Video Game Automated Testing Approaches: An Assessment Framework

Aghyad Albaghajati, Moataz Ahmed*

***Abstract*—The video-game industry has recently grown from focused markets to mainstream. The advancements the industry has been enjoying motivated researches to propose techniques and tools to support the activities across the different phases of the game development lifecycle. Game testing is one of the crucial activities within the game development process. Due to the nature of game testing, many automated game testing techniques have been proposed in the literature. However, there is no framework that could be used to aid practitioners in selecting appropriate techniques suitable for their particular development efforts. In this paper we present an attribute-based framework to classify and compare these techniques and provide such aid to practitioners. The framework is also meant to guide researchers interested in proposing new game testing techniques. The paper discusses a number of prominent techniques against the framework. Analysis of the discussion reveals gaps and suggests open points for future research.**

***Index Terms* — Software Testing, Game Development, Playtesting, Game Testing, Assessment Framework**

I. INTRODUCTION

A video game is a highly sophisticated software system with several unique aspects like nondeterministic behaviors, visual presentations, and creative design [1, 2]. Indeed, games are known to be one of the most complex software systems, where several subsystems come along together to make a game correctly functioning, attractive and entertaining [1] [3]. The video game industry has evolved significantly through the years [4].

As video game industry matures, tremendous fanbase and consumers have grown to look to play games with great quality and user experience. According to a study by gamesindustry.biz [5], the total value of the revenue raised in the global game market of games published on PC, Console, Mobile and Web platforms in 2018 was about 134.9 billion dollars. Therefore, quality assurance and verification processes are of high importance in the game development industry [6] [7]. Verifying the quality of a game can be done through applying several activities, one of these activities is playtesting, which is the process of playing through a game and reviewing it [6].

Software testing and fault detection play key role in software development in general [8]. The process of testing software consists of validating and verifying software products to meet the requirements and design, and to ensure that the software works as expected [9].

With the increasing demands for more sophisticated software development, testing became not only crucial but also more difficult [10] [11]. Testing in the software industry, in general, has been a laborious and time-consuming work. Consequently, automated testing became widely adopted to reduce cost and improve quality [12].

The current state of games testing in the industry is to hire human testers to manually play game builds at various stages of the development process [13][14]. However, the complexity of video games necessities test automation. It is noteworthy too that, in comparison with traditional software development, video game quality assurance takes into consideration other dimensions other than correctness and performance, such as testing the fun factor, game balance, physics, level design, multiplayer networking, etc. [15] [16]. Hence, employing automated agents can improve and optimize the playtesting process [17], thus reducing the testing costs. These agents are known to be faster than human testers, where they can explore the game space in much shorter time [18]. Automated playtesting agents are also capable of playing the game repeating the tests multiple times to help game designers and developers during the development process.

Many techniques have been proposed in the literature to automate game testing; some of such techniques have also been applied and used in the game development industry [19]. For example, companies like KING and UBISOFT used AI based agents to test, balance and enhance newly added levels [20] [21]. Similarly, Electronic Arts (EA) used Machine Learning and A* algorithms to find flaws in their games [22] [23]. However, to the best of our knowledge, there is no framework that could be used to aid practitioners in selecting appropriate techniques suitable for their particular development efforts. In this paper we present a framework to facilitate classifying and comparing techniques based on a set of attributes identified as a result of an extensive survey of existing techniques. We also critically assessed and compared the game testing techniques available in the literature based on the framework. The assessment also revealed eye-opening gaps for future research in the area of automated game testing.

The rest of this paper is outlined as follows. In Section II related work is discussed. Research methodology and research questions are presented in Section III. Section IV presents the

* Corresponding author.
Information and Computer Science Department King Fahd University of Petroleum and Minerals, Saudi Arabia.
A. Albaghajati email: g201703510@kfupm.edu.sa
M. Ahmed email: moataz@kfupm.edu.sa

analysis, results and answers for our study's research questions. Section V discusses the implications of this study and suggests future work in the area. Section VI presents the threats to validity of this study. Finally, Section VII concludes our work.

II. Related Studies

We were able to find only very few studies that address analyzing game testing techniques. Redavid and Adil (2011) [11] presented an overview of game code testing techniques. The study [11] focused on the game development process, its artifacts and their testing such as combinatorial testing to test game software, testing flow diagrams to test game behavior from player's perspective, clean room testing, and more software engineering related approaches. However, the study was not targeting automated game testing, where such automated techniques were not fully mature; instead, the study focused on game development life cycle and game testing processes from the perspective of software engineering practices, where most of the discussed processes were manual approaches.

In another study, Roohi et al. (2018) [24] provided a systematic literature review of the intrinsic motivations such as autonomy, curiosity, competence, or domination and their implementation in AI game-playing agents. The authors analyzed several motivations and found that the most targeted one in the literature is curiosity. Moreover, the authors pointed out at the importance of utilizing players' collected data in multiplayer games to allow mixing between different motivations, and to validate, support and improve human-likeness in simulated agents. Nevertheless, the study did not go through automated game testing approaches in the literature. Furthermore, the study focused mainly on player modeling and took a glance at simulation-based testing. Moreover, the study used Google scholar only to find related studies, although other search engines could give more related studies.

Zarembo (2019) [25] presented a short overview of few automated playtesting techniques. The study discussed and categorized the selected studies. However, the author mentioned that his study is not comprehensive, and more analysis and studies are required. Moreover, this study did not discuss the testing objectives of the reviewed approaches.

Thus, to the best of our knowledge, our study is the first detailed study that compares and discusses the available automated game testing techniques in the literature.

III. Research Methodology

In this section we describe the research methodology carried out in this study. Our review research methodology was inspired and conducted by following Kitchenham's guidelines for conducting literature reviews [26]. Hence, in this section we present the steps that we followed in our research from defining the study goal to describing the data extraction process.

A. Study Goal and Research Questions

With the increasing interest in video games and with its growing fanbases, game development companies have been interested in applying automated game testing, where game designers and developers can focus on more creative processes that could advance the game experience rather than testing the game manually, which could take a lot of time and effort [27]. Thus, having an agent that can test a game automatically would increase the game's quality in a faster manner.

