Know Your Bugs: A Survey of Issues in Automated Game Testing Literature

Riccardo Coppola, Tommaso Fulcini, Francesco Strada
Department of Control and Computer Engineering
Politecnico di Torino
Turin, Italy
first.last@polito.it

***Abstract*—As the complexity of video games continues to evolve, so does the importance of effective game testing methodologies. To this end, automated game testing has emerged as a pivotal approach to ensure the quality and functionality of modern games.**

The objective of the present paper is to identify, through a literature review and the application of Open and Axial coding, the most commonly analysed and mentioned issues in automated game testing literature.

The results of the study provide a taxonomy of 26 different issues that are assessed in the software engineering literature by automated game testing practice, grouped in five higher-level categories.

The elicited taxonomy can serve as an instrument for game testers, researchers and tool developers to evaluate testing approaches and techniques, enable comparability of research results, and design instruments to investigate functional aspects of games in development.

***Index Terms*—Game development, Game testing, Quality Assessment**

I. Introduction

Over the past few decades, the gaming industry has emerged as one of the most lucrative sectors within the software universe, with revenues soaring to an impressive US $249.60 billion in 2023[1]. The development of modern video games often involves massive budgets, extensive teams, and several years of dedicated work before their launch. However, it's also become increasingly common for these games to be released with a host of day-one bugs. These issues can be so severe that they not only jeopardize a game's financial success but can also tarnish the reputation of even the most esteemed game developers [1].

Software testing encompasses a broad range of practices aimed at ensuring the reliability and performance of software products across various platforms. The field of Software Engineering has introduced numerous strategies for planning, conducting, and evaluating software tests, along with a focus on automating these processes to maintain high standards of software quality across all domains.

Despite the critical role of software testing, the gaming industry faces unique challenges in adopting automated testing techniques, as noted in several comprehensive reviews [2]–[4]. To date, there has been a lack of efforts to systematize the analysis of specific game-related issues that can be detected by the application of automated game testing. Such categorization is crucial for evaluating and improving testing methods and tools, providing a basis for comparison in academic and industry research alike.

The present paper aims to categorize, in the form of a taxonomy, the types of issues analyzed or mentioned in the automated game testing literature. The results of the present study are preliminary outcomes of a larger-scale Systematic Literature Review covering various facets of the software testing process when applied to games. These results are complementary with the coverage measures presented in Coppola et al. [5].

The remainder of the manuscript is organized as follows: Section II reports related literature in the field of automated testing for games; Section III briefly describes the literature review process employed for the collection of primary sources and the definition of the taxonomy; Section IV reports the taxonomy and provides pointers to existing literature for each of the coverage metric defined; Section V concludes the paper and provides future research directions.

II. Background and Related Work

The academic literature summarizing the findings and research directions related to automated game testing is quite narrow to date, being a relatively more recent trend than automated software testing in other domains. Within this section, we report and discuss the main secondary studies that are available in the literature.

Albaghajati and Ahmed have presented a comprehensive evaluation framework in their work on testing activities within the gaming domain [4]. In their paper, the authors mainly focus on the way automated testing is performed, considering techniques, objectives, tools and target games, rather than delving into the types of defects that can be verified. The authors categorize automated testing techniques into five main types: search-based approaches, concentrating on exploring a game's state space; goal-directed approaches, employing automated agents with predefined policies to navigate game scenarios; human-like approaches, focusing on mimicking human behaviour through optimized agents; scenario-based approaches, executing test sequences based on predetermined human-made or requested actions; and model-based approaches,

[1]https://bit.ly/472DrQZ

which abstract game workflows into formal representations for event and control verification. The authors also identify key objectives of game testing, including functional correctness (ensuring adherence to specified requirements), multiplayer stability, performance, visual correctness, game design correctness, game balance and fairness, progression and learnability, and physical correctness. Notably, they underscore the absence of universally applicable validation procedures in existing literature. They emphasize the limited applicability of many testing approaches to specific domains or game genres, urging further research efforts to enhance the generalizability of testing activities across diverse contexts.

