A Mapping of Recording-based Game Test Automation Tools

Vinícius Mioto
Universidade Federal do Paraná
Curitiba, Brazil
mioto@ufpr.br
ORCiD: 0000-0003-1343-7183

Fabio Petrillo
École de technologie supérieure - ÉTS Montréal
Montréal, Canada
fabio.petrillo@etsmtl.ca
ORCiD: 0000-0002-8355-1494

***Abstract*—As the gaming industry grows, there is a greater need for high-quality, complex games, which requires efficient and scalable testing methods. This paper maps recording-based game test automation tools that record and replay user actions to make testing easier and reduce manual work. We analyzed 20 tools mentioned in industry blogs and grouped them into game-specific and general-purpose categories. We also evaluated their compatibility, input simulation features, scripting options, and integration with game engines.**

Our findings indicate that general-purpose tools, the most frequently mentioned in blog sources, offer broad adaptability across platforms but are typically limited to user interface and visual testing. In contrast, game-specific tools provide deeper integration with game engines, supporting access to in-game mechanics and making them better suited for testing gameplay aspects. Our observations intend to conduct industry and academic efforts toward more reliable, scalable, and cost-effective approaches to automated game testing.

***Index Terms*—testing, tools, automation, videogame, digital games, systematic mapping, gray literature**

I. Introduction

The gaming industry has experienced exponential growth over the past decades, with projections estimating its worth at US$300 billion by 2028, driven by rapid technological advancements and evolving business models in game distribution and monetization[1]. Thus developers are tasked with producing larger, more intricate games that not only meet but exceed player expectations in terms of performance and quality [1].

To address this demands, extensive work across various roles, from art design to gameplay mechanics, and perhaps most critically, in ensuring that the game is thoroughly tested [2]. Yet, the gaming industry continues to rely heavily on manual testing, where human testers assess gameplay, identify bugs, ensure performance, and evaluate overall user experience [3]. Manual testing, by nature, is time-consuming, prone to human error, and difficult to reproduce at scale [4].

In this paper, we identify testing automation tools of a specific approach called "recording-based", that allows testers to record actions within the game and replay these actions for testing. This study explore the tools for game testing automation, focusing on identifying the available tools and evaluating their features. By targeting and analyzing blog articles through a structured selection process, we compiled a list of tools supporting automated tests with recording-and-replay. Once we extract a list of tools from the selected blogs, we analyzed their official documentation to examine the main features that can be used for video game testing.

The rest of this paper is organized as follows. The Section II delves into the fundamentals of recording-based testing. Section III describes the methodology used to identify and analyze the tools. Section IV shows the acquired list of tools, followed by Section V that describes the main features of them. Section VII contains the discussion of the results, while the Section VIII presents the threats to validity of our study. Finally, Section IX concludes this paper.

II. Recorded-Based Testing

Testing is one of the steps in the verification process to ensure the quality and improvement of software [5]. Automated Testing consists on the use of tools and algorithms to test various aspects of the application without human intervention [5], [6]. In video game development the testing involves primarily manual interactions with the game to check for technical faults, and also evaluate the overall gaming experience [2], [3]. Hence, tools and techniques have been proposed to support the automation of game testing to relieve testers from time-consuming and repetitive tasks [7].

Albaghajati and Ahmed [8] suggest a comprehensive framework to classify and compare automated game testing approaches. We will focus on the "scenario-based approach", more specific on the *record-and-replay* category, which the tools execute tests based on predefined sequences of human made actions or human-requested actions.

Testers can use scripting-based tools, that are often integrated into the game engine, to automate repetitive tasks and expand the test coverage without manual intervention. This tools require the creation of scripts, which need some level of programming expertise and time to write the code [9].

Specifically, recorded-based tools offer the possibility of recording player interactions during gameplay, usually these tools generate the script containing instructions to reproduce the inputs, so they can be replayed automatically [8], [10], which might reduce the time and effort needed for test

[1]https://www.pwc.com/gx/en/issues/business-model-reinvention/outlook/insights-and-perspectives.html

development, especially in projects with frequent updates or a vast array of elements to test.

The Venn diagram in Figure 1 illustrates the relationship between software testing, game testing, and automated testing tools. The diagram emphasizes that recording tools are a subset of script-based tools, offering ease of use while still relying on underlying automation scripts for test execution. Furthermore, there are script-based and recording tools that can be used for both general software testing and game testing.

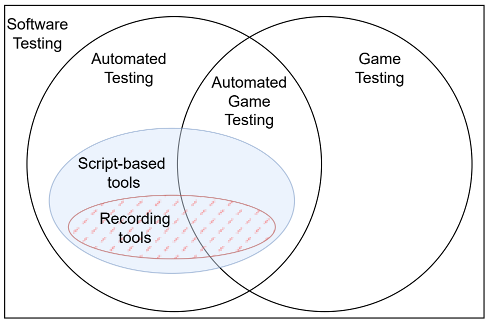

Fig. 1. Venn Diagram - Testing and its tools - A Venn diagram showing the relationships between Software Testing, Automated Testing, Game Testing, Automated Game Testing, Script-based tools, and Recording tools, highlighting overlaps in test automation for games.