The main goal of this study is to investigate and compare the found automated game testing approaches, their applications and their objectives, where we conduct a critical assessment and analysis study using an established attribute-based framework to find answers to our research questions. Moreover, this study points out to open problems in the literature for the future and to further extend the work in this area.

Conducting this study would help us find answers to the following research questions:

- RQ1-What automated game testing approaches are applied in the literature?
  - In answering this research question, we would be able to identify the found automated game testing approaches in the literature, their benefits, and their limitations.
- RQ2-What are the objectives of the automated game testing approaches available in the literature?
  - Answering this research question would investigate the goals of the studied automated game testing approaches, and their support for game verification, balancing, level design, and more.
- RQ3-How do researchers validate their developed automated game testing approaches?
  - Answering this research question would specify game genres used to apply the automated game testing approaches, and whether the approaches are generally applicable or open-source.
- RQ4: What are the shortcomings in the current state of the affairs?
  - This research question is answered by studying and analyzing the studies found and suggesting future work.

B. Search Strategy

To support our study and to find answers to our research questions, we investigated several sources of information. Hence, we collected relevant studies from scientific literature sources only. To collect relevant studies, we focused our search in the following literature search engines and databases: Google Scholar, ACM, Science Direct, IEEE and Springer. Moreover, to be able to cover and collect more sources of interest we used both backward and forward snowballing techniques to collect related studies that were not found while searching in the search engines.

Furthermore, we used the following search strings to collect the relevant studies:

- In Google Scholar, IEEE, ACM, and Springer:
  ("automated" OR "auto*") AND ("game" OR "video game") AND ("verify" OR "verification" OR "testing" OR "game testing" OR "test*" OR "playtest*" OR "playtesting")

- In Science Direct: (due to the limitations of the searching tool in Science Direct database, we modified the search string to be the following):

  ("auto") AND ("video game") AND ("verify" OR "verification" OR "testing" OR "game testing" OR "test" OR "playtest" OR "playtesting")

C. Study Selection and Quality Assessment

The study selection went through several steps as follows:

1) **Initial selection:** The searching process was done on each one of the mentioned databases in Section III-B. The first selection of the studies was based on the article of each study.
2) **Filtering studies:** To find the most relevant studies from the initially collected ones, we applied our quality assessment criteria. Moreover, we went through the collected studies and filtered them based on the contents of their abstract, introduction and conclusion.
3) **Merging:** After filtering the studies, we ended up with a pool of selected studies that are relevant to our research. However, some of them were duplicates due to the outcomes of databases and search engines. Thus, we merged and combined all the found studies under one set of studies with unique and no study duplications.
4) **Snowballing:** To collect more related studies and to make sure that we covered all studies available, we ran backward and forward snowballing processes. Backward snowballing is the process of going through the references list of a study and identifying new papers to include from it [28]. On the other hand, forward snowballing refers to identifying and collecting new related papers based on the papers that are citing the examined paper [28]. The snowballing processes resulted into adding new papers that were not found during the first selection steps.
5) **Final Decision:** After adding the new studies that were collected from the snowballing processes. We filtered our set of studies again to end up with the final set of studies of interest.

The selection process was supported by quality assessment criteria with inclusion and exclusion rules as follows:

1) **Inclusion Criteria:**
   a) Studies written in English
   b) Studies discussing automated video game verification, testing, or playtesting
2) **Exclusion Criteria:**
   a) Studies talking about topics not related to game testing, such as procedural level generation agents
   b) Studies that are duplicates of other studies

D. Data Extraction

In order to have solid extracted data and to easily manage the extraction process, we established a well-structured comparison framework to help in finding answers to the research questions. Hence, for each research question there are certain attributes that the comparison framework studies. These attributes are categorized based on the research questions, and they are defined as follows:

- **RQ1:**
  - **Approach:** This attribute focuses on the methods and approaches used in the proposed solutions in the related studies. An example of such approaches could be based on Deep learning, Reinforcement Learning, Genetic Algorithms, and several other approaches that are used in the literature.
  - **Benefits:** This attribute focuses on the benefits and the outcomes of each approach studied in the literature.
  - **Limitations:** This attribute presents the limitations and drawbacks (if found) of the studied approach.
- **RQ2:**
  - **Testing Goals:** This attribute presents the goals of running automated tests on games, where there might be different goals for testing, such as, verifying the game's functionalities, balancing the game, enhancing the level design, etc.
- **RQ3:**
  - **Targeted Games:** This attribute studies the games that literature studies used and applied their approaches on. There are different types of games under variety of categories of genres [29].
  - **General Applicability:** This attribute studies the applicability of the approaches to other games or software systems other than the ones they have been evaluated with.
  - **Open Source:** This attribute focuses on the availability of the developed approach's code publicly as open-source project.

The analysis of the extracted information from the studies were discussed and visually illustrated in Section IV based on the research questions and their related comparison attributes .

IV. Results and Discussion

In this section we present our findings and primary observations after studying and analyzing the found studies, and by answering the research questions.

A. RQ1 - What automated game testing approaches are applied in the literature?

Studies in the literature varied in the ways of implementing automated game testing approaches. To distinguish between the studied approaches, we categorized the studies based on the implemented algorithms and their common characteristics. These categories are search-based, goal-directed, human-like, scenario-based, and model-based. We will go into the details of the approaches in each category, in addition to discussing the benefits and limitations of each one of them.

1) Search-Based Approaches

Search-based approaches are those focusing on exploring and analyzing the state space of a game, with the goal of finding and reporting the availability of states that match or break predefined criteria. Several algorithms were used in the literature to apply search-based testing, where some studies utilized Evolutionary Algorithms such as Genetic

Algorithms [16, 30, 31, 32], while other studies used graph based algorithms such as Monte Carlo Search [33, 34], and A* [22, 35]. On the other hand, some studies employed Rapidly Exploring Random Tree Search [36, 37, 38].

- *Evolutionary algorithms:* [16, 30, 31, 32, 39]

  Chan et al. (2004) [16] presented a Genetic Algorithms based approach to verify, find, and reach game states that designers did not expect to be existing while designing the game. In another study [30], Salge et al. (2008) developed a Genetic Algorithms agent that helped in verifying the game with its ability of detecting bugs and gameplay flaws. Furthermore, Tan et al. (2013) [32] used Genetic Algorithms with Artificial Neural Networks to understand the state space and to test the game levels. In another study, Garcia Sanchez et al. (2018) [31] approached automated playtesting using AI agent based on Evolutionary Algorithms and a new mutation operator. The developed approach assisted game designers to balance the game undertest. Zheng et al. (2019) [39] developed a testing agent called Wuji which uses Evolutionary Algorithms and multi-objective optimization to explore game space.