In a first 2021 survey, Politowski et al. delved into white and grey literature to provide insights into game testing practices [2]. Their findings revealed a prevalent dependence on manual play-testing and testers' intrinsic knowledge among game developers, leading to a dearth of automation. The authors attributed this scarcity to the limited generalizability of existing techniques and the necessity for testing strategies tailored to the nuances of game projects, in contrast to traditional software in other domains. They concluded by advocating for substantial academic and industrial efforts to broaden the adoption of testing in video game development processes.

In 2022, based on the paper collected in their previous work, Politowski et al. expanded the starting pool of sources with snowballing and conducted a comprehensive analysis of game test automation tools [3], enriching the findings from the literature with survey responses from 12 developers. The surveyed developers, well-versed in both traditional software development and game development (with 58% having more than four years of experience in the latter), expressed scepticism toward automated agents for game testing. The results of the literature review emphasized that existing testing tools and frameworks primarily focus on AI-based approaches for game modelling and exploration rather than advancing functional and user-oriented testing of video games. The authors underlined a preferential reliance towards the adoption of manual play-testing in the game development community, citing a lack of automation due to the low generalizability of available techniques (as already pointed out by Albaghajati et al.) and the need for testing strategies tailored to the unique aspects of game projects.

Categorizing and describing bugs is a crucial activity to be able to recognize and report them, especially with automatic approaches. In their study, Li et al. developed and made publicly available a dataset of bugs and a framework, namely *GBGallery* based on that for automated game testing [6]. The authors, in close contact with the developers, relied on five real games to obtain the data and perform an empirical experiment. The authors found a set of 76 bugs, which they classified into five different categories: Crash, Stuck, Game logic, Game Balance and Display.

Although the said categorization provides valuable support, being based on only five games it cannot be considered exhaustive, as the authors themselves state in their validity threats section. Hence, the present study aims to expand the knowledge around game bugs, developing a non-ambiguous taxonomy that can be adopted in all the game-related domains.

III. Methodology

The work aims to identify, through a literature review, the most common issues detected by automated gaming testing tools and to define a taxonomy for the game-testing community.

The taxonomy shall guide testers to identify which game issues are addressed in research and practice and provide researchers with an instrument to compare different game-testing approaches.

The methodology used in this work can be separated into two steps: (i) literature review, and (ii) formulation of the taxonomy through Open Coding.

A. Literature Review

We performed a targeted literature review by applying a search string on a set of scientific literature repositories. To that extent, we applied a subset of Kitchenham's guidelines to conduct Systematic Literature Reviews [7].

- *Selected Digital Libraries*: As digital libraries for our search, we selected IEEE Xplore, ACM Digital Library, Science Direct, Springer Link, Scopus and Google Scholar.
- *Search String*: We formulated our search string to include the terms *Game* (or gaming), test*, automat**. The search string was adapted to fit the syntaxes of the six selected digital libraries.
- *Inclusion Criteria*: To gather only sources relevant to our research goals, we defined the following criteria: (i) the source is directly related to the topic of game testing; (ii) the source defines or mentions explicitly issues to be found through the application of game testing; (iii) the source is an item of white literature with available full text, with publication date between 2012 and 2023; (iv) the source is written in a language that is directly comprehensible by the authors (English or Italian).

B. Coding

To construct a comprehensive taxonomy guiding game testing, we employed Ralph's guidelines for taxonomy development, adhering to the Grounded Theory approach [8]. We adhered to the Straussian definition of Grounded Theory, defining beforehand our research question, aligning with Van Niekerk's approach [9].

