2024 fall · Mobile Computing

HW4-Advanced Topics

November 17, 2024

Compass: Compiling Natural Language Instructions to Synthesize XR Device Interactions in Metaverse

Dingyuan Xue

1 Industry trends and needs

Extended Reality (XR) technologies, encompassing Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR), have rapidly emerged as transformative tools across various domains, including gaming, education, healthcare, and industrial training. The growing adoption of XR applications has been driven by their ability to provide immersive and interactive experiences, leading to a burgeoning market with significant economic impact. As these technologies continue to evolve, the complexity of their software ecosystems also increases, necessitating advanced approaches for ensuring their reliability and usability.

One of the fundamental tasks in software engineering, particularly in the context of graphically-rich software such as XR applications, Android apps, and web platforms, is the automatic translation of natural language instructions into precise user interactions that can be executed on these systems. This task underpins several critical downstream processes, including automated testing based on natural language testing plans, test generation through the emulation of online tutorials, and bug reproduction from user-reported issues. Successfully automating this translation is essential for improving the efficiency and effectiveness of software testing and maintenance in these complex environments.

2 Current Solutions

2.1 VR/AR testing

Many researchers have made excellent achievements in VR and AR software testing to fill the gaps in this emerging field. [1–6]

In the field of automated testing of VR applications, Gil et al. [1] present Youkai, a framework for unit testing of Android-based VR applications, which can find objects on the screen, and change the camera position and it supports 6 DoF scenarios. Souza, Nunes, and Dias [2] introduce VR-ReST as an approach to automate the generation of test data from requirements specification in the VR domain. It uses a semi-formal language for requirements specification and structural test criteria to generate test requirements and data. Wang [3] introduces VRTest, a testing framework aimed at automating the testing of virtual reality scenes. It utilizes information extraction from VR scenes and user camera manipulation to explore and interact with virtual objects based on predefined testing strategies. Based on VRTest, Wang, Rafi, and Meng [4] introduces VRGuide, a novel testing technique for virtual reality scenes that utilizes the Cut Extension computational geometry method to optimize camera routes and achieve comprehensive object interaction coverage. Recently, Rzig et al. [5] investigated the testing practices of open-source VR applications to summarize the shortcomings of existing testing methods. For AR application testing, Rafi et al. [6] proposes a novel approach called PredART that predicts human ratings of virtual object placements in AR testing using automatic screenshot sampling, crowd sourcing, and a hybrid neural network for image regression. The approach achieves an accuracy of 85.0% and is useful as test oracles in automated AR testing.

2.2 Bug Replay

Automated bug report replay can greatly improve the efficiency of practitioners in fixing bugs. Researchers have tackled this challenge by studying bug reports and guiding automatic replay, achieving excellent results. [7–16]
Extracting high-quality steps to reproduce (S2R) was a core goal in the study of bug reports. Chaparro et al. [7] investigated and found that existing bug reports often lacked the S2R section and proposed an approach to automatically identify bug reports containing S2R. After the investigation, Chaparro et al. [8] introduce Euler, an approach that automatically identifies and assesses the quality of steps to reproduce in bug reports. It provides feedback to reporters, enabling them to improve the bug report. Zhao et al. [9] propose S2RMiner, an approach that automatically extracts the text description of steps to reproduce (S2R) from bug reports for generating reproducing scripts. S2RMiner combines HTML parsing, natural language processing, and machine learning techniques and achieves high accuracy. To get higher quality bug reports, Fazzini et al. [10] present a novel bug-reporting approach named EBug that assists users in writing detailed steps to reproduce (S2Rs) for mobile applications. EBug analyzes natural language information entered by users in real time and suggests potential future steps using predictive models. To automate the entire bug replay process, Fazzini et al. [11] introduce Yakusu, which analyzes the UI configuration files to generate S2R by dependency parse tree. In ReCDroid, Zhao et al. [12] set up a grammar pattern table to classify natural languages in bug reports. Then S2R is generated according to the grammar format of the classification. After that, Liu et al. [13] propose an automated approach called MaCa that identifies and classifies action words in bug reports for mobile apps by machine learning and successfully improves the performance of Yakusu and ReCDroid. More recently, Feng and Chen [14] applied LLM to the extraction of S2R in their work AdbGPT, which utilizes few-shot learning and chain-of-thought reasoning to emulate human knowledge and logical reasoning.
In the guided replay section, Feng and Chen [15] created GIFdroid, a lightweight approach that utilizes image processing techniques to automatically replay the execution trace from visual bug reports. This approach extracts keyframes from the recording, maps them to states in the GUI Transitions Graph, and generates the execution trace of those states to trigger the bug. In their subsequent work Adbgpt [14], they encode the UI information into HTML format to help large language models better understand UI information. Then, the large language model is guided by UI information and S2R to direct the bug replay process. Zhang et al. [16] modeled the matching process between S2R and UI as an MDP (Markov Decision Process) and employed Q-learning to obtain the target UI components. ReCDroid [12] constructs a Dynamic Ordered Event Tree from textual information in the UI to guide the exploration of the GUI. The exploration order of nodes is determined by the matching degree between the target UI components in S2R and the textual information of each component on the current screen.