Due to this overlap of recording-based automation tools in general software and game-specific testing, our study contains tools that are designed to work in both scenarios, as detailed on Section IV. Given the challenges associated with applying automated testing in video games [2], [3], recording-base tools intent to provide a solution by ensuring extensive test coverage in less time.

Albaghajati and Ahmed [8] show that scenario-based tools aim to achieve four testing objectives, they defined these objectives as follows:

- Functional Correctness: ensures that the game's core mechanics, player actions, and progression work as intended, providing a reliable gameplay experience;
- Game Design Correctness: assesses gameplay structure and balance, verifying that rules are clear, levels accessible, and objects correctly placed to support intended player experiences.
- Visual Correctness: confirms visual consistency by identifying graphical issues, ensuring accurate animations, and verifying correct display of UI elements;
- Multiplayer Stability: focuses on reliable and fair multiplayer interactions, including low-latency connections, stable servers, and security against exploits.

III. Study Design

We aim to adopt a systematic mapping approach to explore and compare recorded-based game testing tools. To achieve this, we address the following key research questions:

- $RQ1$ - What are the recorded-based game test automation tools?
- $RQ2$ - What are the main features of the identified tools?

The study employed a multi-phase and systematic search process to collect a list of tools (see Figure 2), we defined a iterative process, where the initial data come from 20 blogs [11]. Therefore, this study aligns with established guidelines for conducting systematic mapping reviews from the grey literature [12]. The search was completed on 2024/10/27.

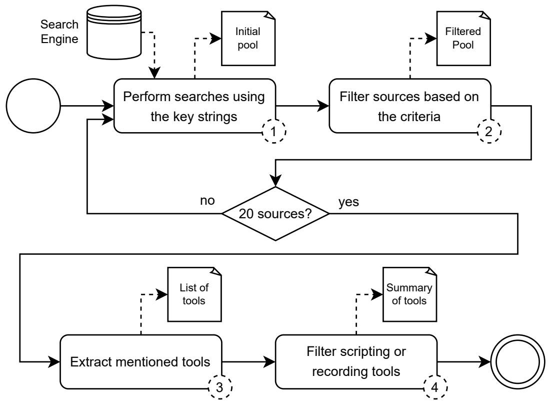

Fig. 2. Tool selection process - A flowchart illustrating the process of selecting game testing tools, from performing searches and filtering results, to extracting and classifying tools for a final list of 20.

Step 1: We started by performing searches on Google to find blogs that could bring us game test automation tools or frameworks that allow developers to automate their testing tasks. The key strings used for the searches were:

- $S1$ - Video game automated testing tools
- $S2$ - Game test automation framework

Step 2: We read the blogs articles and filtered them based on the following inclusion and exclusion criteria:

- Inclusion Criteria
  - *IC1*: blog posts that cite or compare game test automation tools;
  - *IC2*: blog posts that discuss game test automation frameworks focused on recording-and-replay;
  - *IC3*: blogs posts that include a review, case studies, or testimonial of a tool for automating game test;
- Exclusion Criteria
  - *EC1*: blog posts older than 5 years;
  - *EC2*: blog posts not written in English;
  - *EC3*: blog posts about gamification.

We performed the *Steps* 1 and 2 until we get 20 filtered blogs that satisfy our criteria, all the selected sources are included on Table I.

Step 3: In this step, we extracted the tools mentioned in the blog posts and we got a list of 42 tools. Since the depth of descriptions provided in the blogs varied, this initial list required further refinement.

Step 4: We searched for the main website and documentation of the tools to filter only those that support recording-based tasks as stated in Section II, this gave us the final list that contains 20 tools ready for deeper analysis.

Finally, we also examined the official documentation to analyze the main features of each tool in terms of supporting,

TABLE I
LIST OF BLOG POSTS

Ref	Title	Source	IC1	IC2	IC3
[13]	Game Test Automation Tools – A Comprehensive Review and Comparison	iXie	✓	✓	
[14]	Top 6 Game Testing Tools You Need to Know	Test Sigma	✓	✓	
[15]	The Top Game Test Automation Tools You Need to Know About	iXie	✓		
[16]	Automated Game Testing Tools: 10 Types Worth Knowing	modl.ai	✓	✓	
[17]	What Tools do You Need to Automate Video Game Testing?	zappletech	✓	✓	
[18]	How to Automate Video Game Tests	Test Guild	✓		✓
[19]	Game Automation Testing: Things to Consider Before You Go to Market	QAble	✓	✓	
[20]	Game Testing Automation 101: Basic Tips and Strategies (with Case Studies)	VNEXT Global	✓	✓	✓
[21]	A Practical Guide to Test Automation Tools for Mobile Games	TestDevLab	✓		✓
[22]	Level Up Your Game Development: The Power of Game Test Automation	T-Plan	✓		
[23]	Adding test automation to your game development project	AltTester			✓
[24]	Top 10 Game Testing Tools Every Developer Should Know About	KiwiQA	✓	✓	
[25]	Video Game Test Automation: Factors to Consider Before Starting	iXie	✓		
[26]	Game Testing Tutorial: A Comprehensive Guide With Best Practices And Examples	LambdaTest	✓	✓	✓
[27]	10 Best Mobile Game Testing Tools in 2024	HeadSpin	✓		
[28]	Enhance the performance of mobile games with automation testing	QAonCloud	✓	✓	
[29]	Automating Gameplay with TestComplete	SmartBear			✓
[30]	Is Game Automation the Next Generation of Testing?	QAble	✓		
[31]	Appium together with AltTester Unity SDK	Medium			✓
[32]	Automating Mobile Game Testing	Yarsa Labs	✓		

