Agent-based Testing of Extended Reality Systems

Conference Paper · October 2020

DOI: 10.1109/ICST46399.2020.00051

10 authors, including:

Rui Prada
University of Lisbon
214 PUBLICATIONS **2,821** CITATIONS
SEE PROFILE

Wishnu Prasetya
Utrecht University
123 PUBLICATIONS **517** CITATIONS
SEE PROFILE

Fitsum Meshesha Kifetew
Fondazione Bruno Kessler
56 PUBLICATIONS **1,153** CITATIONS
SEE PROFILE

Jean Yves Donnart
Thales Group
16 PUBLICATIONS **258** CITATIONS
SEE PROFILE

Agent-based Testing of Extended Reality Systems

Rui Prada
INESC-ID and Instituto Superior Técnico, Univ. de Lisboa
Portugal, rui.prada@tecnico.ulisboa.pt

I. S. W. B. Prasetya
Utrecht Univ., Netherlands
Orcid: 0000-0002-3421-4635

Fitsum Kifetew
Fondazione Bruno Kessler
Italy, Orcid: 0000-0003-1860-8666

Frank Dignum
Umea Univ., Sweden

Tanja E. J. Vos
Univ. Politecnica de Valencia, Spain

Jason Lander
Gameware, UK

Jean-yves Donnart
Thales AVS, France

Alexandre Kazmierowski
Thales SIX GTS, France

Joseph Davidson
GoodAI, Czech Rep.

Pedro M. Fernandes
INESC-ID and Instituto Superior Técnico, Univ. de Lisboa, Portugal

***Abstract*—Testing for quality assurance (QA) is a crucial step in the development of Extended Reality (XR) systems that typically follow iterative design and development cycles. Bringing automation to these testing procedures will increase the productivity of XR developers. However, given the complexity of the XR environments and the User Experience (UX) demands, achieving this is highly challenging. We propose to address this issue through the creation of autonomous cognitive test agents that will have the ability to cope with the complexity of the interaction space by intelligently explore the most prominent interactions given a test goal and support the assessment of affective properties of the UX by playing the role of users.**

***Index Terms*—agent-based testing, AI-based testing, testing computer game, testing virtual reality, user experience testing**

Fig. 1. *A game called Space Engineers as an example of an XR system, featuring advanced 3D worlds and an elaborate system to construct sophisticated custom objects, from simple solar panels to a complete space station.*

I. Introduction

Extended Reality (XR) systems are advanced interactive systems such as systems with advanced 3D UI, Virtual Reality (VR), and Augmented Reality (AR) systems. They have emerged in various domains, ranging from entertainment, cultural heritage, to combat training and mission critical applications. Testing is particularly critical to assure functional correctness and high quality user experience (UX), driving the typical agile iterative development process. Unfortunately, the current XR development toolset possesses little technology beyond rudimentary record and replay tools that only work for simple test scenarios and furthermore break easily whenever the XR system is changed. XR testing practice involves therefore thousands of hours of manual play by human testers. As one can imagine, this is very costly and difficult to manage. Worse, such practice impedes the industry's growth and its agility to timely market sophisticated virtual and augmented environments with higher quality user experience.

Challenges. Introducing automation to XR testing would greatly benefit the industry, but to get there, the following key challenges need to be addressed first:

1) **Fine-grained interaction space**. More traditional interactive applications such as a webshop or a spreadsheet give the user a strictly controlled and limited number of interaction choices. XR systems more accurately reflect the real world, so they allow fine grained, almost continuous, interactions. XR worlds are also inhabited by independent and dynamic entities simulating the corresponding real world entities. They interact with the user as well as with each other, and often lead to emerging behavior. These result in an interaction space far larger than in traditional interactive digital products, and intractable by existing automated testing approaches such as model based testing [1], combinatoric [2], or search-based testing [3]. On the other hand, humans seem to be relatively unaffected by the immensity of this interaction space. After just some training they are immediately able to do, for example, basic navigation through a virtual world, without having to memorize, or even be aware of, every micro detail of its interaction space. This implies that existing automated testing approaches miss the right abstraction and learning skills to reduce a huge interaction space to a tractable level.
2) **Assessing user experience (UX)**. For any XR system, delivering high quality user experience (UX) is very important. If it is not smooth enough, is too boring, or too overwhelming, the users become unhappy, or worse annoyed. Bad user reviews can be detrimental, causing customers to abandon the product. Business loss is not the only concern. The history of disasters in the interaction between humans and machines highlights human error as the main cause [4]. Presence of UX issues can stress a human operator, making him/her more prone