  Despite the effectiveness of using Evolutionary Algorithms to search and explore game state spaces, the approach had some limitations, where it might be time consuming when it comes to identifying the best fitness function for the agent [16]. Moreover, Chan et al. [16] stated that using Evolutionary Algorithms depends on off-line learning which could sometimes fail to cover all the sequences of the game. Thus, to overcome these problems, Salge et al. [30] used three AI paradigms which are Swarm AI, Councillor AI and Reactive AI to make the approach more useful and be able to reach more states. The authors of [39] utilized Reinforcement Learning to direct the agent while exploring the state space. Moreover, in [32], Tan et al. used Artificial Neural Networks to learn from the explored states.

- *Graph search:* [33, 34, 22, 35, 40, 41, 42]

  Shaker et al. (2013) [41] presented an approach that is based on Depth-First state space search using game simulations. The authors stated that they preferred using Depth-First search because they were interested in checking the playability of the levels instead of investigating all the solution space, where playability is a quality attribute [43] based on usability in the context of video games, gameplay or player interaction [44][45], with an objective of providing enjoyment and entertainment through credibility and satisfaction [46]. To enable sensible search procedure, the authors restricted the search by encoding core game rules' components into a Prolog-based agent, using a set of reachable components to cover context information, and by defining a policy to get possible actions. One of the limitations mentioned by the authors is that the agent might accept generated levels that are unplayable, or sometimes accept generated playable levels that could contain some unnecessary or unused components.

  In other studies, A* search algorithm was used to test games. Silva et al. (2018) [22] used A* based approach to find misbehaving events. Hoyt et al. (2019) [35] proposed two A* based approaches that can be used to assess the playability of game levels. The first is an A* Reachability Check agent, which determines the possibility of going from any two random points in a level [35]. The second is an A* based Survival Analysis agent, which gives an approximation about navigation difficulty in levels [35]. Although the A* approaches seem to be effective for searching through a state space, Silva et al. [22] stated that developing such approach comes with a price which is the need of tuning the heuristic function to fit the situation being tested.

  On the other hand, Monte Carlo search algorithms were used by several studies. Keehl and Smith (2018) [33] presented game testing automation in Unity engine using a framework that is based on Monte Carlo Tree Search (MCTS). The framework consisted of four main parts: First, Jupyter notebook is used for running the experiments and visualizing the results. Second, a module that contains implementation of MCTS which is called python support. Third, a C# based module that is mandatory to any game project that uses this framework, which communicates with the python modules through a TCP protocol socket. Fourth, modifications to the game where it must determine legal moves at each step, request an index of a move to take, and apply actions. In another study [34], Zook et al. (2015) presented the utilization of MCTS to perform planning strategies to simulate player behavior with different skills, where MCTS helped in understanding the space of strategic options of player skills. Isaksen et al. (2015) [40] presented the use of survival analysis and Monte Carlo Simulation to develop an AI agent that allows the exploration of a subset of game variants with parameters changes. MCTS based agents bring several benefits. Isaksen et al. [40] showed that using Monte Carlo can help in enhancing the gameplay quality and the game's playability by modifying game's parameters such as number of obstacles, sizes and dimensions of game objects and scoring criteria. Moreover, Zook et al. [34] stated that the analysis of MCTS based agents could help in understanding and balancing gameplay and levels. Despite the various implementations and use cases of Monte Carlo based approaches, they still have some limitations as pointed out by several authors [33, 34, 40]. One of these limitations is that a Monte Carlo based agent can only look in single dimension of parameters. In addition to that, another limitation is related to the limited representation of the game state space [33] [34]. To avoid such limitations, Keehl and Smith [33] suggested using reinforcement learning algorithms to extract knowledge gained during MCTS rollouts. Moreover, Isaksen et al. [40] recommended looking into higher dimensions of parameters when using Monte Carlo based approaches to get better exploration results.

In another side, Machado et al. (2018) [42] presented Cicero, an AI-assisted game design and debugging tool. The AI agents in this tool can playtest different types of games which are built on top of GVGAI. The agents of the tool use graph search algorithms such as, breadth-first, depth-first search, A* and, MCTS. Moreover, they can explore the state space of the games they play using predefined heuristics to find the best actions to take. However, the limitation of this tool is its subjectivity to GVGAI, where the authors stated that the tool is nowhere near having general-purpose game-playing agents that can work in other game engines or development environments. Furthermore, they mentioned that the tool still has open questions related to the lack of fast simulations.

- *Rapidly exploring random tree search:* [36, 37, 38]

  Zhan et al. (2018) [36] proposed a search approach based on Rapidly Exploring Random Tree (RRT), which works by growing a tree and capturing the reachability of points in the game's state space from some initial state in the game, where the algorithm takes advantage of its ability of exploring continuous feature space. Moreover, the approach has the ability of understanding the relations between explored states. To allow progression during the exploration process, random goals are picked, and the algorithm tries to reach them by finding closest nodes to the goal and by picking and performing the best actions.

  In another study, Chang et al. (2019) [37] presented an approach based on RRT called Reveal-More. The approach combines automatic exploration with few minutes of human gameplay, which results in a better game state coverage. The authors [37] stated that using this approach would lower the testers' efforts in testing and finding all paths within a game.

  Tremblay et al. (2014) [38] proposed a game testing approach that abstracts game state space into a high dimensional geometric space to support pathfinding. This approach used RRT to support design decisions by integrating it with Unity engine.

  From the studied approaches, we can see that RRT has a great potential in exploring continuous state spaces. However, Chang et al. [37] stated that RRT has a limitation of not picking the most useful actions. Thus, more studies that investigate the usefulness of the selected actions is required.

2) Goal-Directed Approaches

Goal-directed approaches aim at injecting the goals of an agent to its implementation through defining policies, rewards, and penalties, which guide the agent to explore the potential paths that lead to the desired goals and objectives. The found approaches varied in their implementation. Some studies used Reinforcement Learning [47, 48, 49, 50], while others used different algorithms with a defined policy that forces the search to be directed to a certain goal [51, 52, 53].

