MERLIN - ModEl-guided Reinforcement LearnINg for Automated Testing of Endless Runner Games

Nigar Azhar Butt*†[1], Muhammad Uzair Khan[1], Muhammad Zohaib Iqbal[2], and Atif Aftab Jilani [1]

[1]Department of Software Engineering, National University of Computer and Emerging Sciences, Islamabad, Pakistan
[2]QUEST, Islamabad, Pakistan

Experiment Design: Comparison of MERLIN with Human Testers

1 Research Question

RQ3: How does the bug detection effectiveness of MERLIN compare to that of human exploratory testers? *RQ3* seeks to assess how well MERLIN can detect faults compared to human exploratory testers by comparing its mutation score with manual testers.

2 Experiment Design

We conducted a small-scale experiment with one case study (C01). We designed our experiment using guidelines presented by Ko et al. [1]. The details are mentioned in the proceeding subsections.

2.1 Variables

- **Independent Variable:** Testing method (human testers vs. MERLIN).
- **Dependent Variables:** Number of faults detected, types of faults detected, mutation score.

2.2 Hypotheses

- **Null Hypothesis (H_0):** There is no difference in the effectiveness of human testers and MERLIN in detecting faults in software variants.
- **Alternative Hypothesis (H_1):** There is a difference in the effectiveness of human testers and MERLIN in detecting faults in software variants.

*Email: nigar.azhar@isb.nu.edu.pk
†Corresponding author

2.3 Participants

Five game testers with varying experiences were provided by the industrial partner[1]. We ensured that the participants were not previously involved in mutant generation process. This was done to prevent potential bias due to prior knowledge of seeded mutations.

2.4 Materials

- Original (correct) version of C01, which is a clone of flappy bird game. It is identified as V00.
- 35 variants of C01, each corresponding to a mutant, and 15 copies of the original version of the game (V00).
- MERLIN for automated testing.
- Google Form to get demographic information about the participants.
- Google Form for collecting tester responses corresponding to each mutant.
- Google Form for debriefing.

2.5 Procedure

2.5.1 Preparation Phase

- Participants fill out the initial demographic form[2].
- Participants are given up to 60 minutes to play the original version of the game for familiarization.
- Participants are provided instructions on completing the Google Form corresponding to variants.

2.5.2 Testing Phase (Human)

- Participants are provided with ten variants, including a mix of seven faulty and three correct games without disclosure of the specific faults within a variant. We also randomized the order in which testers encountered these variants. This encourages unbiased exploration and enhances realism. It also mitigates expectation bias, ensuring testers thoroughly explore the game for unforeseen issues rather than focusing solely on expected faults.
- Participants perform exploratory testing of each variant for 15 minutes which is more than twice the average gameplay session for such games [2]. This allows testers to thoroughly explore the game within a realistic timeframe while having sufficient time to encounter various scenarios.
- Participants submit a google form[3] containing tester and variant IDs, textual fault descriptions if detected, and additional comments on gameplay. Hence, providing structured and organized feedback. This allows for easy tracking and analysis of detected faults and gameplay observations. Additionally, it ensures that relevant information is captured consistently across all testing sessions, facilitating comparison and evaluation of results.

[1] Quest : `https://www.questlab.pk`
[2] `https://bit.ly/MERLIN-ET-registeration-form`
[3] `https://bit.ly/MERLIN-experiment-form`

– The experiment was supervised to ensure adherence to the guidelines and to address any queries or concerns raised by the participants. This supervision helps maintain the integrity and reliability of the experiment by minimizing potential biases, ensuring participants understand the tasks, and providing support when needed. It also helps in resolving any issues promptly, ensuring the smooth conduct of the experiment and reliable results.

2.5.3 Automated Testing Phase

– MERLIN is tasked with testing the same set of mutants for C01. We used the results of C01 from RQ1 in which we found mutation scores achieved by MERLIN across 30 runs. To ensure the statistical significance of our results and minimize random variations, we repeated the experiment 30 times, following the recommendations of Arcuri et al[3]. MERLIN had 10 playthroughs to eliminate each mutant, recognizing the dynamic nature of endless runner games where the agent may encounter obstacles or fail before eliminating any mutants.

2.5.4 Debrief Phase

– Participants fill out the debrief form[4] to provide feedback on the experiment and to gain deeper insights into the tester's experiences and perceptions.

2.6 Data Collection

– Collect data from initial demographic forms filled out by participants[5].

– Collect data from Google Forms submitted by participants during the testing phase[6].

– Collect data from MERLIN on the number and types of faults detected in each variant using the previous mutation testing experiment(RQ1).

– Collect feedback and suggestions from the debrief forms filled out by participants[7].

2.7 Analysis

In this section, we present the results of our experiment involving human testers, where we compared their performance with that of MERLIN in detecting faults based on quantity and type. While human testers were generally successful in identifying faults by the end of each session, certain mutants associated with ADD, AVI, AVD, GFR, and GFT evaded detection. For instance, in variants V24 and V29, which were mutated using the AVI mutation operator, the testers (HT04) failed to identify the faults. This lack of detection was attributed to the minimal variation in speed introduced by the mutations, which was so subtle that it went unnoticed by the tester. Similarly, for variants V33 and V38, employing the ADD and AVD mutation operators respectively, the faults were not detected by tester HT03. In both cases, the delay in action activation resulting from the mutations was exceptionally small, rendering it imperceptible to the tester during gameplay. Additionally, variants V44, V48, V49, V50, mutated using GFT and GFR operators, remained undetected by the testers (HT03, HT04, and HT05). The reason behind this was that their highest score was capped at 5,8, and 10 respectively. Consequently, the testers did not play long enough or achieve the necessary score threshold to encounter the faults introduced by these mutations. These

[4] `https://bit.ly/MERLIN-debrief-form`
[5] `https://bit.ly/MERLIN-ET-registeration-data`
[6] `https://bit.ly/MERLIN-ET-experiment-data`
[7] `https://bit.ly/MERLIN-ET-debrief-data`

faults remained undetected as testers did not meet the conditions to trigger them, although, with greater familiarity with the game, they could potentially be identified.

