from two different tools or not. The specific details of each comparison are provided in the following subsection. For all comparisons, the participants were asked to conduct the same steps:

1. Open the Java files containing the two test cases. The method under test is indicated in the question.
2. Provide one readability score in the range [0, 10] to each test case.
3. Explain the reasons behind their decision.

An open space for free comments was also available to be filled after replying to the three questions. Finally, some general aspects were taken into account:

- The test cases extracted from the interactions of one participant with InterEvo-TR were not assigned to that participant.
- The three comparisons assigned to a participant refer to three different methods of the class under test.
- The order of the comparisons (i.e., questions) was randomized to prevent any assumption about the procedure.
- The statement of the question did not provide information about the source of each test case under comparison.

- The file containing the test case and the test method inside it were renamed to avoid possible identification of its origin.
- The order of each test case under comparison was randomized, so that two participants do not see test cases in the same order.
- The same pair of test cases for a given comparison could be assigned to several participants, but none of them were asked to evaluate the same three pairs of test cases than another participant.

Next, we describe the steps performed to choose the pairs of test cases for each comparison. The resulting pairs are available in the replication package.

2.1.1 Comparison #1: Interactive vs. Interactive

The goal of this comparison is to check whether participants follow similar evaluation patterns when assigning readability scores to test cases selected by InterEvo-TR. We focus on pairs of test cases whose readability differed significantly for one of the participants. By doing this, we want to observe if other participants also perceive such strong difference in score. Particularly, we follow the next steps:

1. Identify participants whose scores were varied in range (see Figure 4 in the paper).
2. Extract from their interaction log file pairs of test cases evaluated in the same interaction and meeting one of the following constraints:
   (a) One test case has a score higher than 5 and another test case has a score lower than 5, and the difference between the values is equal or greater than 2 points.
   (b) A pair of test cases whose readability scores differ in, at least, 3 points in the scale [0, 10].
3. Keep those pairs whose testing goal is a method different from other pairs already selected. If two or more pairs satisfy this point, the one with a larger distance between readability scores is chosen. In the event of a tie, select the pair whose code is less similar to other pairs.

At the end of the process, we have selected 10 pairs of test cases to be distributed among the participants. The pairs belong to executions of InterEvo-TR done by 7 different participants.

2.1.2 Comparison #2: Interactive vs. Automatic

The goal of this comparison is to check whether participants prefer test cases highly-ranked by other participants (using InterEvo-TR) instead of test cases automatically selected by EvoSuite to create the final test suite. Although we do not know the readability score that the original participant would have assigned to the "automatic" test case, he/she could have seen it (or at similar one) during the interaction. The next steps were followed to create the pairs of test cases for comparison:

1. Identify participants who assigned 9 or 10 to some test cases. To add variety, we exclude participants selected in step 1 of Comparison #1.
2. Select a couple of test cases, one with score 10 and another with score 9. Check that the selected test case appears in the final test suite. Exclude it otherwise.

3. Among all the collected test cases, select a subset to have a variety of methods under test.

4. Take the final test suite generated by EvoSuite using the same random seed than the configured for the original participant.

5. Find one test case covering the same method and whose code looks more similar to the "interactive" test case.

We created 10 pairs following the above procedure, based on the original interactions of 5 different participants.

2.1.3 Comparison #3: Automatic vs. Automatic

A final comparison is designed to act as a control case. Participants looking at pairs of test cases may intuitively think that one test case should be better than the other one in the pair, or that they come from different sources. If both test cases belong to automated executions, and the test cases look pretty similar, we will expect similar ratings unless the participants are biased by those guesses. The process to select the test cases was as follows:

1. For each participant, two automatic executions of EvoSuite with different random seeds are considered.

2. Select one of the test cases of the two test suites. At the beginning of the process, we choose the first test case appearing in one of the test suites. As the process of selection of pairs advances and methods are already covered in other pairs, subsequent test cases in the test suite can be included to promote diversity of test codes.

3. Find one test case covering the same method and whose code looks more similar to the test case selected in the previous step.