Funded by EU ICT-2018-3 H2020 Programme, grant nr. 856716 and by national funds through FCT, Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020

to making mistakes. In mission-critical applications this is a serious concern. Since manually assessing the UX quality is very labour intensive, automation would help a lot, for example, to find scenarios with potential UX issues on which testers can focus their more refined manual effort. Unfortunately, existing tools are too simplistic, e.g. they simply check if every screen is not too crowded. These tools lack deeper models of human emotion and cognitive capabilities to be able to judge the different emotional states that an interaction event might evoke on users. Furthermore, users have consistent differences that make them react differently to the XR, e.g. due to personality, individual expertise or different cultural background. For example, an elderly user might respond differently than a young user. Current tools are not able to deal with such diversity nor are they able to judge the progression of the UX that is built up over time as users engage in long term interactions.

Vision statement. The Intelligent Verification/Validation of Extended Reality Systems (**IV4XR**) project is a recently started EU project (2019-2022) with partners from academia and XR industry to deliver an answer to the above challenges, demonstrable with real world pilots such as shown in Fig. 1. This paper/poster presents the project's vision and approach[1], whose main thesis is the following:

> A missing key ingredient in the current automated testing approaches to scalably handle XR systems' fine grained interaction space is *cognition*. Its inclusion would enable a testing algorithm to gain insight of what the entities in a virtual or augmented world represent to human users, hence allowing it to distinguish the parts of the interaction space that matter for the test goal at hand from the parts that can be ignored. Also, functional cognitive inference can be extended with emotion inference, to open a way to automated UX assessment.

The IV4XR project seeks to combine advances in *cognitive AI*, *affective computing*, and software testing, in particular *search-based testing* to develop a new generation of testing algorithms with cognitive skills, either where such skills can be simply derived from a library, or can be modularly programmed by developers. The project will deliver a proof-of-concept XR testing technology where developers can attach their XR systems and get access to the whole range of automated testing capabilities (implementing the said algorithms).

The next section will present IV4XR approach.

II. Agent-based Cognitive Approach to XR Testing

Software agents have been used for many years to model human cognitive based behaviour [5]. Over the last decades, various *computational models* have been developed. A quite popular model because of its versatility is the *Belief-Desire-Intent* (BDI) agency [6]. In this model, the agent may have one or more *goals* ("desires" in the model) and creates plans

[1]For a working initial implementation see: https://github.com/iv4xr-project

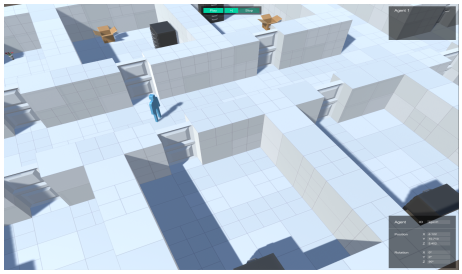

Fig. 2. *A 3D game under-development using IV4XR testing.*

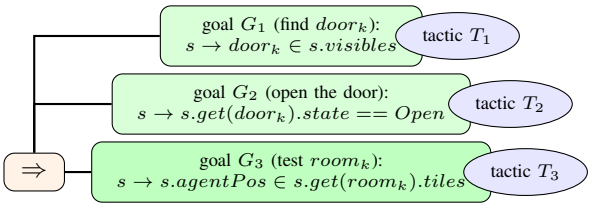

Fig. 3. *A testing task specified as BDI-style goals* $G_1; G_2 \Rightarrow G_3$.

(the intended sequence of actions) that it will try to execute in the environment towards reaching its goals. Cognition is programmed as inference rules to infer new knowledge/belief and to decide best actions or strategies to take. Goals can be *testing* related, e.g. to test that certain entities in a virtual world would interact correctly. Agents can act as automated programs, let's call them *test agents*, that use their cognition to solve given testing goals.