3 Critical Analysis

The existing VR automation testing methods can effectively apply basic software testing techniques to test VR software, such as unit testing and goal-free global exploration. These testing methods are well-suited for performing initial tests on VR programs that have not yet received bug reports or user feedback, helping to eliminate most bugs. Regarding bug replay, the current methods are effective in generating "steps to reproduce" based on relatively standardized bug report formats, allowing for bug replay on mobile applications like Android programs. This helps expose bugs, making it easier for developers to fix them.

My preliminary investigations reveal that existing methods fall short when replaying bugs to XR environments. The existing VR automation testing methods do not adopt a bug replay strategy similar to that used in Android testing. And the existing bug replay methods, originally designed for traditional 2D interfaces, struggle to handle the unique challenges posed by the spatial and temporal complexity of XR interactions. Specifically, they encounter difficulties in accurately interpreting and executing multi-modal commands that involve spatial navigation, gesture recognition, and time-sensitive interactions. The limitations are exacerbated by the high degree of variability in user actions within immersive environments, where the semantics of interactions are often more complex and less deterministic compared to

conventional software applications.

4 Approach

The rapid advancement of XR technologies presents unique challenges in accurately reproducing and analyzing user-reported issues, as well as in documenting gameplay strategies. Traditional methods of bug reporting and strategy documentation often fall short in capturing the complexity and temporal nuances of XR interactions. To address this gap, I propose a format to satisfy formatted representation of events in VR applications and use LLM to automatically extract S2Rs in DSL format by using chain-of-thought.

4.1 Formatted Representation

Manipulation statements provide direct control over the properties of XR devices. These statements are the primary means through the user's physical interactions with the XR environment:

```
<manipulation_statement> ::= <device>"."<property> <value> ";"
```

For example, a manipulation statement might set the position or orientation of a controller or adjust the state of a button, directly reflecting the user's actions in the XR space.

The format explicitly models XR devices and their properties, enabling detailed and precise control over the elements that users interact with. The grammar defines devices such as head-mounted displays (HMD), left controllers (LCT), and right controllers (RCT), each of which has specific properties that can be manipulated:

```
// Devices
<device> ::= "HMD" | "LCT" | "RCT"

// Properties
<property> ::= <pose_property> | <orientation_property> | <button_property> | <axis_property>
<pose_property> ::= <position_coordinate> | <orientation_coordinate>
<position_coordinate> ::= "X" | "Y" | "Z"
<orientation_coordinate> ::= "Qx" | "Qy" | "Qz"
<button_property> ::= "Grip" | "A" | "B" | "Menu" | "System" | "Trigger"
<axis_property> ::= "Thumbstick_x" | "Thumbstick_y"
```

These properties can describe user's actions such as pressing buttons, moving controllers, or adjusting the orientation of the HMD. And <value> can be a float number which be assigned to the property. For example, <LCT> <A> <1> means press the button "A" on the left controller. And <HMD> <X> <120> means turn the head-mounted displays around x axis by 120 degree.