user inputs, integrations, scrips generation, and the testing objectives defined by Albaghajati and Ahmed [8].

IV. RQ1 - What are the recorded-based game test automation tools?

Table II contains the list of the 20 tools, mentioned in the blogs, that support recording-based testing. By checking the documentations, we notice that there are two categories of tools: (i) Game-specific tools, and (ii) General-purpose tools, so we indicate as **(GS)** the tools that are specifically tailored for game testing. The table shows that the blogs mentioned 18 tools that can be used for general-purpose software testing and only 2 tools that are game-specific tools.

TABLE II
MENTIONED TOOLS FOR VIDEO GAME TESTING

ID	Tool	References	#
1	Appium	[14], [15], [17]–[22], [24]–[28] [30]–[32]	16
2	Selenium	[14]–[19], [22], [24], [27], [28]	10
3	TestComplete	[14]–[17], [19], [20], [22], [24] [27], [29]	10
4	AltTester **(GS)**	[13], [14], [21], [23], [31], [32]	6
5	GameDriver **(GS)**	[13], [18], [20], [22]	4
6	AirTest [2]	[13], [21]	2
7	ZAPTEST	[13]	1
8	TestSigma	[14]	1
9	BrowserStack	[16]	1
10	Ranorex	[16]	1
11	Applitools	[19]	1
12	Repeato	[21]	1
13	Kobiton	[21]	1
14	Katalon	[21]	1
15	TestRigor	[21]	1
16	Perfecto	[21]	1
17	TestGrid	[21]	1
18	T-Plan Robot	[22]	1
19	LambdaTest	[26]	1
20	HeadSpin	[27]	1

The frequency of these mentions provides a glimpse into the perceived popularity and adoption within parts of the game development community. Appium, Selenium, and TestComplete appear on the top 3, showing their recognition among the selected sources, all of them are general-purpose testing tools, and they have 10 or more mentions. For instance 16 of 20 blogs (80%) mentioned Appium for video game testing.

Moreover, 14 tools were cited only once each, which suggests niche use cases or limited awareness. TestSigma [14], T-Plan [22], LambdaTest [26], and HeadSpin [27] appear only in the blogs from their own company. AltTeser [23] and TestComplete (from SmartBear) [29] also have their own blog, although other blog posts cite them, as shown in the Table II. These 6 companies maintain a blog as part of a marketing strategy to compare their tool with others or to sustain a community with hints, use cases, updates, and other topics related to their product.

V. RQ2 - What are the main features of the identified tools?

We collect and organize the main features of the identified tools to allow easy comparisons of them, through an examination of the official documentation of these tools. Due to the recording-based scope of this work, all tools are designed to test User Interface (UI) with the aim to ensure the correct interface functionality. Yet, all the selected tools provide the option to generate testing reports, which is an important feature for the workflow.

The other features presented more differences for each tool, and therefore we describe and compare them in the next subsections.

[2]Although AirTest can be used for general-software testing, documentation presents a dedicated section for game testing

A. License, Supported Languages, and Operating Systems

Table III summarizes the licenses of each tool, the supported languages, and the platform of use. In terms of licensing, 7 tools are classified as open-source software, at least for a community version. We found 4 tools with Apache License, 2 with MIT license, and 1 with GPL (GNU General Public License). The other 13 tools have proprietary licenses.

TABLE III
LICENSE, LANGUAGES, AND SUPPORTED PLATFORMS

ID	Tool	License	Language	Windows	macOS	GNU/Linux	Cloud
1	Appium	Apache	Multiple	✓	✓	✓	
2	Selenium	Apache	Multiple	✓	✓	✓	
3	TestComplete	Proprietary	Multiple	✓			
4	AltTester	GPL	Multiple	✓	✓	✓	
5	GameDriver	Proprietary	C#	✓	✓	✓	
6	Airtest	Apache	Python	✓	✓	✓	
7	ZAPTEST	Proprietary	Multiple	✓	✓	✓	
8	TestSigma	Apache	Java	✓	✓	✓	✓
9	BrowserStack	MIT	Multiple	✓	✓	✓	✓
10	Ranorex	Proprietary	Multiple	✓			
11	Applitools	Proprietary	Multiple				✓
12	Repeato	Proprietary	JavaScript	✓	✓		
13	Kobiton	Proprietary	Multiple	✓	✓		✓
14	Katalon	Proprietary	Groovy	✓	✓	✓	✓
15	TestRigor	MIT	JavaScript	✓	✓	✓	✓
16	Perfecto	Proprietary	Multiple				✓
17	TestGrid	Proprietary	Multiple				✓
18	T-Plan Robot	Proprietary	Java	✓	✓	✓	
19	LambdaTest	Proprietary	Multiple				✓
20	HeadSpin	Proprietary	Multiple				✓
				15	13	11	10

