Scriptless Testing for Extended Reality Systems

Fernando Pastor Ricós[0000−0002−5790−193X]

Universitat Politècnica de València, Spain fpastor@pros.upv.es

Abstract. Extended Reality (XR) systems are complex applications that have emerged in a wide variety of domains, such as computer games and medical practice. Testing XR software is mainly done manually by human testers, which implies a high cost in terms of time and money. Current automated testing approaches for XR systems consist of rudimentary capture and replay of scripts. However, this approach only works for simple test scenarios. Moreover, it is well-known that the scripts break easily each time the XR system is changed. There are research projects aimed at using autonomous agents that will follow scripted instructions to test XR functionalities. Nonetheless, using only scripted testing techniques, it is difficult and expensive to tackle the challenges of testing XR systems. This thesis is focus on the use of automated scriptless testing for XR systems. This way we help to reduce part of the manual testing effort and complement the scripted techniques.

Keywords: Scriptless testing, State model inference, Extended Reality

1 Introduction

XR systems have been on the rise in recent years to allow users to interact with simulated environments. XR software systems have emerged in different domains, ranging from medicine, for marketing purposes, to computer games [19]. The latter sector accounts for about 50% of the Virtual Reality (VR) software market. The complexity of the navigation and interaction in 3D spaces, the increasing importance of user experience, the existence of randomness and non-determinism in VR games, together with the agile development requirements of the market [9], makes XR testing a challenging and critical task [10].

Nowadays, XR testing practice is mainly done manually. This implies high costs in terms of time and money. The sector lacks automation processes, frameworks, and tools [13]. Even though automated software testing has been significantly researched to reduce the problems of manual testing (time, cost, etc.), for instance, as shown for testing through the Graphical User Interface (GUI) [16], the existing testing tools do not consider the complexity of testing XR systems.

Automated scripted testing is based on scripts that are manually crafted, generated from models, or recorded with a tool to replay them later [15]. These scripts contain the interactions that the automated tool must execute to validate some functional requirements. However, the current automated testing approaches for XR systems are often based on the rudimentary capture and replay of scripts, which only works for simple test scenarios. Furthermore, the scripts break easily when the XR system is changed.

In order to tackle the problems of applying automated scripted testing to XR systems, the Intelligent Verification/Validation for Extended Reality Based Systems (IV4XR, 2019-2022) project is developing a framework to allow the observation of XR entities and the use of autonomous Functional Test Agents (FTAs) to achieve testing goals [14]. These FTAs follow scripted tactics and imply the need to manually create and maintain scripts.

Manually defining and maintaining scripts to cover all possible functional System Under Test (SUT) dependencies would mean investing a lot of money and time. Therefore, using only scripted techniques, it is difficult and expensive to tackle the challenges of testing XR systems. To reduce the effort related to test scripts, additional and complementary testing approaches are required.

Scriptless testing is an outstanding approach intended to automatically interact and validate the software without the use of scripts. These tools automatically explore and generate test sequences by selecting and executing the available actions in the discovered states. The use of scriptless testing tools has proven to be complementary to scripted tools for covering different parts of the SUT and for detecting unexpected software failures [11]. Moreover, scriptless testing can be beneficial to check offline oracles [7], to visualize model transitions and changes between different versions of the same software [2], to automatically generate test cases from the model [1], or to automatically apply regression testing to detect changes [8]. Despite all these benefits, there is a lack of research to apply the scriptless approach for XR environments.

The goal of this research is to evaluate the benefits of using scriptless testing for two VR software systems. To do that, we select the scriptless open source tool TESTAR [22]. TESTAR is an actively maintained tool that tests desktop, web and mobile applications through the GUI. In this research, TESTAR must be evolved to be an intelligent exploratory agent that tests XR environments. Thus, we advocate that scriptless testing can help developers and testers at XR companies by reducing manual effort and complementing scripted techniques.

The rest of this paper is structured as follows. Section 2 presents the related work. Section 3 introduces the the main characteristics of TESTAR. Section 4 exposes research challenges for XR systems. Section 5 presents the completion plan for this thesis research, including the validation plan. Section 6 concludes.

2 Related Work

Testing XR systems is difficult. The development of XR systems differs from traditional systems due to less clear requirements, high level of interactivity and increased realism, entities behavior dependent on the context, among others; making testing a challenging work [17]. Testing XR systems is mainly performed manually, because it is needed to deal with interfaces in 3D spaces, human behavior is difficult to reproduce, and entities are highly dependent on the context.