At the end of this process, we created 10 pairs of test cases related to 7 distinct methods of the class under test.

2.2 Results

2.2.1 Comparison #1: Interactive vs. Interactive

Table 3 shows the proportion of participants with a similar or different perception from the original participant that saw the same tests during one interaction, in total and per each of the 10 pairs of tests (P1 to P10). As it can be seen, 68% of the participants followed a similar evaluation pattern when assessing the pair of tests assigned to them: they selected the same test case as their favorite, exactly as the original participant did during the interaction. In contrast, 19% of them had the contrary perception (they preferred the worst-valued test by the original participant). The rest of the participants, 13%, gave the same score to both tests, resulting in a tie.

We also observe that other participants actually perceived a strong difference in score like the original participant. As shown in Table 4, 48% of the participants gave differences in score of 3 points or more when assessing the pair of tests. This percentage gets more relevant when compared to the differences in score in the comparisons #2 y #3, also shown in this table.

Table 3: Results of comparisons #1 and #2 in total and per pairs (P1 to P10).

Comparison #1	**Total**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
Same perception	68%	4/4	0/3	2/4	2/3	1/2	2/3	3/4	4/4	2/2	1/2
Tie/Different perc.	13%/19%										
Comparison #2	**Total**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
Better interactive	74%	4/4	2/4	1/3	3/4	3/3	2/2	3/3	1/2	2/3	2/3
Tie/Better automatic	16%/10%										

2.2.2 Comparison #2: Interactive vs. Automatic

Table 3 also collects the results related to this comparison. Even a greater proportion of the participants than in the previous comparison, 74% of them, preferred the well-valued interactive test case to the similar test automatically selected by EvoSuite. Only 10% of them showed preference for the automated test case. The evaluation of the rest of the participants resulted in a tie (16%). It is worth noting that the results in favor of the best-rated tests were quite spread across the 10 pairs of test cases generated for the study, revealing that the concrete selection of pairs did not have a decisive influence. The same happened in Comparison #1, except for one pair (P2).

Focusing now on the differences in score (Table 4), we observe fewer strong differences when compared to Comparison #1. For instance, the percentage of participants that gave differences in score of 3 points or more decreased from 48% to 23%. This was an expected result, as the automatic tests had not received a score yet and we could not know whether they were readable for the participants beforehand. Therefore, it is reasonable that the differences in score are not so extreme and mainly concentrate in the group 1-2 points (61%).

Table 4: Differences in the score given to the test cases included in the pairs

Difference in score	**0 points (tie)**	**1-2 points**	**3-4 points**	**>5 points**
Comparison #1	13%	39%	32%	16%
Comparison #2	16%	61%	13%	10%
Comparison #3	3%	87%	10%	0%

2.2.3 Comparison #3: Automatic vs. Automatic

Given that these automatic tests did not receive readability scores, we can only analyze the difference in score between them (Table 4) and compare the results with the two other comparisons as a control case. After the analysis of these results, we can say that this comparison was successful in serving as a control mechanism. For 87% of the participants, the difference in score for both automated test cases was between 0 and 2 points. Those similar ratings tell us that they were not biased by the intuitive belief that the test cases in the pair could come from different sources, or that one of the presented test cases should be better than the other. In fact, we observe a broader range of differences in score in the comparisons #1 and #2. As an example, the difference was never over 5 points in any case, while this situation happened in 16% and %10 of the cases in comparisons #1 and #2, respectively.

References

[1] E. Daka, J. Campos, G. Fraser, J. Dorn, and W. Weimer, "Modeling readability to improve unit tests," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2015. New York, NY, USA: ACM, 2015, pp. 107–118.

[2] G. Bavota, F. Carnevale, A. De Lucia, M. Di Penta, and R. Oliveto, "Putting the developer in-the-loop: An interactive ga for software re-modularization," in *International Symposium on Search Based Software Engineering*, 2012, pp. 75–89.

[3] A. Dantas, I. Yeltsin, A. A. Araújo, and J. Souza, "Interactive software release planning with preferences base," in *International Symposium on Search-Based Software Engineering*, 2015, pp. 341–346.