We notice that 13 tools support 3 or more programming languages, we classified them as "multiple" in the fourth column of Table III. With this feature, developers can write scripts or even modify the record-generated scripts using their preferred language, ensuring integration with their workflows.

The right side of the table shows the platform of use, for instance the 15 tools that are not cloud-only present a desktop application, such as an integrated development environment (IDE) and a software development kit (SDK). In this scenario, Windows is the most supported operating system (OS) for this applications, with 15 tools in total, followed by macOS and GNU/Linux, with 13 and 11, respectively.

In addition to that, 10 tools offer the possibility to create and execute the tests in a cloud environment. They allow the teams to execute tests in different devices and platforms, by using emulators or real devices on cloud, so the developers do not need to use their own infrastructure for testing [21].

B. Testing Platforms

General-purpose software testing tools are designed to work with a wide range of applications and platforms beyond video games. Table IV shows that these tools typically feature cross-platform compatibility. As we can see, 17 tools can be used for mobile applications testing. Additionally, mobile game testing is specifically discussed in 5 blogs [19], [21], [27], [28], [32]. There are 16 tools that allow web applications testing, and 12 that support desktop software testing. Only 3 tools are limited to one specific environment, the other 17 can be used at least for two platforms.

TABLE IV
SUPPORTED TESTING PLATFORMS AND ENGINES

		Platforms			Game engines					
ID	Tool	Mobile	Web	Desktop	Unity	Unreal	Custom	Godot	Cocos	Egret
1	Appium	✓	✓	✓						
2	Selenium		✓							
3	TestComplete	✓	✓	✓						
4	AltTester				✓		✓			
5	GameDriver				✓	✓		✓		
6	Airtest	✓	✓	✓	✓	✓	✓		✓	✓
7	ZAPTEST	✓	✓	✓						
8	TestSigma	✓	✓	✓						
9	BrowserStack	✓	✓							
10	Ranorex	✓	✓	✓						
11	Applitools	✓	✓	✓						
12	Repeato	✓								
13	Kobiton	✓								
14	Katalon	✓	✓	✓						
15	TestRigor	✓	✓	✓						
16	Perfecto	✓	✓							
17	TestGrid	✓	✓	✓						
18	T-Plan Robot	✓	✓	✓						
19	LambdaTest	✓	✓	✓						
20	HeadSping	✓	✓							
		17	16	12	3	2	2	1	1	1

Differently to general-purpose software testing tools, the analyzed game-specif tools are designed to work with game engines. For instance, AltTester is compatible with Unity and with custom engines, and GameDriver supports Unity, Unreal, and Godot. AltTester and GameDriver supplement their written documentation with quickstart videos available on YouTube [3][4], allowing users to visually grasp the process and see real-time demonstrations of how to integrate the tool with engines and how to create the first automated tests in their own games.

Although we classified AirTest as a general-purpose tool, it's documentation shows that beyond the multiple supported platforms such as mobile, web, and desktop, it shows how to setup the tool with Unity, Unreal, Cocos2d-x, Cocos-Creator, and Egrete. The documentation also describes how to setup AirTest SDK with other engines in a "self-integration" tutorial.

Considering the presented differences, the supported platform or the integration with game engines is an important topic when choosing a testing tool to incorporate to the developers workflow [25], [26].

[3]AltTester Unity SDK Tutorial - Get Started with Unity Test Automation https://www.youtube.com/watch?v=L4yAgv8Jc8s

[4]Game Testing Tutorial Series pt 1: Adding GameDriver to your Unity Project https://www.youtube.com/watch?v=HNajsVdQemY

C. Workflow Integrations

We examined the documentation to identify features that focus on integrations that enhance functionality for development and testing workflows, we shown them on Table V.