- *Reinforcement learning:* [47, 48, 49, 50]

  Pfau et al. (2017) [47] developed a system for intelligent completion of adventure riddles via Unsupervised Solving (ICARUS) which is based on discrete Reinforcement Learning in a dualistic fashion, with short-term and long-term memory, and built for Visionaire game engine. Moreover, the approach utilizes specialized heuristics that reduce the search space, in addition to the ability of using pre-defined situation-dependent action choices to support the agent's playthroughs. ICARUS is used to autonomously play, test, and report bugs in games.

  Napolitano (2020) [48] proposed using Deep Reinforcement Learning with a Dueling Deep Q-Network strategy, to gain efficient and performant results that allowed the agent to extract essential information from the game environment and take decisions on the next moves to support game balancing and design.

  Joakim et al. (2020) [49] used Deep Reinforcement Learning, to create a self-learning agent that can explore and exploit the game mechanics based on a user-defined reinforcing reward signal. The authors stated that using Reinforcement Learning agents is better suited to complement the testing environment.

  Shin et al. (2020) [50] presented applying Reinforcement Learning via strategic play learning. Moreover, the employed policy-based learning method that was used in the study is called actor-centric, which provides probabilistically strategic actions befitting randomly changing states via learning policies suitable to the state. This approach can be used to verify game design and level balancing.

  There were some limitations when applying reinforcement learning. Joakim et al. [49] mentioned that not all problems are better solved using Reinforcement Learning, where using more focused agents is better. Moreover, Pfau et al. [47] stated that according to their approach of implementing Reinforcement Learning, one of the limitations is the subjectivity towards a certain genre of games and a specific development environment. Shin et al. [50] mentioned that training such agents might be time consuming, and to overcome this limitation predefined strategic play is recommended.

- *Restricted heuristics:* [51, 52, 53]

  De Mesentier Silva et al. (2017) [51] presented two agents with specialized heuristics, where the first agent approaches the game in a conservative manor, whereas the second plays aggressively and proactively. The authors have shown that their approaches were cheaper in terms of computational time and they outperformed A* and MCTS agents. In [52] (which is a continuation work of [50]) the authors developed other agents that were able to discover and identify faulty states in the game. Moreover, the authors stated that the new agents would help in saving cost and time in the early stages of development. The agents presented [51, 52] can be used for game analysis as stated by the authors.

  Jaffe et al. (2012) [53] developed a restricted-play framework, which is capable of measuring the game balance. Though, one of the limitations mentioned by the

authors [53] is that such approaches can only be utilized to test discrete games. In addition, agents based on restricted heuristics are subjective to the game being tested [51, 52].

3) Human-Like Approaches

Human-like approaches are focused on imitating the human behavior, where agents are optimized to produce results similar to those obtained from humans. Such approaches are very useful when testing human centric features, such as emotions, curiosity, challenges, difficulty, aggressiveness and more. There were several studies found that implemented human-like agents, either through utilizing human data and learning algorithms, or by using specialized and restricted heuristics, or mixing between different algorithms.

- *Machine learning:* [23, 54, 55, 20, 56, 57]

  A semi-automated gameplay analysis by Machine Learning using active learning approach was presented by Southey et al. (2005) [23, 54]. The approach uses player data to train the agent, and it was used to find flaws in the gameplay.

  Zook et al. (2014) [55] used an Active Learning approach with regression model that utilizes four acquisition functions for regression models which are variance, probability of improvement, expected improvement, and upper-confidence bounds. It was used to fix game controls.

  Gudmundsson et al. (2018) [20] used Deep Learning approach based on Convolutional Neural Networks to analyze game screens and to make decisions based on that. The authors stated in the study that in order to train human-like agents, training data was required. Thus, they collected those data from 1% of randomly selected players from the game in a duration of 2 weeks. The obtained dataset was nearly $1.2 * 10^7$ samples. Then the dataset was split into 3 subsets used per level: training set with 4500 samples, validation set with 500 samples and test set with 500 samples. Their agent was used to check the playability of newly added levels. The approach was able to play and assess the design of the game, and it was giving better results than an MCTS-based agent.

  Borovikov et al. (2019) [56] presented two case studies of game playtesting using Machine Learning algorithms. In the first case study, the authors developed an agent that is composed of three main components which are, an Ensemble of Multi-resolution Markov models that capture the style of the teacher from the main game features perspective, a Deep Neural Networks component which was trained as a supervised model on samples bootstrapped from an agent playing the game following the Markov ensemble, and an interactivity component between the game designer and the agent where the game designer can take the controller from the agent allowing the agent to learn from the samples generated while the game designer is taking control. In the second case study, they utilized Reinforcement Learning to allow the agent to play in different styles like offensive or defensive. This approach can help in reducing the effort of developing such agents that act as game AI.

  Pfau et al. (2020) [57] applied an approach based on Deep Player Behavior Modeling (DPBM), that can be used to automate game balancing. Decisions and player models are created by DPBM through mapping preference distribution of actions to game states via machine learning and state-action architecture. The approach allows considering many playing styles, instead of reducing the decision-making strategies.

  Despite the various applications of Machine Learning in human-like agents, some limitations were mentioned in the literature. The availability of player data is the main and most common limitation, where the amount of player data could affect the performance of the agent [54, 55, 56, 57]. Though, Borovikov et al. [56] stated that interactive sessions could solve such limitation. On the other hand, the space complexity of the game could affect the performance and the accuracy of the agent [54, 55], which can be mitigated by specifying and training agents based on data that is relevant to a specific situation [55].

- *Restricted heuristics:* [58, 59, 60, 61, 62]

  Devlin et al. (2016) [58] used MCTS based agent that imitates human behavior by using a biased policy that uses Bradley-Terry value and UCT scoring formula, in addition to the utilization of human data. The authors stated that the performance of the agent was competitive and efficient for discrete action games. The approach can be used to analyze and study game balancing.