4.2 Generate S2Rs by Chain-of-thought

The Chain of Thought (CoT) technique for generating S2Rs requires manually creating a sample input and sample output, and then logically explaining to the large language model how to derive the corresponding output from the input. This technique can significantly improve the reasoning ability of large language models in customized scenarios. I collected the text of the game walkthourgh as data. The following is one of the examples of CoT. And also we need to tell LLM the available devices and properties.

Example input: Press the red button next to the elevator door.

Chain-of-Thought: The action is "press", and the target is "red button". Therefore the whole step can be devided into two steps "move the right hand to the button" and "press the button". For the first step, the purpose is to change the player's right hand's position to the position of the button. The action to change the player's right hand's position is to move with the right controller, which is represented by "RCT", so choose "RCT" as the Device. The "move" in the x, y and z coordinates are represented by "X", "Y", and "Z", respectively, so choose X, Y or Z as the Properties. The coordinates of the destination are the x, y, z coordinates of the button, "x_button", "y_button", "z_button" respectively. Therefore, the corresponding Action primitives for this sentence are "RCT, X, x_button", "RCT, Y, y_button", "RCT, Z, z_button". For the third step, the purpose is to yaw the player's right hand counterclockwise a bit. The action to yaw

the player's right hand is to yaw the right controller, which is represented by "RCT", so choose "RCT" as the Device. The "yaw" is represented by "Qx", so choose "Qx" as the Operation-property. "-" means "counterclockwise" and "20" means "a bit", so the Value is "-20". Therefore, the corresponding Action primitives for this sentence are "RCT, Qx, -20".

Example output:
Overall, the extracted S2R entities are:
RCT, X, x_button
RCT, Y, y_button
RCT, Z, z_button
RCT, W, -20

5 Conclusion

Due to the current lack of bug replay attempts in VR automation testing methods and the difficulty of adapting existing bug replay techniques to VR scenarios, I propose a language format to standardize and accurately express operational events in VR environments. Additionally, I attempt to use the Chain of Thought (CoT) technique to enable large language models to automatically extract S2Rs from natural language text, such as bug reports or VR game walkthroughs, and convert them into the format described above. This would facilitate the extraction of S2Rs in the bug replay process.

References

[1] A. M. Gil, T. S. Figueira, E. Ribeiro, A. R. Costa, and P. Quiroga, "Automated test of VR applications," in *HCI International 2020 - Late Breaking Posters - 22nd International Conference, HCII 2020, Copenhagen, Denmark, July 19-24, 2020, Proceedings, Part II*, ser. Communications in Computer and Information Science, C. Stephanidis, M. Antona, and S. Ntoa, Eds., vol. 1294. Springer, 2020, pp. 145–149. [Online]. Available: https://doi.org/10.1007/978-3-030-60703-6_18

[2] A. C. C. Souza, F. L. S. Nunes, and M. E. Delamaro, "An automated functional testing approach for virtual reality applications," *Softw. Test. Verification Reliab.*, vol. 28, no. 8, 2018. [Online]. Available: https://doi.org/10.1002/stvr.1690

[3] X. Wang, "Vrtest: An extensible framework for automatic testing of virtual reality scenes," in *44th IEEE/ACM International Conference on Software Engineering: Companion Proceedings, ICSE Companion 2022, Pittsburgh, PA, USA, May 22-24, 2022*. ACM/IEEE, 2022, pp. 232–236. [Online]. Available: https://doi.org/10.1145/3510454.3516870

[4] X. Wang, T. Rafi, and N. Meng, "Vrguide: Efficient testing of virtual reality scenes via dynamic cut coverage," in *38th IEEE/ACM International Conference on Automated Software Engineering, ASE 2023, Luxembourg, September 11-15, 2023*. IEEE, 2023, pp. 951–962. [Online]. Available: https://doi.org/10.1109/ASE56229.2023.00197

[5] D. E. Rzig, N. Iqbal, I. Attisano, X. Qin, and F. Hassan, "Virtual reality (VR) automated testing in the wild: A case study on unity-based VR applications," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, R. Just and G. Fraser, Eds. ACM, 2023, pp. 1269–1281. [Online]. Available: https://doi.org/10.1145/3597926.3598134