TABLE V
WORKFLOW INTEGRATIONS OF GAME TESTING TOOLS

ID	Tool	Use plugins/extensions	CI/CD support	Integrated with Selenium	Integrated with Appium	Appium-based	Selenium-based
1	Appium	✓					✓
2	Selenium	✓	✓				
3	TestComplete	✓	✓	✓	✓		
4	AltTester	✓		✓		✓	
5	Game Driver	✓	✓	✓	✓		
6	Airtest	✓		✓	✓		
7	ZAPTEST	✓		✓			
8	TestSigma	✓	✓				
9	BrowserStack	✓	✓	✓	✓		
10	Ranorex	✓	✓	✓		✓	
11	Applitools	✓	✓	✓	✓		
12	Repeato		✓		✓		
13	Kobiton	✓	✓			✓	
14	Katalon	✓	✓	✓		✓	
15	TestRigor	✓	✓				
16	Perfecto	✓	✓	✓	✓		
17	TestGrid	✓	✓			✓	✓
18	T-Plan Robot	✓	✓	✓			
19	LambdaTest	✓	✓	✓	✓		
20	HeadSpin	✓	✓	✓	✓		
		19	16	13	9	5	2

One of the most frequently mentioned features in the workflow and integration category is the possibility to use plugins and extensions with the selected tool. In total, 19 tools mentioned this possibility in the documentation. We consider this to be a relevant topic for the usage, as it allows developers to integrate the selected tool with other tools or frameworks, so they can overcome a limitation or include new automation functionalities in the current workflow [27]. For example, Lovreto et al. [33] conducted a study using Appium combined with OpenCV[5] to perform testing in 16 mobile games available on Google Play Store, such as Sonic Boom, Moto Rider GO, Ludo King, Spider Card, etc. OpenCV was used to find and interact with UI elements like buttons, characters, items, and enemies through image recognition.

Another included feature for the workflow, mentioned by 16 tools, is the Continuous Integration/Continuous Deployment (CI/CD) support which enables a more efficient and streamlined development process, so the issues can be flagged and resolved quickly, promoting better collaboration and workflows [15], [16], [31]. The cited platforms for this feature are GitHub Actions, GitLab CI/CD, Jenkins, and Azure DevOps (Pipelines), they provide functionalities that streamline builds, testing, and deployment.

[5]OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

The last four features of the workflow integration category were included on Table V due the number of citations to Selenium and Appium in the documentations. We notice that 13 tools offer integration with Selenium, and 9 tools offer integration with Appium. Yet, 5 tools are based on Appium, and 2 are based on Selenium. Therefore Selenium's and Appium's features are available in the tools that allows integrations or are build on top of them. This highlight the importance of both tools for testing, and give us more reasons for they to be the most cited tools on the blogs as well.

Tools based on or integrated with Appium and Selenium interact with UI components in the same way as these frameworks. Appium primarily uses component locators, such as resource IDs for Android and iOS class chains, while Selenium interacts through the Document Object Model (DOM). Tools, like ZAPTEST and AirTest use image recognition techniques for locating buttons, menus, input fields, and other items [13], [21]. Other tools did not mention in the documentation how exactly they interact with UI components.

D. User Inputs and Scripts Generation

We analyzed the features related to the types of user inputs simulations, which is used for interacting with UI components during the testing, and also available methods for creating the tests, beyond recording [26]. These tools are summarized on Table VI.

Although every tool can interact with UI components, they differ in the user inputs simulations aspect. Thus, the supported inputs is relevant for game testing aspects [18]. Inputs like mouse (e.g. clicking, holding, dragging and dropping, etc.) and touch (e.g. tapping, swiping, pinching, etc.) are the most common, with 18 tools. Followed by keyboard inputs (e.g. key down, holding, and sequence of keys) with 16 tools, we did not assign this feature in the table for the tools that only allow to enter key strings on text fields such as forms. Joystick inputs are supported only by the game-specif tools, and GameDriver also allows extended reality (XR) inputs.

Besides the actions recording and manual scripting, we detected 15 tools that allow the user to use the GUI to create test cases. This feature, usually called "no-code", is common in tools that have an IDE or a web-app that offer a way to create tests by dragging and dropping commands to structure the sequence of test actions [21]. The documentation highlights that this feature can save time when creating tests and is helpful for those who do not have expertise in programming.

There are 9 tools labeled as "low-code", they also cater to users with minimal programming experience, considering that it is easy and friendly to create and understand the test scripts, they usually provide writing the test with key-words (e.g. *enter* menu, *click* button, *press* play, etc.). Additionally, ZAPTEST, TestSigma, testRigor, and LambdaTest are tools that allows

TABLE VI
USER INPUTS AND SCRIPTS GENERATION OF GAME TESTING TOOLS

		Inputs simulations				Scripting		
ID	**Tool**	**Mouse/Touch**	**Keyboard**	**Joystick**	**XR**	**Using GUI**	**Using low-code**	**Using LLM**
1	Appium	✓	✓					
2	Selenium	✓	✓					
3	TestComplete	✓	✓			✓	✓	
4	AltTester	✓	✓	✓				
5	GameDriver	✓	✓	✓	✓			
6	Airtest	✓	✓			✓		
7	ZAPTEST	✓	✓			✓		✓
8	TestSigma					✓	✓	✓
9	BrowserStack	✓	✓			✓	✓	
10	Ranorex	✓	✓			✓	✓	
11	Applitools	✓	✓			✓	✓	
12	Repeato	✓				✓		
13	Kobiton	✓				✓		
14	Katalon	✓	✓			✓	✓	
15	TestRigor	✓	✓			✓	✓	✓
16	Perfecto	✓	✓			✓	✓	
17	TestGrid	✓	✓			✓		
18	T-Plan Robot	✓	✓			✓	✓	
19	LambdaTest	✓	✓			✓		✓
20	HeadSpin	✓				✓		
		19	16	2	1	16	9	4