In comparison, search-based testing, which has been successful in other domains, relies on e.g. genetic operators such as cross-over and mutation [3] to create candidate plans to solve test goals. However, such operators incorporate very little domain specific reasoning. For example Fig. 2 shows a fragment of a 3D virtual world of a game that is under development. Developers need to verify that all rooms in this world are reachable, which amounts to checking that all doors can be opened, which in turn require specific switches to be turned. The game itself contains many other interactive objects. Searching the right plan to solve this testing task without some form of cognition will be extremely difficult. Direct application of unsupervised machine learning without the help of cognition would also face the same problem. Therefore in IV4XR we will be exploring the **integration** of artificial cognition, e.g. a la BDI, with existing software testing techniques such as search-based testing. Using BDI-like agents facilitates the incorporation of knowledge in the search process. Some programming will be inevitably needed to impart the knowledge, but this can be done declaratively. In return, this allows the agents to concentrate the search on those aspects that are important for a current test focus. E.g. in the scenario above we might assume that the switches are clear for human users. Thus the agents get basic actions to get to the switch and turn it, rather than finding out which objects in the room might be switches.

As an illustration, Fig. 3 shows how a testing task for a test

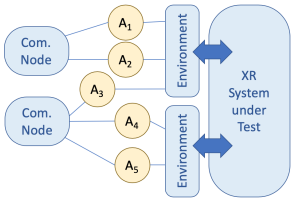

Fig. 4. *Deploying multiple IV4XR test agents. Agents can form a group by registering to a communication node, allowing them to exchange messages.*

agent can be declaratively expressed as a BDI-style goal[2]. It specifies how to test if $room_k$ is reachable. This is specified as the goal G_3. The goal structure in Fig. 3 introduces the inference $G_1; G_2 \Rightarrow G_3$: to solve G_3 we would first need to solve G_1 (to find the door to the room, represented by $door_k$) and then G_2 (to open the door). Note that ultimately what matters is G_3. We do not have to introduce the inference $G_1; G_2 \Rightarrow G_3$. However, doing so adds some bit of insight to the test agent, which results in the reduction of the search space. Goals are essentially state predicates. To solve them, *planning tactics*/heuristics (blue ellipses) are used. E.g. T_1 (to solve G_1) can be a search algorithm, whereas for T_2 some reasoning would be needed when the door turns out to be closed, so that the agent can infer, for example, that it first needs to generate a new goal to find the corresponding switch.

There are more benefit in using BDI-based test agents:

- By relying on cognition test agents become **more adaptive** towards emerging behavior (often present in XR systems) and lower level design changes that often happen during development. E.g. in Fig. 3, the goal G_1 does not restrict which route the agent should take to find the said door. Provided the planning tactic T_1 is smart enough, the task can still be completed even if some random entity blocks the test agent's shortest route to the door.
- All BDI implementation platforms (see e.g. [8], [9]) also provide constructs for designing multi-agent systems. Fig. 4 shows how multiple IV4XR test agents can be deployed to an XR system under test [7]. Two or more agents can form a group, allowing them to communicate within the group and coordinate their actions. This would provide a natural instrument to test cooperative or adversarial multi-user features that are often present in many XR systems.

Combining our approach with recent works on Machine Learning (ML) based automation e.g. [10] is another interesting direction to explore, e.g. to train certain testing-related tasks rather than programming them. Additionally, taking a cognitive agent approach offers a new opportunity to improve the efficiency of the training, e.g. by using cognitive models to control ML heuristics to focus their exploration on cognitive viable strategies rather than just any possible strategy.

A. Testing User Experience

Although software testing is usually thought of as testing a system for errors, the errors can be of different types. Sometimes the software is performing what it is intended to do, but the end result is not as expected. E.g. in a game where the user has to overcome all kinds of obstacles it can happen that the obstacles are all too easy and the user gets through the obstacles very quickly without any enjoyment. It leads to boredom and possibly abandoning the game. In the other extreme obstacles might theoretically be solvable, but too hard for most users and thus lead to frustration and abandoning the game or erratic behavior to alleviate the frustration. It is also the case that different users experience the software in different ways. For example, a set of obstacles may be too easy for an expert player but not so much for a novice player. The developers may want to test what parts of the interaction experience are suitable for experts and what parts for novices. It is also interesting to test what is the path that a user takes from novice to become and expert. In particular, how long it takes and what are the barriers that delay such transition.