  Ariyurek et al. (2019) [59] proposed a human-like based tester and a synthetic based agent which uses MCTS. The human-like agent uses human-like test goals that were extracted and trained from human tester data. On the other hand, the synthetic based agents were based on synthetic test goals that were created from game scenarios and represented using a graph-based approach where the agent rewarded all valid and some invalid transitions. The authors applied these test goals on agents based on state-action-reward-state-action and MCTS, and generated test sequences and validated the game behavior according to the game constraints automatically. The agent's decision making procedure was based on user parameters that were supplied to the agent. The agent proposed in this study plays the goals sequentially through their feature vector. The sequence can then be checked for evaluation using a criteria threshold. Furthermore, inverse reinforcement learning was used in the study to capture and learn experiences and to automatically generate tests using human testers' expertise. Moreover, the authors proposed a multiple greedy-policy inverse reinforcement learning to overcome the problem of complex human tester actions that are difficult to model. The study stated that it was easier to build an agent that targets a simple goal to play rather than a complex one. Moreover, it was better to verify one goal at a time when playing different levels, where skipping a feature might happen because of the level composition and execution order of the test steps that are important for the agents. The proposed approach

has some limitations, first, the approach is based on a greedy solution, which can be improved with dynamic programming [59]. However, according to the authors, using dynamic programming will increase the test goal creation time [59]. Another limitation is that the multiple greedy-policy inverse reinforcement learning agent generalizes the exploited sequences to all situations, which can cause problems when learning the behavior of the tester. Moreover, the agent may not fulfill a goal due to two reasons, the infeasibility of the goal, and a prevention caused by a bug. Thus, the authors stated that they allowed the agent to play the game for a specific number of steps. When the last step occurs, if the goal was not reached, then it may be unreasonable to let the agent target the next goal [59]. In 2020, Ariyurek et al. [60] proposed several modifications to their human-like agent presented in [59] to enhance bug finding. The proposed modifications were applied to the MCTS policy, where they tested different strategies such as using transpositions, tree reuse, knowledge-based evaluations, Boltzmann rollouts, MixMax, and Single-Player-MCTS.

In another study, Mugrai et al. (2019) [61] proposed an agent based on MCTS and an evolving utility function to create different procedural personas and human playstyles, which allowed creating automated playtesting system. The player style developed in the study imitates a long-term human player who follows a strategy of optimizing number of points by maximizing them via a series of actions and after a specific number of moves. This approach can be used for game balancing.

Stahlke et al. (2019) [62] developed a framework with expert systems and artificial intelligent agents that can perform simulated testing sessions. The goal of the agent is to behave like human players when navigating the game world by mimicking the player's tendency to explore, wander, and becoming lost. The framework is built on top of Unity game engine. Moreover, the framework is able to log and record the agent's exploration behavior for game design analysis. The authors stated that the framework could be limited because of the nature of expert systems. Though, using their approach does not require training data.

Most of the discussed approaches that apply restricted strategies face difficulties in generalizing their solutions to other games [62].

- *Mixed algorithms:* [63, 64, 65]

MCTS along with Stratabots were used by Horn et al. (2018) [63] to improve state exploration and enhance the search speed of Stratabots, and to imitate human behavior while playing a game, in addition to studying the effectiveness of the game difficulty. Stratabots in the study were crafted to allow the agents to understand the scores and take and undo actions. Moreover, the authors mentioned that their Stratabots take greedy approaches to select actions that maximize the score. MCTS was introduced in the study to improve the creation of Stratabots and minimize the hand-crafted features, where performance can be considered a Stratabots' limitation.

Holmgård et al. (2018) [64] presented the use of procedural personas which are archetypal player models that are non-player characters provided with human-like personalities to automatically playtest game content. The developed procedural personas were based on MCTS and Evolutionary Computation algorithms to test and improve the game design and balance game's difficulty. The limitations of this study are that the developed personas were inherently subjective where a utility function should be constructed by the game or the level designer to test the content of the game. The authors suggested using a technique that allows learning utility functions from demonstration such as using inverse reinforcement learning methods.

In another aspect, Keehl and Smith (2019) [65] proposed an extension to their previous work [33] to allow using MCTS with Machine Learning to imitate human behavior and to improve the agent's exploration. Using imitation learning in this approach allowed the agent to summarize a collection of gameplay samples into a reactive decision policy which could help game designers. One of the limitations mentioned is that in order for the agent to work effectively, it has to be testing deterministic games only.

4) Scenario-Based Approaches

We define scenario-based approaches as techniques and frameworks that run tests which are based on predefined human made sequences of actions, or human requested actions. Some approaches were found to record human sequences of actions and replay them [66, 67]. Other investigated studies performed sequences of actions automatically through game simulations [68, 69, 70]. While some others used semi-automatic approaches that work with the help of a human tester commanding the tool to debug the visuals of the game via predefined criteria using image processing [71, 72, 73].

- *Record and Replay:* [66, 67]

Ostrowski and Samir (2013) [66] proposed a model that creates and executes regression tests within video games, where the approach utilizes record and playback mechanisms. The authors stated that the record and playback technique is easy to use and it requires minimal programming skills, which makes it easily used by game testers to create meaningful tests. The approach works by recording events' steps which are compared later to a playback of the game to check for bugs and errors. Another benefit is that it can be generally applicable to any type of games as mentioned in the study, where they applied the approach to different genres of games. Though, generalization might not be feasible due to integration issues.

In another study, Bécares et al. (2017) [67] presented a theoretical framework for beta testing games. The proposed approach provides two testing methods. The first is based on recording game sessions as input commands from keyboard or mouse that capture playing

through the whole game, and then reapply these commands to test the game again automatically. The second approach is based on recording the internal messages that are defined as high level actions and the timeframe context of these actions when the game is fully played. In addition to that, they proposed using petri nets to assess and support the testing process in this method, especially when the levels are updated and the input traces are not accurate. This approach supports compatibility and regression testing. However, one of the limitations of this approach is that there are some cases where this approach cannot work properly when levels are updated and playthrough recordings are not modified. On the other hand, the approach has AI behavior to allow harmonization between the traced messages (high level actions) and the created petri nets, to prevent agents getting stuck during replays.

- *Game simulation:* [68, 69, 70]

  Jung et al. (2004) [68] purposed a Virtual Environment Network User Simulator (VENUS) system to perform automated beta tests. This approach supports game testers and reduces development resources. The authors stated that VENUS system supports generality and scalability, where VENUS virtual client engine can easily simulate any kind of online games. Moreover, the users of the system can make large number of simulations to test the game server's capacity and to perform stress tests. One of the benefits mentioned in the study is the ability of storing internal game data from the game server to the database for analysis purposes. However, there are some limitations in this approach. One of them is that the system must be coupled with the game code to allow simulations and testing. This limits the generality of the approach. Thus, to apply the system on different games, major modifications need to be introduced to match with the targeted game for testing.