TABLE VII
TOOLS AND TESTING OBJECTIVES ACHIEVEMENTS

ID	**Tool**	**Functional correctness**	**Visual correctness**	**Multiplayer stability**	**Game design correctness**
1	Appium	✓			
2	Selenium	✓			
3	TestComplete	✓	✓	✓	
4	AltTester	✓		✓	✓
5	Game Driver	✓		✓	✓
6	Airtest	✓	✓	✓	✓
7	ZAPTEST	✓	✓	✓	
8	TestSigma	✓	✓	✓	
9	BrowserStack	✓	✓	✓	
10	Ranorex	✓	✓		
11	Applitools	✓	✓		
12	Repeato	✓	✓		
13	Kobiton	✓	✓	✓	
14	Katalon	✓	✓	✓	
15	TestRigor	✓	✓	✓	
16	Perfecto	✓	✓	✓	
17	TestGrid	✓	✓	✓	
18	T-Plan Robot	✓	✓		
19	LambdaTest	✓	✓	✓	
20	HeadSpin	✓	✓	✓	
		20	16	14	3

the user to create the test with plain english, they are powered by large language models (LLMs), so the developers must specify the test cases with a natural language prompt or they be assisted by an AI model to write the scripts [21].

VI. Tools and Testing Objectives Achievements

Table VII compile the achievements of the testing objectives for each tool, as defined by described on Section II and defined by Albaghajati and Ahmed [8].

A. Visual Correctness and Multiplayer Stability

We noticed that 16 tools provide support for sisual correctness testing, a method used to ensure that the graphical elements and UI components of a game appear as expected on different screens, devices, and resolutions. We highlight the importance of visual aspects in game development, where aesthetics and visual consistency play a significant role in the player's experience [27]. As mentioned in Section V, the ability to integrate tools with other frameworks, such as Appium with OpenCV [33], might be useful to test visual aspects, although it requires more steps to setup the tools and the testing environment.

Regarding the game-specif tools, AltTester's documentation does not mention any feature related to visual testing. GameDriver's documentation, show one simple example of visual aspects by verifying the color of a component is the same as a RGB code (red, green, and blue values). In terms of comparison, the other tools are equipped with image recognition features, which allow them to detect visual discrepancies.

Parallel executions are supported by 14 tools. They allow tester to simulate real-world multiplayer environments by evaluating how multiple users or devices interact with the game server concurrently. For example, the automation approach for Unity-based multiplayer card games, introduced by Sajid Shaik and Pragna P. Pal [34], addresses the challenges associated with unpredictable game states in a multi-player context. This approach leverages AirtestIDE manager to identify and customize game elements like cards and regions dynamically.

B. Functional Correctness and Game Design Correctness

All 20 tools support functional correctness testing to some extent. However, general-purpose tools are limited to testing UI elements, such as buttons, menus, forms, and basic gameplay interactions. Tibell and Kholi [35] conducted a comparative study of Selenium and TestComplete, where they designed and implemented four test cases for an e-commerce application and a web-based game. The game testing cases included verifying the play button, game startup, and character movements (by pressing arrow keys). The authors highlight the limitations of Selenium which does not provide access to game-level objects due to its reliance on DOM-based structure. Tuovenen et al. introduced MAuto, an automated mobile game testing tool that records and replays user interactions on Android games using the Appium and computer vision techniques. MAuto was tested on the mobile game Clash of Clans, it successfully replayed the recorded tutorial actions in the game.

Game-specific tools enhance functional correctness testing by providing direct access to game-level objects, functions, and properties within the game engine. This deeper integration enables testers to load specific scenes, locate and manipulate in-game components, and simulate complex events to verify outcomes directly [18]. With this capability, game-specific tools allow comprehensive testing of intricate mechanics, including elements like acceleration, velocity, character movement, and AI responses, ensuring that core game actions and processes operate as intended.

Due to the lack of integration with the game engine, general-purpose tools cannot access to gameplay structure or rules, thus they are not suited for testing game design correctness. In contrast, game-specific tools like AltTester, GameDriver, and AirTest support this type of test by providing deep integration within the game engine, allowing testers to interact directly with game rules, level layouts, and object placements. These tools enable precise control over various game configurations, such as scene navigation, camera angles, and input handling, which are essential for assessing gameplay balance and structural coherence [21], [32]. Additionally, their documentation includes tutorials and examples with pre-made games, making it more intuitive for testers to evaluate game-specific scenarios and design elements directly within the game environment.