[6] T. Rafi, X. Zhang, and X. Wang, "Predart: Towards automatic oracle prediction of object placements in augmented reality testing," in *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022*. ACM, 2022, pp. 77:1–77:13. [Online]. Available: https://doi.org/10.1145/3551349.3561160

[7] O. Chaparro, J. Lu, F. Zampetti, L. Moreno, M. D. Penta, A. Marcus, G. Bavota, and V. Ng, "Detecting missing information in bug descriptions," in *Proceedings of the 2017 11th Joint Meeting*

on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017, E. Bodden, W. Schäfer, A. van Deursen, and A. Zisman, Eds. ACM, 2017, pp. 396–407. [Online]. Available: https://doi.org/10.1145/3106237.3106285

[8] O. Chaparro, C. Bernal-Cárdenas, J. Lu, K. Moran, A. Marcus, M. D. Penta, D. Poshyvanyk, and V. Ng, "Assessing the quality of the steps to reproduce in bug reports," in *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*, M. Dumas, D. Pfahl, S. Apel, and A. Russo, Eds. ACM, 2019, pp. 86–96. [Online]. Available: https://doi.org/10.1145/3338906.3338947

[9] Y. Zhao, K. Miller, T. Yu, W. Zheng, and M. Pu, "Automatically extracting bug reproducing steps from android bug reports," in *Reuse in the Big Data Era - 18th International Conference on Software and Systems Reuse, ICSR 2019, Cincinnati, OH, USA, June 26-28, 2019, Proceedings*, ser. Lecture Notes in Computer Science, X. Peng, A. Ampatzoglou, and T. Bhowmik, Eds., vol. 11602. Springer, 2019, pp. 100–111. [Online]. Available: https://doi.org/10.1007/978-3-030-22888-0_8

[10] M. Fazzini, K. Moran, C. Bernal-Cárdenas, T. Wendland, A. Orso, and D. Poshyvanyk, "Enhancing mobile app bug reporting via real-time understanding of reproduction steps," *IEEE Trans. Software Eng.*, vol. 49, no. 3, pp. 1246–1272, 2023. [Online]. Available: https://doi.org/10.1109/TSE.2022.3174028

[11] M. Fazzini, M. Prammer, M. d'Amorim, and A. Orso, "Automatically translating bug reports into test cases for mobile apps," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2018, Amsterdam, The Netherlands, July 16-21, 2018*, F. Tip and E. Bodden, Eds. ACM, 2018, pp. 141–152. [Online]. Available: https://doi.org/10.1145/3213846.3213869

[12] Y. Zhao, T. Yu, T. Su, Y. Liu, W. Zheng, J. Zhang, and W. G. J. Halfond, "Recdroid: automatically reproducing android application crashes from bug reports," in *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, J. M. Atlee, T. Bultan, and J. Whittle, Eds. IEEE / ACM, 2019, pp. 128–139. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00030

[13] H. Liu, M. Shen, J. Jin, and Y. Jiang, "Automated classification of actions in bug reports of mobile apps," in *ISSTA '20: 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, USA, July 18-22, 2020*, S. Khurshid and C. S. Pasareanu, Eds. ACM, 2020, pp. 128–140. [Online]. Available: https://doi.org/10.1145/3395363.3397355

[14] S. Feng and C. Chen, "Prompting is all you need: Automated android bug replay with large language models," in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024*. ACM, 2024, pp. 67:1–67:13. [Online]. Available: https://doi.org/10.1145/3597503.3608137

[15] ——, "Gifdroid: Automated replay of visual bug reports for android apps," in *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*. ACM, 2022, pp. 1045–1057. [Online]. Available: https://doi.org/10.1145/3510003.3510048

[16] Z. Zhang, R. Winn, Y. Zhao, T. Yu, and W. G. J. Halfond, "Automatically reproducing android bug reports using natural language processing and reinforcement learning," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, R. Just and G. Fraser, Eds. ACM, 2023, pp. 411–422. [Online]. Available: https://doi.org/10.1145/3597926.3598066