The codes for the taxonomy were defined by applying the *Open Coding* technique, i.e., analyzing text data to capture the information of the theory under construction. Through Open Coding, we identified and defined lower-level categories or codes within the taxonomy. A set of common definitions served as the framework, with each individual issue or bug sourced from literature assigned to a specific code. The categories are considered mutually exclusive (i.e., one issue in the literature sources can be assigned to only one code). The taxonomy took shape incrementally, with new codes added

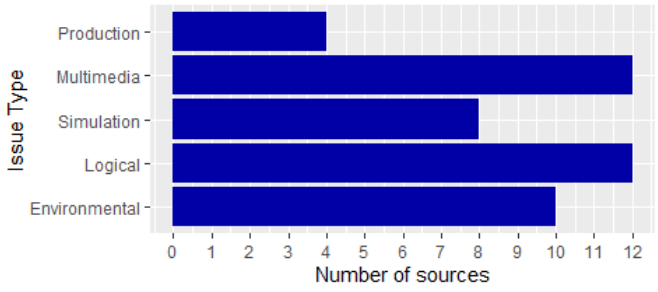

Fig. 1. Number of distinct literature sources per category

whenever an issue definition or usage in the reviewed literature did not align with the existing pool of codes.

Following Open Coding, we transitioned to Axial Coding to establish categories of codes. Per Straussian Grounded Theory, Axial Coding is the process of understanding how codes and related concepts are linked to one another, to identify a structure in the taxonomy and define levels in it. Axial Coding was applied by performing two passes (by all the authors) over all the codes of the taxonomy and defining themes (i.e., higher-level categories) of issues.

IV. Results

After the application of the search string on the selected digital libraries, the application of inclusion criteria and the removal of duplicates, 29 sources were found mentioning the types of issues that can be found by performing automated game testing.

For the formulation of the bug taxonomy, we did not rely on any previous definitions that are typical from a software testing background. The motivation behind this choice is that we did not want to influence the taxonomy with bug definitions commonly referred to in software. These, in fact, typically originate from flaws in the program code or the underlying layers. In the gaming context, however, user experience is a priority and the environment is complex, resulting in different types of issues that need to be characterized.

By analyzing the manuscripts, and the application of the coding procedure, 26 codes (i.e., issue definitions) were defined. All the metrics definitions are reported in Table I.

In fig. 1 we report the number of distinct literature sources mentioning each of the coded macro-categories.

The application of axial coding led to the identification of 6 bug categories. We describe below the categories of issues found:

- Environmental: environmental issues are related to the movement of the agents in the game scene and the reachability of the different areas of the game. Environmental issues can be different according to the type of in-game scenes that are available in the game (e.g., 2D vs 3D games) and the different explorations that are available to the game agents.
- Logical: logical issues are related to wrong implementation and unexpected outcomes of the game mechanics and features.
- Simulation: simulation issues are related to any kind of simulation applied to the game objects in the game scene by the adopted engine.
- Multimedia: multimedia issues are related to the rendering of the audiovisual aspects of the game. They are unrelated to gameplay mechanics and features, i.e., they typically do not render the game unplayable like the other categories of issues. Multimedia issues can be related to underlying resource issues but are distinguished from generic bugs in the sense that the fault occurrence affects the multimedia aspect of the game.
- Production: production issues are malfunctions in the game production code or engine that are not related to the game aspects of the software. It can lead to freezes or termination of the game under test.

V. Discussion

The higher-level analysis in Fig. 1 highlights that literature in automated testing for games appears principally focused on the detection of issues related to the game logic and to the audio-video appearance of the games, rather than on environmental and simulation issues. A limited amount of sources focused on non-game related issues (i.e., Production issues). We justify this aspect with two reasonings: (i) literature in game testing is highly specialized in the execution of playtesting activities, leaving the search for resource and regression issues to established testing techniques; (ii) automated game testing tools are in most cases used to execute End-2-End test cases (e.g. IV4XR [34]), thereby allowing only the detection of issues that are reflected by the multimedia appearance of the game under test.