VII. Discussion

Although the general-purpose software testing tools are compatible with different platforms, they still focus on specific environments that might not work for complex game projects. For instance, Repeato and Kobiton work only on mobile devices, and Selenium is limited to web applications. As a result, these tools are not well-suited for integration with game engines and are more commonly restricted to web-based or mobile games. Moreover, general-purpose tools, except Airtest, require additional setup for game environments, as the documentation does not have a dedicated section for integrating with video game projects or examples of use cases for game testing [27]. Game-specific tools offer comprehensive documentation dedicated to game-testing tasks, making them easier to adopt. Additionally, they allow game developers to benefit from community support and efficient troubleshooting due to their focus on video games [14].

Recording-based game test automation tools can offer significant advantages to video game quality assurance (QA) teams for automating repetitive UI tasks by recording and replaying gameplay scenarios [8], [28]. This automation saves time compared to manual testing, reduces human error, and ensures consistent execution across multiple testing sessions [28]. These tools are versatile and can be broadly applied across different game types and genres, enhancing the adaptability and effectiveness of the testing process [8]. Nevertheless, these tools can be costly to maintain, as updates to game mechanics or UI often necessitate re-recording or script adjustments, adding to the maintenance burden [16], [27].

In this context, we also highlight the blogs [13], [16], [19], [30] that mention testing with AI-based models, such as machine learning and reinforcement learning, to support specifically the gameplay test. In fact, game companies like Electronic Arts (EA), UBISOFT, KING, and Activision invested in this approach, their results show how the models can adapt to different scenarios during the gameplay and enhance the test coverage [36]–[38]. However, the demand for computational resources, specialized expertise, and substantial time investment is significantly high.

VIII. Threats to validity

This study faces threats to validity that may influence the comprehensiveness and accuracy of our findings. Primarily, selection bias is a concern, as we sourced our initial list of tools from blog articles, which may emphasize more popular or recently updated tools. In addition to that, there are 6 blogs included in our work that are written by the same companies that create the tools, as discussed on Section IV.

Another threat to validity is that even official documentation can vary in depth and focus, particularly since 18 are designed for general software testing rather than specifically for video games. Hence, the documentation may emphasize web, mobile, or desktop applications rather than the unique requirements of game testing, potentially impacting the quality of information related to game-specific features. We addressed this by focusing on widely applicable features, though we acknowledge that some specialized tools and features may exist outside the scope of this study.

IX. Conclusion

This study maps recording-based game test automation tools, focusing on their features, platform compatibility, and suitability for game-specific requirements. We compiled a set of tools that offer diverse functionalities for automated game testing. Our mapping reveals the main options available to developers seeking to implement automation in their workflows, according to blog posts. The systematic approach used in this study also brought a comparative analysis of the features for game testing that these tools offer.

While many general-purpose automation tools can be used for game testing, they are typically limited to validating basic UI elements like menus, buttons, and forms. In contrast, game-specific tools such as AltTester and GameDriver enhance testing by integrating deeply with game engines, which allows QA teams to more effectively testing in-game objects, supporting more thorough gameplay testing beyond what general-purpose tools can achieve.

This study provides insights into recording-based test automation tools, detailing their main features and contributions to game testing. By mapping the landscape of these tools and offering a comparison, it serves as a resource for game developers seeking to implement or enhance automation in their testing workflows.

Future work may focus on the continued evaluation of these tools with a survey of game QA practitioners. The survey can generate new results to evaluate the features and usage of the tools and to understand if they really address the demands of video game testing. Moreover, future work may concentrate on the potential for new solutions explicitly tailored for game testing.

Acknowledgements

This research was supported by Mitacs (Canada) and Fundação Araucária (Brazil) through the Globalink Research Internship program. Their support made this international research opportunity possible, fostering collaboration between their countries. We sincerely thank both organizations for their commitment to advancing research and education.

References

[1] G. C. Ullmann, C. Politowski, Y.-G. Guéhéneuc, and F. Petrillo, "What makes a game high-rated? towards factors of video game success," in *Proceedings of the 6th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation*, ser. GAS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 16–23. [Online]. Available: https://doi.org/10.1145/3524494.3527628

[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 Computer Society, May 2021, pp. 33–36. [Online]. Available: https://www.computer.org/csdl/proceedings-article/ast/2021/356700a090/1tB7rABhRDO

[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*, ser. GAS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 37–43. [Online]. Available: https://doi.org/10.1145/3524494.3527627

[4] C. Politowski, F. Petrillo, G. C. Ullmann, and Y.-G. Guéhéneuc, "Game industry problems: An extensive analysis of the gray literature," *Information and Software Technology*, vol. 134, p. 106538, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584921000252

[5] K. Naik and P. Tripathy, *Software Testing and Quality Assurance: Theory and Practice*, 2nd ed. Wiley Publishing, 2018.

[6] V. Garousi and M. V. Mäntylä, "When and what to automate in software testing? a multi-vocal literature review," *Information and Software Technology*, vol. 76, pp. 92–117, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584916300702

[7] R. Coppola, T. Fulcini, and F. Strada, "Know your bugs: A survey of issues in automated game testing literature," in *2024 IEEE Gaming, Entertainment, and Media Conference (GEM)*, 2024, pp. 1–6.