In the case of V02, a unique situation arose where one of our testers (HT02) reported a fault that didn't exist. However, due to the experiment being supervised, we promptly discovered that a background process was overloading their system, causing a lag in the game. This made it seem to the tester as if the pipes were appearing slower than in the original version, leading them to identify this variant as faulty. Nevertheless, we recognized this issue and instructed the tester to switch to a different system. This was the only instance of a false positive in our experiment. However, it is important to note that such false positives do not impact the mutation score, as it focuses solely on the faults detected in the faulty variants and disregards the correctness of the original version.

Consequently, for C01, human testers achieved a mutation score of 80%, while MERLIN demonstrated superior performance with an average score of 93.05% for C01. The Table 1 presents a summary of an exploratory testing experiment with testers for C01, detailing the outcomes for various game variants. Each row corresponds to a specific variant, identified by a Variant ID (VID), and lists the applied Mutation Operator (MO). Additionally, it records the Tester ID (TID) of the individual who tested the variant and its resulting status. The status indicates whether the variant was 'Killed' due to identified faults, 'Alive' indicating fault was not caught, or 'N/A' for correct variants. Furthermore, MERLIN processed all mutants across all case studies in an average time of 15 minutes, whereas human testers required an average of 5 minutes per mutant. In future experiments, we intend to explore the impact of providing human testers with variable or extended time frames across different case studies.

2.8 Threats to validity

In selecting human testers, we ensured diversity in experience and skill levels to broaden the spectrum of insights and bug-detection capabilities. This approach mitigates the threat of limited perspectives and enhances the likelihood of detecting a wider range of bugs. Furthermore, testers were not informed about the specific bugs injected into the games prior to testing to maintain objectivity. During testing sessions, we randomized the order of game versions presented to each tester to prevent expectation bias and ensure unbiased exploration of the games. Additionally, each tester was allocated sufficient time to thoroughly examine each game version, thereby reducing the risk of overlooking potential issues due to time constraints. To systematically document findings, testers were required to complete a structured form after each testing session, facilitating organized data collection and analysis. Throughout the experiment, careful oversight was maintained to ensure adherence to protocols and address any uncertainties or queries raised by testers. These steps help ensure the reliability of our research findings.

3 Conclusion

To answer RQ3, based on the results of the evaluation, we can say that MERLIN is more effective in bug detection than human testers.

Table 1: Summary of Exploratory Testing Experiment with Testers - RQ3

VID	MO	TID	Status
V01	RUSD	HT01	Killed
V02	V00	HT02	Killed
V03	RUAR	HT03	Killed
V04	V00	HT04	N/A
V05	RUOR	HT05	Killed
V06	RUOR	HT01	Killed
V07	RUOR	HT02	Killed
V08	RUOR	HT03	Killed
V09	DCD	HT04	Killed
V10	V00	HT05	N/A
V11	DCD	HT01	Killed
V12	DCD	HT02	Killed
V13	DAL	HT03	Killed
V14	ARR	HT04	Killed
V15	V00	HT05	N/A
V16	V00	HT01	N/A
V17	ADD	HT02	Killed
V18	V00	HT03	N/A
V19	ADD	HT04	Killed
V20	ADD	HT05	Killed
V21	V00	HT01	N/A
V22	V00	HT02	N/A
V23	AVI	HT03	Killed
V24	AVI	HT04	Alive
V25	AVI	HT05	Killed

VID	MO	TID	Status
V26	RUOR	HT01	Killed
V27	DCD	HT02	Killed
V28	V00	HT03	N/A
V29	AVD	HT04	Alive
V30	AVD	HT05	Killed
V31	AVI	HT01	Killed
V32	AVD	HT02	Killed
V33	ADD	HT03	Alive
V34	V00	HT04	N/A
V35	GFA	HT05	Killed
V36	V00	HT01	N/A
V37	GFT	HT02	Killed
V38	AVD	HT03	Alive
V39	V00	HT04	N/A
V40	GFT	HT05	Killed
V41	GFT	HT01	Killed
V42	V00	HT02	N/A
V43	V00	HT03	N/A
V44	GFT	HT04	Alive
V45	V00	HT05	N/A
V46	GFR	HT01	Killed
V47	RUAR	HT02	Killed
V48	GFR	HT03	Alive
V49	GFR	HT04	Alive
V50	GFR	HT05	Alive

References

[1] A. J. Ko, T. D. LaToza, and M. M. Burnett, "A practical guide to controlled experiments of software engineering tools with human participants," *Empirical Software Engineering*, vol. 20, pp. 110–141, 2015.

[2] A. Knezovic, "Mobile game session length: How to track & increase it," *Udonis Blog*, Dec. 2023, Last accessed on March 21, 2024. [Online]. Available: `https://www.blog.udonis.co/mobile-marketing/mobile-games/session-length/`.

[3] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings of the 33rd international conference on software engineering*, 2011, pp. 1–10.