The analysis of issues in the game's internal logic or how the game objects are rendered and simulated by the game engine requires careful instrumentation and integration of the tool with the game engine. Such practice would imply game testers being domain experts and having a profound knowledge of the production code of the games under test. The instrumentation may also require significant effort and be a time-consuming and costly practice, hardly fitting in the game development lifecycle.

The categories of issues found in this study can be compared with existing related categorizations available in the literature. One prominent example is the GBGallery benchmark by Li et al. [6]. The authors of such paper identify five main categories of issues: crashes (i.e., abnormal exits from the game), *stuck* (i.e., the graphical user interface is stuck and preventing the user from continuing); *game logic* (anomalies in how the game logic works); *game balance* (i.e., non-functional bugs impacting the experience perceived by players); *display* (i.e., issues pertaining the GUI or audio playing). These categories of issues can be directly mapped to (sub)categories of the taxonomy discussed in the present paper and therefore validate

TABLE I
DEFINITIONS OF THE CODED ISSUES

Category	Bug	Definition	Refs
Environmental	Stuck	Issues leading players to abnormal states, where they can no longer continue the natural flow of interactions preventing them from proceeding any further due to wrong level design.	[10], [11]
	Wrong interaction with objects	Wrong interaction with objects occurs when a player interacts with items or elements within the game world in a manner that does not behave as intended or expected.	[12], [13], [14]
	Reachability issues	Issues related to positions in the game scene that should be reachable by the player under certain circumstances, but that is not due to environmental design errors.	[13], [14], [15], [16], [17], [18]
	Unintended path to goal	Issues related to positions in the game scene that should not be reachable by the player, but that can be reached by exploiting environment design errors.	[16]
	Wrong positioning	Wrong positioning refer to instances where objects, characters, or environmental features are placed incorrectly within the game world.	[19]
Logical	Incorrect implementation	A scenario where a game's feature, mechanic, or element does not work as intended due to errors in its design or programming	[20], [21], [22]
	Exploitable features	Elements or mechanics that players can manipulate or use in unintended ways to gain advantages	[23]
	Invalid or conflicting states	These issues occur when a game enters a condition that was not anticipated by the developers. This can happen due to a variety of reasons, such as conflicting game logic or unforeseen player actions	[20], [10], [19], [24]
	Deadlocks	Deadlocks refer to a situation in which the game makes the game unplayable due to unforeseen combinations of game states.	[20]
	Softlocks	Softlocks refer to a situation in which the game continues to run but the player is unable to proceed any further because of combinations of attributes or parameters of the game.	[25], [14]
	Balancing issues	Balancing issues refer to the lack of equilibrium among game elements, leading to certain strategies, characters, items, or abilities	[10], [26], [27], [28]
Simulation	Incorrect object movement	Issues related to inconsistencies in the movements of the players or the objects in the game scene.	[29], [30]
	Wrong Collisions	Issues related to inconsistencies in the simulated outcomes of collisions between the players and objects in the environment.	[30], [19]
	Object Mutual Interactivity	Issues related to inconsistencies in the simulated outcomes of collisions between multiple objects in the environment.	[13], [14]
	Missing or wrong collision box / mesh	Issues related to errors in the size, shape or position of collision boxes of the objects rendered on screen.	[16], [23], [22]
Multimedia	FPS drops	FPS drops refer to a decrease in the frame rate, leading to less smooth gameplay and visual stuttering.	[12]
	Sound issues	Sound issues encompass a range of problems related to the game's audio output, including missing sound effects, music, or dialogue; distorted or glitchy audio; or sound effects playing at inappropriate times.	[29], [11], [31]
	Animation issues	Animation issues refer to problems with the movement and behaviour of characters or objects in the game. This can include jerky, unrealistic, or glitched animations, characters moving in ways that defy the game's physics, or animations failing to play under the correct circumstances.	[11]
	Rendering issues	Rendering issues involve problems with the game's graphical display, such as textures not displaying correctly, objects appearing transparent or black, lighting glitches, or shadows behaving unrealistically.	[29], [32], [33], [16], [19], [34], [24], [35]
	Voice issues	Voice issues pertain to problems with the voice acting or dialogue in the game, including synchronization problems where the voice does not match the character's lip movements, lines of dialogue cutting off or overlapping, or the quality of the voice recordings being poor.	[29], [11]
	Graphical output issues	Graphical output issues cover a broad spectrum of problems related to how the game visually presents itself, beyond just rendering issues. This includes incorrect resolution, aspect ratio problems, flickering, screen tearing (where a frame shows information from multiple frames at once), and colour inaccuracies.	[11], [35], [36]
	Hud issues	HUD (Heads-Up Display) issues refer to problems with the game's on-screen interface that provides players with critical information, such as health bars, maps, and ammunition counters.	[29]
Production	Crash	Bugs leading the game to an incorrect state causing the exit from the game and/or eventual shutdown of the underlying environment.	[10], [37], [11]
	Freeze	Freezes refer to situations where a game becomes unresponsive or "freezes" during gameplay, causing the game to stop functioning or responding to user input.	[11]
	Regression Bugs	Regression bugs are defects that occur when a feature that was previously working correctly becomes broken after the introduction of new changes or updates to the game's codebase.	[38]
	Resource Issues	Resource availability issues pertain to situations where the game does not efficiently manage its resources (e.g., memory, CPU usage, GPU usage), leading to performance degradation, crashes, or other undesirable behavior.	[38]