Little research has been performed on automated testing of XR systems. In [12], the difference in the evolution of the requirements between traditional desktop, web or mobile software, and XR video game open source projects is presented. This study shows that, as a XR video game project evolves, the development effort is reduced from code functionality and focuses more on the

multimedia features. It also shows that testing objectives of XR systems evolve differently than those of traditional software, and that malfunctions of XR systems are more related to the interface than the programming errors.

In [4], a script based approach for automated functional testing of XR systems is proposed. It offers a semi-formal language to describe the requirements specification and then automates the generation of test cases using scene graph concepts to represent the virtual environment. Although this proposal can be a useful scripted approach, the effort to maintain the scripts is problematic.

The IV4XR project uses autonomous agents that follow declarative tactics to test XR systems [14]. Artificial Intelligent (AI) techniques are applied to have robustness tactic-based tests and reduce the maintenance effort [18]. MBT is used to generate tactic-based test cases automatically [5]. However, manual effort is still required if the XR testing needs to cover a high amount of system paths, either to create tactic tests or to craft a model of an existing level.

The scriptless approach and the use of AI agents has not been thoroughly investigated in terms of test effectiveness, efficiency and usefulness. Consequently, the motivation of this work is to research the benefits of using the scriptless tool TESTAR as an exploratory agent within the IV4XR project.

3 TESTAR Tool

TESTAR is a scriptless tool that, while testing, infers a state model that is stored in an OrientDB graph database. TESTAR connects and interacts with the SUT using an API to obtain the current state and execute actions. Windows Automation and Java access bridge can be used for desktop applications, Selenium WebDriver for web pages, and Appium for mobile applications. To be able to connect and interact with XR systems, we use the open source IV4XR Java software plugin that allows TESTAR to interact with two IV4XR use cases: LabRecruits [1] and Space Engineers (SE) [2]. LabRecruits is a VR open source demo game and SE is an industrial VR game with millions of players.

The logical flow of TESTAR together with the integration of the IV4XR plugin is shown in Figure 1. The World Object Model (WOM) interface allows TESTAR to observe the information of the virtual entities through a set of properties such as the entity type, the position and the size. Thus, the XR state consists on the set of entities that exist in the observation range. Because the type of entities and their properties are different depending on the virtual SUT, TESTAR needs to define the XR state differently for LabRecruits and SE. For example, SE entities contain properties that indicate the actual integrity and 3D position of blocks, which is something that does not exist for LabRecruits game.

XR actions are the available interactions that TESTAR can execute in each XR state. We derive two types of XR actions: basic commands and compound tactics. A basic command action is the most basic event that TESTAR can execute, e.g. move or rotate one step, equip a tool and start or stop using a tool. However, due to the essence of XR systems, most of the time it is necessary to

[1] https://github.com/iv4xr-project/labrecruits
[2] https://github.com/iv4xr-project/iv4xr-se-plugin

execute a compound tactical action that contains several basic commands. For example, to interact with a SE block entity TESTAR needs to rotate to aim the block, move to reach the block, equip a tool and start using this tool.

Based on the available actions in a state s, TESTAR selects and executes an action a and obtains a new state s'. The transition (s, a, s‘) is then stored into the state model and TESTAR continues with state s‘. It generates test sequences until the STOP condition is met. Thus, the state model is a knowledge graph that contains information about the elements in the states, as well as the executed and non-executed actions.

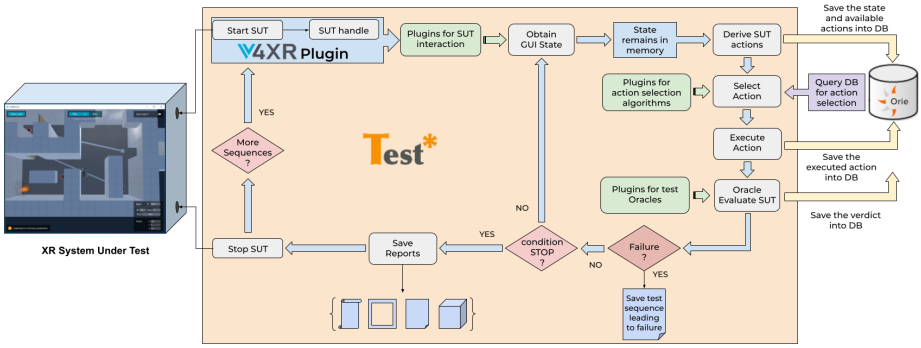

Fig. 1. TESTAR operational flow

During the integration of the IV4XR plugin, TESTAR was able to discover two failures. In LabRecruits, we detected a hang exception when interacting with a non-interactive entity. In SE, we discovered a plugin exception when TESTAR took off his helmet and died while exploring SE system. For this reason, we expect to use possible failures found by TESTAR as a measure of test effectiveness.