In order to perform these user experience tests, the best way is to have test persons testing all aspects of the game. But this is very time consuming if a game (like e.g. Space Engineers[3], Fig. 1) is very elaborate and can take days to play. Therefore, we are also exploring how to test user experiences with software agents. We will use more elaborate cognitive agents that have not just goal directed behavior but also have emotions and needs, and different capabilities, knowledge and limitations. There are variations and extensions of BDI platforms that would allow emotional process to be modelled to complement logical thinking. A good example of such an extension is provided by FAtiMA[4] [11], which implements the so-called OCC model [12] which has been used in various studies on computational frameworks of emotion. The model originates from the Psychology, structuring 22 emotion types (e.g. joy,satisfaction, hope, disappointment, distress and fear) based on focus of attention.

When emotion-enhanced agents start exploring the environment (representing an XR system, or any system with UI), the effects of their interactions will have impact on their mental state. This can be used as a first approximation of possible user experience. Hence, the tests will check, on one hand, properties of the interactions (e.g. the number of wrong actions they take, or how long they take to learn) and, on the other hand, will check variables of their internal state (e.g. what emotions are raised, or what knowledge they build).

B. Maximizing coverage

Beyond solving a particular test goal set out by the human tester, the test agents in IV4XR will strive to explore alternative ways of achieving the same test goal, considering different paths. In doing so, the test agents ensure that the test scenario is invoked and covered in all the possible ways it could be, hence the test becomes less susceptible to different execution contexts of the XR system under test. To achieve such a diversified coverage of the test goals, we will employ

[2]This is already supported by the current IV4XR prototype framework [7].

[3]https://www.spaceengineersgame.com/

[4]https://fatima-toolkit.eu/

advanced search strategies that encourage the test agents to explore paths that were never explored before. This can be achieved by embedding self-motivation into the test agents such that, beyond satisfying a given test goal, they will be self-driven towards new paths for achieving the same goal and earn rewards. Such a scheme will be applied in the context of search-based heuristics that favor the discovery of *novel* solutions. Our idea is to explore classes of objective-less search heuristics where, differently from traditional search heuristics where the focus is on optimizing a given objective function, candidate solutions are chosen and promoted based on their diversity with respect to previously seen solutions [13], [14].

Computer generated solutions can easily be too 'synthetic' (humans would never produce them). Such solutions should not be ignored, but we should make sure that we *also* explore diversity over 'natural' solutions. Again, cognition will be deployed to characterize the interactions that functionally matter for the test goal at hand (hence what humans would try), thus helping the diversity search in generating diversity that matters (rather than just random diversity).

III. Related Work

Despite its potential, the application of artificial cognition for software testing has been mostly ignored. Much of the studies were on the exploitation of machine learning, see the survey [15]. Supervised learning is often studied for automating test cases evaluation and prioritization, or to train artificial test oracles, whereas unsupervised learning is used to improve coverage [10], [16]. Early work on using cognitive agents, see e.g. [17], focused mainly on the architecture (providing implementation templates of BDI concepts for testing) rather than on how to actually exploit cognition. There were some works on using cognition to select test cases, but this of course does not answer the question on how to generate our XR test cases in the first place. An attempt was made in the context of GUI testing [16] where Prolog-style inference rules were used to specify which user action to trigger on which GUI state, but the approach is not agent-based (hence does not benefit from the latter either), nor did it not explore its scalability towards solving complex testing goals.

The realm of UX evaluation still heavily relies on user based testing with very little automation. A correlation between heart rate, electrodermal activity and player experience could be used to automate UX evaluation [18], avoiding the time consuming Likert scale based questionnaires. However, such an approach would still rely on user testing. Visual and behavioural clues have also been used to train models to predict certain UX characteristics [19]. These can be used to create player tailored game levels that would invoke a certain degree of, for example, frustration. Being based on artificial neural networks, such models are black-box models, having no intrinsic modelling of human cognition and emotion.