the present study as an extension of the current state of the art.

VI. CONCLUSION AND FUTURE WORK

In future research endeavours, several research directions are present for further exploration and development in the context of automated testing tools and bug database creation

within the game development domain.

Firstly, there is a compelling opportunity to engage tool developers, as well as those interested in contributing to the establishment of bug databases, to foster a shared knowledge repository of issues within the gaming landscape, as done by Li et al. [6]. Collaboration with tool developers can facilitate the enhancement of existing tools and the creation of new ones tailored to efficiently detect various categories of bugs. This collaborative effort aims to streamline bug detection and resolution processes, thereby improving overall game quality.

Secondly, further investigation is warranted into leveraging existing automated testing tools, such as IV4XR [34], to detect different categories of bugs and issues effectively. By implementing tailored detection mechanisms within these tools, game developers can benefit from more comprehensive bug identification and resolution capabilities, ultimately leading to more robust and polished game experiences.

Additionally, future research endeavours could focus on extending the survey to encompass the mapping of coded issues to coverage measures, tools, approaches, and testing methodologies. This extension aims to provide deeper insights into the testing landscape by correlating identified issues with specific testing strategies and methodologies. By gaining a better understanding of the relationship between testing practices and identified bugs, developers can refine their testing approaches to mitigate common issues more effectively.

Continued efforts in these areas have potential for advancing the state-of-the-art in automated testing within game development, ultimately contributing to the creation of higher quality and more resilient gaming experiences.

ACKNOWLEDGEMENT

This study was carried out within the "EndGame - Improving End-to-End Testing of Web and Mobile Apps through Gamification" project (2022PCCMLF) – funded by European Union – Next Generation EU within the PRIN 2022 program (D.D.104 - 02/02/2022 Ministero dell'Università e della Ricerca). This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

REFERENCES

[1] P. Siuda, D. Reguła, J. Majewski, and A. Kwapiszewska, "Broken promises marketing. relations, communication strategies, and ethics of video game journalists and developers: The case of cyberpunk 2077," *Games and Culture*, p. 15554120231173479, 2023.

[2] C. Politowski, F. Petrillo, and Y.-G. Guéhéneuc, "A survey of video game testing," in *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*. IEEE, 2021, pp. 90–99.