4 XR Research Challenges

To evaluate scriptless testing for XR systems, we work to solve four challenges that are described in this section.

When TESTAR interacts with desktop, web or mobile SUTs, most of the GUI actions are based on mouse movements in the 2D screen followed by click, drag or type events. However, for XR systems there is greater complexity when executing actions. An XR action may require move and orientate TESTAR in 3D environments and deal with obstructive objects to reach the entities. Therefore we have to research: *1. How to develop an action derivation mechanism to perceive the environment and realize smart interaction movements.*

TESTAR maintains two types of states in its model: concrete and abstract states. Concrete states are created using all the properties of all the entities. Abstract states creation can be customized by selecting which properties are used for state abstraction. A suitable level of abstraction allows TESTAR to select non-dynamic properties to create a traceable model avoiding a state explosion. However, for XR systems, some properties, such as the position of the agent, are important but too concrete to determine the XR state. Thus, we need to analyze: *2. How to define a suitable approach to abstract the area of the* TESTAR *agent.*

For GUI applications, when TESTAR obtains the GUI state, the tool detects the available widgets to interact with. However, for XR systems, the observation about the reachable entities is restricted to the observation range and obstructive objects. A basic movement can modify the observed entities and discover that new entities can be reached by navigating to certain positions. Since TESTAR does not contain such a complex navigation feature, we need to investigate: *3. How to implement a feature that allows* TESTAR *to navigate the* XR *space and remember how to reach the existing entities.*

For XR systems we need a new definition of test oracles. Crash and hang failures are generic oracles that, in principle, can be used for most XR systems. However, other oracles intended to check the functionality of the XR interactions or the correct visualization of virtual entities, can be specific and different between XR systems. Thus, we need to analyze the test requirements for each SUT and study: *4. How to adapt* TESTAR *oracles for* XR *systems.*

5 Completion Plan

This thesis aims to improve four functionalities of TESTAR to tackle the challenges presented above. It is the first time presented in a doctoral consortium.

5.1 Smart Interaction Movements

TESTAR needs to be able to observe the environment, select an entity to interact with, and perceive which virtual objects obstruct the movement to the entity. For LabRecruits, TESTAR can use tactical actions, together with the automatic NavMesh [21] map created by Unity to follow the NavMesh positions to reach the desired entity. However, in other XR systems such as SE VR game, this NavMesh map does not exist by default. We are working on creating a 2D as well as 3D pathfinding (with jetpack) algorithm for SE. This constructs a sparse grid on-the-fly as the agent moves around and avoids obstacles.

5.2 XR State Abstraction

Current abstraction mechanisms are not feasible for some XR systems. In SE, TESTAR movements change the observation range and, therefore, the SE blocks of the XR state. This implies the constant creation of new abstract states in the model. A possible solution for SE is to use the location area of TESTAR together with the SE grid entities. One grid is a group of blocks that generally represents a structure such as a spatial base. The exploration movements along these grids will modify the observed blocks but not the grid structure.

5.3 Navigable State Model Layer

In XR systems, especially in virtual games, the user has the possibility to move around the virtual world or a navigable area, in order to reach the interactive entities. To determine which are the reachable entities, TESTAR needs to explore the navigable areas and store the position of the discovered entities.

A navigable state has been implemented in the TESTAR state model. Figure 2 shows an example of this feature with LabRecruits. TESTAR prioritizes the exploratory movements to discover the reachable entities of the navigable state. After fully exploring the navigable state, TESTAR selects an interactive action not executed previously. Then, TESTAR starts a new exploration to be able to map which interactive actions of the SUT connects the existing navigable states.