[8] A. Albaghajati and M. Ahmed, "Video game automated testing approaches: An assessment framework," *IEEE Transactions on Games*, vol. 15, no. 1, pp. 81–94, 2023.

[9] M. Ostrowski and S. Aroudj, "Automated regression testing within video game development," *GSTF Journal on Computing (JoC)*, vol. 3, 08 2013.

[10] J. Hernández Bécares, L. Costero Valero, and P. P. Gómez Martín, "An approach to automated videogame beta testing," *Entertainment Computing*, vol. 18, pp. 79–92, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1875952116300234

[11] A. Rainer and A. Williams, "Using blog articles in software engineering research: Benefits, challenges and case–survey method," in *2018 25th Australasian Software Engineering Conference (ASWEC)*, 2018, pp. 201–209.

[12] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, pp. 101–121, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584918301939

[13] iXie. (2023) Game test automation tools – a comprehensive review and comparison. [Online]. Available: https://www.ixiegaming.com/blog/comprehensive-review-game-test-automation-tools/

[14] T. Sigma. (2024) Top 6 game testing tools you need to know. [Online]. Available: https://testsigma.com/blog/game-testing-tools/

[15] iXie. (2023) The top game test automation tools you need to know about. [Online]. Available: https://www.ixiegaming.com/blog/the-top-game-automation-tools/

[16] modl.ai. (2024) Automated game testing tools: 10 types worth knowing. [Online]. Available: https://modl.ai/automated-game-testing-tools/

[17] zappletech. (2021) What tools do you need to automate video game testing? [Online]. Available: https://zapple.tech/blog/test-automation-frameworks/what-tools-do-you-need-to-automate-video-game-testing/

[18] T. Guild. (2020) How to automate video game tests. [Online]. Available: https://testguild.com/automate-video-games/

[19] QAble. (2023) Game automation testing: Things to consider before you go to market. [Online]. Available: https://www.qable.io/blog/game-automation-testing

[20] V. Global. (2023) Game testing automation 101: Basic tips and strategies (with case studies). [Online]. Available: https://vnextglobal.com/category/blog/game-testing-automation-101-tips-case-studies

[21] TestDevLab. (2024) A practical guide to test automation tools for mobile games. [Online]. Available: https://www.testdevlab.com/blog/a-practical-guide-to-test-automation-tools-for-mobile-games

[22] T-Plan. (2024) Level up your game development: The power of game test automation. [Online]. Available: https://www.t-plan.com/level-up-your-game-automation/

[23] AltTester. (2024) Adding test automation to your game development project. [Online]. Available: https://alttester.com/adding-test-automation-to-your-game-development-project/

[24] KiwiQA. (2024) Top 10 game testing tools every developer should know about. [Online]. Available: https://www.kiwiqa.com.au/top-10-game-testing-tools/

[25] iXie. (2023) Video game test automation: Factors to consider before starting. [Online]. Available: https://www.ixiegaming.com/blog/factors-of-game-test-automation-to-consider/

[26] LambdaTest. (2023) Game testing tutorial: A comprehensive guide with best practices and examples. [Online]. Available: https://www.lambdatest.com/learning-hub/game-testing

[27] HeadSpin. (2024) 10 best mobile game testing tools in 2024. [Online]. Available: https://www.headspin.io/blog/best-mobile-game-testing-tools

[28] QAonCloud. (2023) Enhance the performance of mobile games with automation testing. [Online]. Available: https://www.qaoncloud.com/blog/automation-testing-for-mobile-games

[29] SmartBear. (2024) Automating gameplay with test-complete. [Online]. Available: https://smartbear.com/blog/automating-gameplay-with-testcomplete/

[30] QAble. (2024) Is game automation the next generation of testing? [Online]. Available: https://www.qable.io/blog/is-game-automation-the-next-generation-of-testing

[31] Medium. (2023) Appium together with alttester unity sdk. [Online]. Available: https://www.headspin.io/blog/unity-test-framework-for-running-automated-testing

[32] Y. Labs. (2023) Automating mobile game testing. [Online]. Available: https://blog.yarsalabs.com/mobile-game-testing/

[33] G. Lovreto, A. T. Endo, P. Nardi, and V. H. S. Durelli, "Automated tests for mobile games: An experience report," in *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, 2018, pp. 48–488.

[34] P. Pal and S. Shaik, "Automation approach for unity based multi-player card game," Jun. 2021. [Online]. Available: http://dx.doi.org/10.36227/techrxiv.14806149.v1

[35] S. Tibell and M. Kholi, "Choosing the right automated ui testing tool : - a comparative study of selenium and testcomplete," p. 55, 2023.

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

[37] ommy Thompson. (2020) The secret ai testers inside tom clancy's the division. [Online]. Available: https://www.gamedeveloper.com/design/the-secret-ai-testers-inside-tom-clancy-s-the-division

[38] S. F. Gudmundsson, P. Eisen, E. Poromaa, A. Nodet, S. Purmonen, B. Kozakowski, R. Meurling, and L. Cao, "Human-like playtesting with deep learning," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 2018, pp. 1–8.