[3] C. Politowski, Y.-G. Guéhéneuc, and F. Petrillo, "Towards automated video game testing: still a long way to go," in *Proceedings of the 6th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation*, 2022, pp. 37–43.

[4] A. M. Albaghajati and M. A. K. Ahmed, "Video game automated testing approaches: An assessment framework," *IEEE transactions on games*, 2020.

[5] R. Coppola, T. Fulcini, and F. Strada, "How to measure game testing: a survey of coverage metrics," in *Proceedings of the 8th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation*, 2024.

[6] Z. Li, Y. Wu, L. Ma, X. Xie, Y. Chen, and C. Fan, "Gbgallery : A benchmark and framework for game testing," *Empirical Software Engineering*, vol. 27, no. 6, p. 140, Jul 2022. [Online]. Available: https://doi.org/10.1007/s10664-022-10158-x

[7] B. A. Kitchenham, "Systematic review in software engineering: where we are and where we should be going," in *Proceedings of the 2nd international workshop on Evidential assessment of software technologies*, 2012, pp. 1–2.

[8] P. Ralph, "Toward methodological guidelines for process theories and taxonomies in software engineering," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 712–735, 2018.

[9] J. C. Van Niekerk and J. Roode, "Glaserian and straussian grounded theory: similar or completely different?" in *Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, 2009, pp. 96–103.

[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 *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 772–784.

[11] J. Pfau, J. D. Smeddinck, and R. Malaka, "Automated game testing with icarus: Intelligent completion of adventure riddles via unsupervised solving," in *Extended abstracts publication of the annual symposium on computer-human interaction in play*, 2017, pp. 153–164.

[12] X. Wang, "Vrtest: an extensible framework for automatic testing of virtual reality scenes," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, 2022, pp. 232–236.

[13] R. Tufano, S. Scalabrino, L. Pascarella, E. Aghajani, R. Oliveto, and G. Bavota, "Using reinforcement learning for load testing of video games," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 2303–2314.

[14] R. Ferdous, F. Kifetew, D. Prandi, I. Prasetya, S. Shirzadehhajimahmood, and A. Susi, "Search-based automated play testing of computer games: A model-based approach," in *International Symposium on Search Based Software Engineering*. Springer, 2021, pp. 56–71.

[15] C. Lu, R. Georgescu, and J. Verwey, "Go-explore complex 3d game environments for automated reachability testing," *IEEE Transactions on Games*, 2022.

[16] A. Sestini, L. Gisslén, J. Bergdahl, K. Tollmar, and A. D. Bagdanov, "Automated gameplay testing and validation with curiosity-conditioned proximal trajectories," *IEEE Transactions on Games*, 2022.

[17] S. Iftikhar, M. Z. Iqbal, M. U. Khan, and W. Mahmood, "An automated model based testing approach for platform games," in *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 2015, pp. 426–435.

[18] R. Mawhorter and A. Smith, "Automated testing in super metroid with abstraction-guided exploration," in *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 2023, pp. 1–9.

[19] S. Varvaressos, K. Lavoie, S. Gaboury, and S. Hallé, "Automated bug finding in video games: A case study for runtime monitoring," *Computers in Entertainment (CIE)*, vol. 15, no. 1, pp. 1–28, 2017.

[20] A. Albaghajati and M. Ahmed, "A co-evolutionary genetic algorithms approach to detect video game bugs," *Journal of Systems and Software*, vol. 188, p. 111261, 2022.

[21] S. G. Ansari, I. Prasetya, D. Prandi, F. M. Kifetew, M. Dastani, F. Dignum, and G. Keller, "Model-based player experience testing with emotion pattern verification," in *International Conference on Fundamental Approaches to Software Engineering*. Springer Nature Switzerland Cham, 2023, pp. 151–172.

[22] T. Ahumada and A. Bergel, "Reproducing bugs in video games using genetic algorithms," in *2020 IEEE Games, Multimedia, Animation and Multiple Realities Conference (GMAX)*. IEEE, 2020, pp. 1–6.