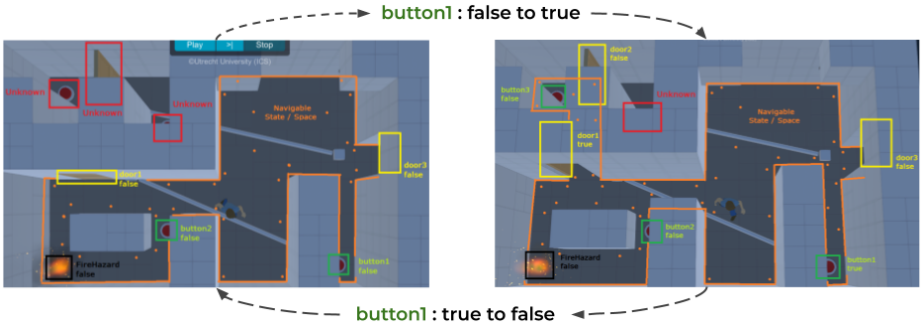

Fig. 2. TESTAR navigable state transition

5.4 XR Test Oracles

Although there are generic oracles that can be applied to most systems to detect crashes, hangs or exception messages, the definition of what a failure is and the oracles to detect it, depends on each system. In the development of XR systems, a lot of effort is dedicated to the visualization of virtual objects. Thus, the oracles aiming to verify the correct visualization of these objects are of great interest in the development processes of virtual systems.

IV4XR project has a trained model that helps to verify the correct visualization of SE blocks according to their integrity values. TESTAR is able to detect the type and the integrity of the blocks using the IV4XR plugin and take screenshots of the SE system using the Windows Accessibility API. Thus, we aim to research the integration of the model in TESTAR to be used as a visual oracle while TESTAR explores the virtual environment.

5.5 Validation Plan

LabRecruits and SE partners are interested in conducting this research and publishing the validation results. We will evaluate TESTAR effectiveness, efficiency, and the usefulness of the test results. Let us look into each of these separately.

– `RQ1`: How does TESTAR improve test effectiveness of XR systems?

On non-XR systems, TESTAR test effectiveness can be measured by the usual code coverage, GUI coverage and failures found. Although these are still desired (two interesting system failures were already found during the TESTAR integration), the usual metrics are not enough. This is because, for XR systems, test effectiveness also depends on game-specific properties like the entities, the interactions, the reality, the high level game objectives, the levels and the players.

Therefore, to answer `RQ1`, first we need to discuss with the product owners to determine what they consider to be effective for their particular system. Then, we need to execute a series of experiments to obtain and evaluate these metrics.

For LabRecruits, it is considered effective to test the transition coverage of the buttons and doors at different levels. Also, because this system is developed with Unity, it is possible to obtain code coverage metrics [20]. For SE, it is considered effective to test the integrity and the visualization of the blocks when using a tool to interact with it, or the correct use of materials to build blocks.

In this first PhD year, we expect to execute a series of experiments with LabRecruits to measure how effective is TESTAR in terms of transition and code coverage. Since LabRecruits can have different types of levels, and because TESTAR always contains a certain level of randomness, we need to make sure we obtain significant evidence. For this we will realize an empirical evaluation and use for example the Bayesian statistical analysis [6]. Furthermore, for SE, we need to research and determine which metrics to use to measure effectiveness.

For the second PhD year, we plan to conduct an industrial experiment with SE. We expect to use TESTAR to explore SE and interact with different blocks to validate properties such as integrity, and research the use of an image recognition model as an oracle to validate blocks visualization. Then, realize an empirical evaluation by using the effectiveness metrics defined during this PhD year.

– `RQ2:` How useful are the test results of TESTAR?

TESTAR creates logs, HTML reports with screenshots, and the state model, as test result artifacts. Logs and HTML reports are useful to visualize how a test sequence was executed, especially if a failure has been found. The state model is an artifact that can be useful for more purposes, as was stated in Section 1.

One of the research topics of the IV4XR project is the use of a MBT tool to generate button-interaction test cases. To use the MBT tool with an existing LabRecruits level, it is necessary to manually craft the finite interaction model before being able to generate test cases [5]. TESTAR navigable state model contains the necessary information that the MBT tool needs to automatically generate these test cases. For this reason, we expect to analyze the useful complementarity of TESTAR and the MBT tool before the end of this PhD year.

– `RQ3:` How efficient is the TESTAR exploration process on XR systems?

Besides investigating effectiveness and usefulness of the TESTAR results, we still need to research the third important property of testing: efficiency [3].

Because TESTAR does not test a specific set of actions but explores the existing amount of available actions, it needs more time than scripted approaches to reach significant effectiveness. For the third PhD year, we expect to research the integration of a distributed framework that allows the execution of multiple TESTAR instances that use the state model as a central knowledge database.

6 Conclusion

In this doctoral consortium paper we have presented the challenges for testing XR systems, the research performed so far, the goal of our research, and the completion activities and validation plan that will be carried on during the PhD.