  Because of the limitations in VENUS [68], Cho et al. (2010) [69, 70] presented a newer version of the system called VENUS II that supports black box testing, to separate the tool from the game's code. The new tool supports black box testing by passing the game's grammar and virtual maps to the tool using Game Description Language format, to create a way of simulating the game through defined scenarios that virtual agents can perform automatically.

- *Visual debugging:* [71, 72, 73]

  Nantes et al. (2008) [71] proposed a semi-automatic framework that applies Computer Vision technologies to support the testing team, which helps in improving and accelerating the testing process. The approach uses a combination of two algorithms, Harris corner as descriptor and Canny edge as detector. This combination allows identifying jagged edges and visual distortions within the game environment.

  In another study, Mozgovoy and Evgeny (2017) [72] proposed a semi-automated smoke tests framework. The goal of smoke tests is to perform system checkups. Automated user interface smoke tests should be able to access the system under test as any user and perform actions. However, the authors stated that when it comes to testing graphical elements and hand drawn user interfaces, some artistic changes might be implemented such as animations and changes of sprites and positions, which are difficult to interact with and test using traditional smoke testing frameworks. As a result, the authors proposed a framework that utilizes image processing and recognition algorithms with UI automation framework called Appium. The authors used OpenCV library and integrated it with Appium tests to recognize game objects and hand drawn UI elements in Unity game engine. The approach can perform test scenarios and use image recognition to analyze screen content. The presented approach can be used to check visual failures in the game under test.

  In a similar way, Tuovenen et al. (2019) [73] presented an approach called MAuto, which focuses on testing mobile video games. The tool records user-interactions and exports them for playback in Appium. MAuto integrates image recognition by using AKAZE features to recognize objects in the taken screenshots. When MAuto's users perform recording, the tool creates test scripts to allow reproducing the recorded events. The application recognizes objects and UI controls which are stored as images during the recordings, and then it compares them to ground truth objects and images to identify errors.

  Computer Vision is useful when making decisions about objects and game visuals, where this evaluation process can be thought of as game environment inspection. However, such agents might not be able to capture the psychological part of the game, meaning entertainment inspection might not be feasible [71]. Moreover, image recognition-based testing agents might fail in recognizing errors [72] [73]. In addition, the image recognition process and the pixel comparison algorithms could be time consuming [72] [73]. Another limitation is that testing dynamic objects through recognition is hard to apply [73].

5) Model-Based Approaches

These approaches are based on abstracting the game's workflow into formal representations and models, which allow verification of flow of events, data, logic, or control. Several approaches were found in the literature that used different types of modeling techniques. Some studies used petri nets [74, 75]. While some others used Unified Modeling Language (UML) [76, 77].

- *Petri nets:* [74, 75]

  Yessad et al. (2014) [74] presented a formal framework that is based on petri nets to assist designers in modeling and automatically verifying games at design stage and before the implementation starts. The authors chose colored petri nets, which is one of the petri nets types [78], because these models give better specification by using colors to model data. Moreover, they used a

specific class of colored petri nets called symmetric nets with bags. The approach was integrated with Temporal Logic language to compare states and to verify errors using model checking and counter example mechanisms.

Reuter et al. (2015) [75] adopted using petri nets to support the automated testing process of video games. The authors presented the use of colored petri nets to find and detect structural errors in a scene-based game. The colored petri nets models were automatically generated from the game engine that the approach was integrated with. The proposed approach was used to verify the game through applying reachability analysis, and it was able to detect livelocks and deadlocks. The limitation of this approach is that it depends on scene-based games and the engine it was integrated with.

- *Unified Modeling Language:* [76, 77]

  Schaefer et al. (2013) [76] presented an automated testing framework called Crushinator that reduces the dependency upon beta testing. This framework was developed to provide a game-independent testing tool to verify event-driven client-server based game applications via automating multiple testing methods such as model based testing, load and performance testing, and exploratory testing. The model based procedure is applied by using behavior models of the system being tested, where a UML package within the tool is employed to extract UML state machine models from the game for testing purposes. The extracted models represent the behavior of the system under test, and the extracted paths from the models can be used to generate test cases. The authors stated that the tool can perform a complete coverage testing compared to other beta testing tools. Where coverage testing is defined as the process of identifying parts of the software that have not been exercised during testing, which guides the testing of important parts of the software and gives a clear checklist of test completeness [79]. The framework is easy to integrate with other games due to the nature of isolation and game independency. However, one of the limitations mentioned in the study is that due to the focus on event-driven game servers, the framework might not be applicable nor beneficial for other systems and games.

  Iftikhar et al. (2015) [77] presented a model based testing approach using UML diagrams. Their approach automates three main steps of software testing which are test case generation, test oracle generation, and test case execution. The approach works by simulating the inputs required for playing games. The authors stated that the proposed approach was able to find and detect major faults. However, there were some limitations, where the authors stated that the users must have software engineering background to be able to understand the models. Moreover, the models must be created for each game before using the approach. The authors stated that this approach might not be able to cover all paths and variations of functionalities.

- *Game Description Language:* [80]

  Haufe et al. (2012) [80] introduced a formal language to describe game-specific knowledge as state sequence invariants by following the semantics of Game Description Language. The proposed language uses Prolog-like inference mechanism to represent rules that allow players to make legal moves. Moreover, the approach allows automated verification of systems by checking state sequence invariants using Temporal Game Description Language extension, which supports checking local properties of games that can be proved by induction rather than by complete exhaustive search.

- *Linear Temporal Logic:* [81]

  Varvaressos et al. (2014) [81] presented a framework that can perform runtime monitoring of games, which can greatly speed up the testing phase through detecting bugs automatically while games are being played. The framework uses logical specification language known as first order linear temporal logic LTL-FO+, where the usage of this specification language would make it possible to write and check safety and temporal properties in games.