[23] J. Bergdahl, C. Gordillo, K. Tollmar, and L. Gisslén, "Augmenting automated game testing with deep reinforcement learning," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 600–603.

[24] F. Macklon, M. R. Taesiri, M. Viggiato, S. Antoszko, N. Romanova, D. Paas, and C.-P. Bezemer, "Automatically detecting visual bugs in html5 canvas games," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–11.

[25] R. Mawhorter and A. Smith, "Softlock detection for super metroid with computation tree logic," in *Proceedings of the 16th International Conference on the Foundations of Digital Games*, 2021, pp. 1–10.

[26] E. Castellano, X.-Y. Zhang, P. Arcaini, T. Takisaka, F. Ishikawa, N. Ikehata, and K. Iwakura, "Explaining the behaviour of game agents using differential comparison," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–8.

[27] J. Pfau, A. Liapis, G. N. Yannakakis, and R. Malaka, "Dungeons & replicants ii: automated game balancing across multiple difficulty dimensions via deep player behavior modeling," *IEEE Transactions on Games*, 2022.

[28] C. Politowski, F. Petrillo, G. ElBoussaidi, G. C. Ullmann, and Y.-G. Guéhéneuc, "Assessing video game balance using autonomous agents," *arXiv preprint arXiv:2304.08699*, 2023.

[29] C. Paduraru, M. Paduraru, and A. Stefanescu, "Rivergame-a game testing tool using artificial intelligence," in *2022 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2022, pp. 422–432.

[30] G. Liu, M. Cai, L. Zhao, T. Qin, A. Brown, J. Bischoff, and T.-Y. Liu, "Inspector: Pixel-based automated game testing via exploration, detection, and investigation," in *2022 IEEE Conference on Games (CoG)*. IEEE, 2022, pp. 237–244.

[31] R. Marczak, P. Hanna, J.-L. Rouas, J. Van Vught, and G. Schott, "From automatic sound analysis of gameplay footage [echos] to the understanding of player experience [ethos]: an interdisciplinary approach to feedback-based gameplay metrics," in *40th International Computer Music Conference (ICMC) joint with the 11th Sound & Music Computing conference (SMC)*, 2014.

[32] K. Chen, Y. Li, Y. Chen, C. Fan, Z. Hu, and W. Yang, "Glib: towards automated test oracle for graphically-rich applications," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 1093–1104.

[33] B. Jiang, W. Wei, L. Yi, and W. Chan, "Droidgamer: Android game testing with operable widget recognition by deep learning," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2021, pp. 197–206.

[34] I. Prasetya, F. Pastor Ricós, F. M. Kifetew, D. Prandi, S. Shirzadehhajimahmood, T. E. Vos, P. Paska, K. Hovorka, R. Ferdous, A. Susi *et al.*, "An agent-based approach to automated game testing: an experience report," in *Proceedings of the 13th International Workshop on Automating Test Case Design, Selection and Evaluation*, 2022, pp. 1–8.

[35] G. Rani, U. Pandey, A. A. Wagde, and V. S. Dhaka, "A deep reinforcement learning technique for bug detection in video games," *International Journal of Information Technology*, vol. 15, no. 1, pp. 355–367, 2023.

[36] M. R. Taesiri, M. Habibi, and M. A. Fazli, "A video game testing method utilizing deep learning," *Iran Journal of Computer Science*, vol. 17, no. 2, 2020.

[37] C. Sun, Z. Zhang, B. Jiang, and W. Chan, "Facilitating monkey test by detecting operable regions in rendered gui of mobile game apps," in *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2016, pp. 298–306.

[38] J. Yu, Y. Wu, X. Xie, W. Le, L. Ma, Y. Chen, J. Hu, and F. Zhang, "Gamerts: A regression testing framework for video games," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1393–1404.