There have been some approaches that use agents with characteristics of human cognition, e.g. short term memory, to better understand how users would explore a map. Modelling personas based on different pruning methods for Monte Carlo tree search algorithms has also been proposed [20]. However, to our knowledge, there is no stand-alone model of human cognition and emotion that could be used to assess UX without extensive support of user testing.

IV. Conclusion and Future Work

We believe artificial cognition to be a key enabler for automated XR testing. This is the direction that the IV4XR project will take in the coming three years. An initial implementation of a BDI agency for testing is available for interested readers, keeping in mind that the development of the agents' cognitive skills is still on going. Addressing VR and AR is future work[5].

References

[1] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," *Software Testing, Verification and Reliability*, vol. 22, no. 5, 2012.

[2] D. R. Kuhn, R. N. Kacker, and Y. Lei, *Introduction to combinatorial testing*. CRC Press, 2013.

[3] P. McMinn, "Search-based software test data generation: a survey," *Softw. testing, Verification and reliability*, vol. 14, no. 2, 2004.

[4] S. Dekker, *The field guide to understanding human error*. Aldershot, 2006.

[5] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson Education, 2016.

[6] M. Wooldridge, "Intelligent agents," *Multiagent systems*, vol. 35, no. 4, p. 51, 1999.

[7] I. Prasetya, "Aplib: Tactical programming of intelligent agents," *arXiv preprint arXiv:1911.04710*, 2019.

[8] K. Kravari and N. Bassiliades, "A survey of agent platforms," *Journal of Artificial Societies and Social Simulation*, vol. 18, no. 1, p. 11, 2015.

[9] M. Dastani, "2APL: a practical agent programming language," *Autonomous agents and multi-agent systems*, vol. 16, no. 3, 2008.

[10] Y. Zheng, X. Xie, T. Su, L. Ma, J. Hao, Z. Meng, Y. Liu, R. Shen, Y. Chen, and C. Fan, "Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning," in *34th Int. Conf. on Automated Software Engineering (ASE)*, 2019.

[11] S. Mascarenhas, M. Guimarães, R. Prada, J. Dias, P. A. Santos, K. Star, B. Hirsh, E. Spice, and R. Kommeren, "Virtual agent toolkit for serious games developers," in *Proc. of the IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018.

[12] A. Ortony, G. Clore, and A. Collins, "The cognitive structure of emotions. cam (bridge university press," *Cambridge, England*, 1988.

[13] J. Gomes, P. Mariano, and A. L. Christensen, "Devising effective novelty search algorithms: A comprehensive empirical study," in *Annual Conf. on Genetic and Evolutionary Computation*. ACM, 2015.

[14] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *ICML*, 2017.

[15] V. Durelli, R. Durelli, S. Borges, A. Endo, M. Eler, D. Dias, and M. Guimarães, "Machine learning applied to software testing: A systematic mapping study," *IEEE Transactions on Reliability*, 2019.

[16] S. Bauersfeld and T. E. J. Vos, "User interface level testing with TESTAR; what about more sophisticated action specification and selection?" in *Series on Adv. Techniques & Tools for Softw. Evolution*, 2014.

[17] V. Lazarou, S. Gardikiotis, and N. Malevris, "Agent systems in software engineering," in *Tools in Artificial Intelligence*. IntechOpen, 2008.

[18] A. Drachen, L. E. Nacke, G. Yannakakis, and A. L. Pedersen, "Correlation between heart rate, electrodermal activity and player experience in first-person shooter games," in *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*. ACM, 2010, pp. 49–54.

[19] N. Shaker, S. Asteriadis, G. N. Yannakakis, and K. Karpouzis, "Fusing visual and behavioral cues for modeling user experience in games," *IEEE Transactions on cybernetics*, vol. 43, no. 6, pp. 1519–1531, 2013.

[20] C. Holmgard, M. C. Green, A. Liapis, and J. Togelius, "Automated playtesting with procedural personas with evolved heuristics," *IEEE Transactions on Games*, 2018.

[5]In principle, VR shares much similarity with a 3D world, and AR can be made testable if it can be simulated through a virtual world.