- *ModelMMORPG:* [82, 83]

  Schatten et al. (2017) [82, 83] used a game logic-oriented approach that allows testing game quests and objectives, in addition to supporting load and stability testing. The tool depends on modeling the game logic using ModelMMORPG modeling language that allows modelling a wide range of various large-scale multi-agent systems scenarios. The model in the tool is translated into concrete implementation of agent classes facilitated through an agent-based platform called (Smart Python Agent Development Environment). A variation of game related tasks like walking around, fighting etc., are allowed to be performed by agents by establishing relationships between agents and low-level implementation of the game under test.

In this study we reviewed and analyzed 51 studies and their proposed game testing approaches. The following figure (see Figure 1) presents the trends in the literature, where the figure depicts the number and category of each approach per year.

B. RQ2 - What are the objectives of the automated game testing approaches available in the literature?

From our analysis, we categorize the testing objectives of the studied game testing approaches as follows:

1) Testing for functional correctness

The goal of this testing objective is to ensure that the game is behaving as expected. To do so, games could be checked through verifying their functionalities, software code, the control flow of events, or the flow of data. This objective was implemented by several approaches, search-based approaches could check the availability of faulty states [16, 30, 22, 36, 39], goal-directed approaches [47, 49, 52] and human-like approaches [54, 59, 60, 63] could be used in

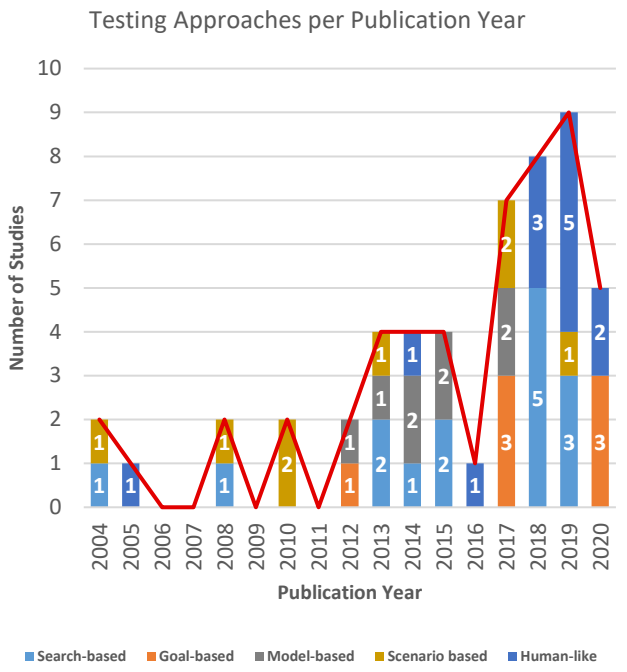

Figure 1. Analysis of Testing Approaches per Year

triggering invalid states and executing incorrect events, scenario-based approaches would give some insights on broken software code [66, 67, 69, 70, 73], and model-based approaches can proof-check the reachability of deadlocks, livelocks, or invalid states [75, 77, 80, 81, 82].

2) Testing for multiplayer stability

This testing objective aims at multiplayer games and testing the networking stability and capacity. The approaches that implemented this objective were using massive number of agents to check the network performance, stability, and the ability of handling huge capacity of active players sending commands and events through the network. This objective was targeted by some of the approaches, like, scenario-based approaches through running multiple simulation agents at the same time [69, 70], and model-based approaches by checking concurrent events and mimicking the flow of data and events in multiplayer games [82, 83].

3) Testing for performance

Testing games' performance is the goal of implementing this testing objective. This testing objective was applied by one study only, which implemented a goal-directed approach. Pfau et al. [47] stated that their agent can record performance metrics such as (FPS, RAM, CPU usage) at certain time, and these metrics can be continuously tracked over the game iteration life time resulting into the ability of recognizing performance problems and their causes.

4) Testing for visual correctness

The goal of this testing objective is to verify games' visuals, such as shaders, game UI, 3D models and animations, etc. [6]. Scenario-based studies that focused on visual debugging tackled this objective by implementing image recognition algorithms and analyzing pixels differences [71, 72, 73].

5) Testing for game design correctness

This testing objective is meant to test various aspects that affect the user experience, and which are related to the gameplay and its rules. The correctness of the created game environment and the placement of its objects is one of these aspects. Incorrectly placed game objects might affect the user experience and the playability of a game, where some bugs could be introduced because of that, such as stuck spots, and world holes [6]. Another aspect is imprecise and inappropriate game rules and constraints that are part of the game designers' job to create. Thus, testing such aspects is crucial to maintain a better user experience. This objective was targeted by several approaches, search-based approaches could help in exploring states that violate game rules [30, 22, 35, 39, 40, 41, 42], goal-directed approaches [47, 49, 50, 51] and human-like [20, 57, 64, 65] could help in getting some insights related to agents and players getting lost, stuck within game levels or getting distracted by other elements in the game rather than the game's objective, and scenario-based approaches that help in getting insights and analysis on how changes of level would affect the game winning scenarios [67].

6) Testing for game balance and fairness

This testing objective focuses on verifying the fairness of the game and the balance of game's parameters. Search-based approaches could check the availability of states that affected choosing certain paths repeatedly in the decision graphs because of game parameters [31, 32, 33, 34, 39, 42], goal-directed approaches could help in checking the variations of actions taken and their effects in winning or losing games [48, 50, 51, 52, 53], and human-like approaches could help in checking game fairness by checking the winning conditions or checking game parameters that motivated agents' to take similar actions because of their advantage [54, 55, 20, 56, 57, 58, 61, 64, 65].

7) Testing for progression and learnability

The goal of this objective is to verify that the player is going to be able to learn the game, progress through levels, and complete the game. Some approaches implemented this objective in their agents, for instance, search-based [37] and goal-directed [48] approaches could help in checking the ability of finishing a game. Moreover, human-like approaches could help in studying and analyzing players' psychological behaviors in learning and discovering game areas [56, 61, 62].

8) Testing for physical correctness

This objective aims at verifying the state of physical properties in the game world. These properties could be related to collisions, frictions, gravity, etc. Only one study was found to target this testing objective, where a goal-directed approach by Joakim et al. [49] utilized Reinforcement Learning to check collision properties and the ability of going through walls or getting stuck in defined stuck points.

The following figure (see Figure 2) shows analysis of the testing approaches and the associated testing objectives.