Acknowledgements

This work has been funded by iv4XR (H2020, 856716) project.

References

1. Aho, P., Alégroth, E., Oliveira, R., Vos, T.: Evolution of automated regression testing of software systems through the graphical user interface. In: 1st ACCS. pp. 16–21 (2016)

2. Aho, P., Suarez, M., Kanstrén, T., Memon, A.M.: Murphy tools: Utilizing extracted gui models for industrial software testing. In: ICSTW. pp. 343–348 (2014)
3. Böhme, M., Paul, S.: A probabilistic analysis of the efficiency of automated software testing. IEEE Transactions on Software Engineering **42**(4), 345–360 (2015)
4. Correa Souza, A.C., Nunes, F.L., Delamaro, M.E.: An automated functional testing approach for virtual reality applications. STVR **28**(8), e1690 (2018)
5. Ferdous, R., Kifetew, F., Prandi, D., Prasetya, I., Shirzadehhajimahmood, S., Susi, A.: Search-based automated play testing of computer games: A model-based approach. In: 13th SSBSE. pp. 56–71. Springer (2021)
6. Furia, C., Feldt, R., Torkar, R.: Bayesian data analysis in empirical software engineering research. IEEE TSE **47**(9), 1786–1810 (2021)
7. de Gier, F., Kager, D., de Gouw, S., Vos, T.E.J.: Offline oracles for accessibility evaluation with the testar tool. In: 13th RCIS. pp. 1–12. IEEE (2019)
8. Grilo, A.M., Paiva, A.C., Faria, J.P.: Reverse engineering of gui models for testing. In: 5th Iberian Conf. on Information Systems and Tech. pp. 1–6. IEEE (2010)
9. Kropp, M., Meier, A., Anslow, C., Biddle, R.: Satisfaction, practices, and influences in agile software development. In: 22nd EASE. pp. 112–121 (2018)
10. Lin, D., Bezemer, C.P., Hassan, A.E.: Studying the urgent updates of popular games on the steam platform. Empirical Software Eng. **22**(4), 2095–2126 (2017)
11. Machiry, A., Tahiliani, R., Naik, M.: Dynodroid: An input generation system for android apps. In: 9th FSE. pp. 224–234 (2013)
12. Pascarella, L., Palomba, F., Di Penta, M., Bacchelli, A.: How is video game development different from software development in open source? In: 2018 IEEE/ACM 15th MSR. pp. 392–402 (2018)
13. Politowski, C., Petrillo, F., Guéhéneuc, Y.G.: A survey of video game testing. arXiv preprint arXiv:2103.06431 (2021)
14. Prasetya, I., Dastani, M., Prada, R., Vos, T.E.J., Dignum, F., Kifetew, F.: Aplib: Tactical agents for testing computer games. In: 8th EMAS. pp. 21–41 (2020)
15. Rafi, D.M., Moses, K.R.K., Petersen, K., Mäntylä, M.V.: Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In: 7th Int. W. on Automation of Software Test (AST). pp. 36–42. IEEE (2012)
16. Rodríguez-Valdés, O., Vos, T., Aho, P., Marín, B.: 30 years of automated gui testing: A bibliometric analysis. In: 14th QUATIC. pp. 473–488. Springer (2021)
17. Santos, R., Magalhães, C., Capretz, L., Correia-Neto, J., da Silva, F., Saher, A.: Computer games are serious business and so is their quality: particularities of software testing in game development from the perspective of practitioners. In: 12th ESEM. pp. 1–10 (2018)
18. Shirzadehhajimahmood, S., Prasetya, I., Dignum, F., Dastani, M., Keller, G.: Using an agent-based approach for robust automated testing of computer games. In: 12th A-TEST. pp. 1–8 (2021)
19. Stanney, K., Nye, H., Haddad, S., Hale, K., Padron, C., Cohn, J.: extended reality (xr) environments. Handbook of human factors and ergonomics pp. 782–815 (2021)
20. Unity: Code coverage (2019), https://docs.unity3d.com/Packages/com.unity.testtools.codecoverage@1.0, last accessed: 18 Feb 2022
21. Unity: Navigation system in unity (2021), https://docs.unity3d.com/Manual/nav-NavigationSystem.html, last accessed: 18 Feb 2022
22. Vos, T., Aho, P., Pastor Ricos, F., Rodriguez-Valdes, O., Mulders, A.: testar–scriptless testing through graphical user interface. STVR **31**(3), e1771 (2021)