C. *RQ3 - How do researchers validate their developed automated game testing approaches?*

Several games were used to validate the testing approaches. The results and analysis depicted in Figure 3 show that Tile-matching games were the most used to test human-like approaches. Moreover, search-based approaches were tested on more variations compared to other approaches. Nevertheless, most of the approaches presented were not generally applicable, where most of the studies applied their testing approach to one genre. However, Zheng et al. [39] argued that the implementation of their search-based approach could be generally applicable to different types and genres of games. Moreover, to support the validation of some of the approaches, some authors shared the source code of their approaches publicly to allow other researchers to improve, extend, and learn from their work. Keehl et al. open sourced both of their search-based [33] and human-like [65] approaches Monster-Carlo [84] and Monster-Carlo 2 [85]. Furthermore, Schatten et al. [82, 83] publicly shared their implementation of their model-based approach [86]. Also, Varvaressos et al. [81] open sourced their model-based solution [87].

D. *RQ4-What are the shortcomings in the current state of the affairs?*

After studying and analyzing the found playtesting techniques, this question is answered with the analysis outcomes discussed in Section IV, and the implications of the study in Section V. (Table I. summarizes our findings).

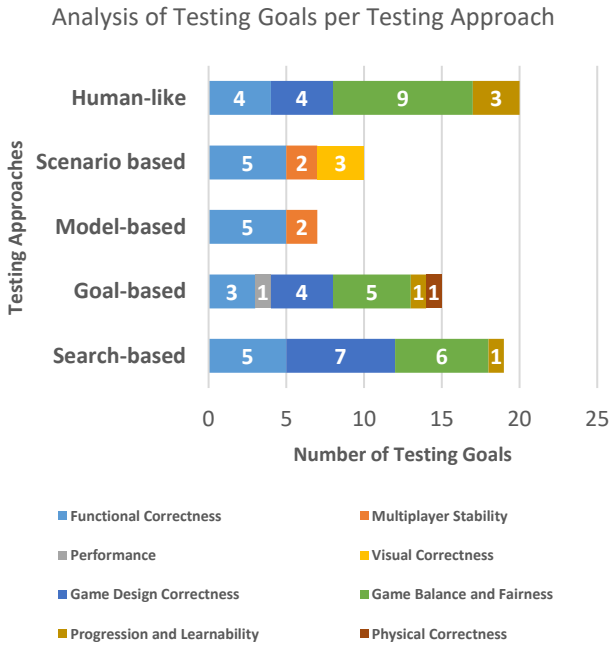

Figure 2. Analysis of Testing Approaches and Testing Objectives

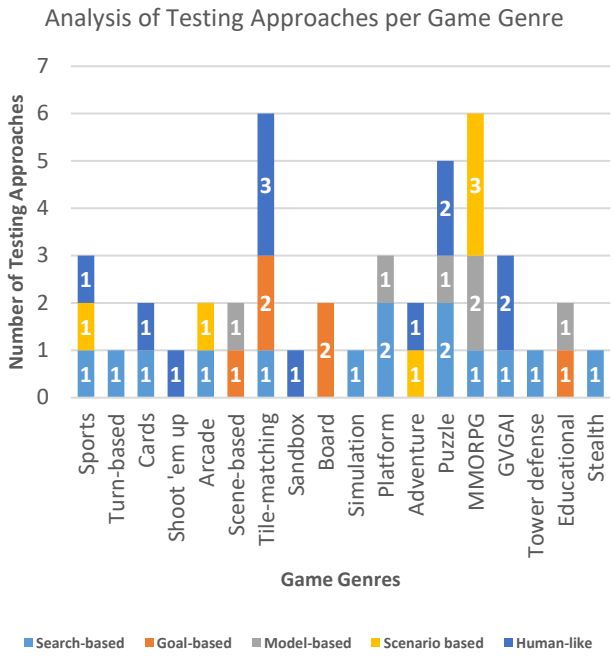

Figure 3. Analysis of Tested Game Genres

V. Implications of the Study

The results and observations in Section IV reveal that there are still gaps and open issues that have not been addressed in the literature. We discuss such gaps and open issues, and suggest future work, in the sequel.

A. *Game testing approaches and testing objectives*

- We presented automated game testing categories in Section IV based on the studied approaches. However, more categories could be added. For example, collaborative approaches were not discussed in the literature. Thus, it might be a new area of research to show how multi-agents with their stand-alone behaviors can interact and collaborate to test a game. Collaborative testing would support verifying different testing goals, such as, multiplayer stability, performance, functional correctness, game design correctness, etc.
- Another interesting field that requires more research is using computer vision and image processing methods to check and verify testing goals other than visual correctness.
- Model-based approaches were used to mainly test functional correctness of games. However, they could also be used to check game design and game balancing issues, which are missing in the literature.
- We observe that model-based approaches in the literature were mostly applied without looking at the software, meaning, the verification checks were directly applied to models without any communication with the code and the game environment. Thus, we suggest that a future work could consider

implementing testing techniques for creating models from code to show flow of events similar to that in the code. Moreover, future research might also consider connecting models with agents that are interacting with the game environment, to observe more information and to allow different types of testing goals.

- Several implementations of the scenario-based approaches were lacking the adaptation to environmental changes in the game, those implementations were either based on recorded playthroughs or known sequences of actions. Thus, we recommend that future work in this field shall consider researching techniques that can inform and adapt predefined scenarios to work with the newly added environmental changes.
- Another area that could be researched in the scenario-based approaches is the development of counter-example scenarios and checking whether these scenarios can be reached or not. This approach would help in checking the correctness of both game's functionality and design.
- All the studied testing approaches were used to show faults or to give insights related to the game. However, none of them were used to verify and recommend changes. Thus, we observe that new research direction could be implementing a recommendation system that can check the game design and recommend changes to enhance game experiences.
- Building playtesting agents that use automated online learning and parameter tuning techniques that are based on self-learning and self-playing without the need of prior human knowledge or previously available data samples. This would help in solving the limitations related to requiring training data, parameters tuning, and interactive sessions applied to the agents.
- Human-like agents were proven to be working by several of the recent studies. These approaches were used to test different objectives. However, using these human-like agents with the injected human data to analyze psychological aspects such as enjoyment, excitement, feeling fear and being bored, were not yet studied.
- Several testing objectives were found to be targeted by the literature approaches. However, none of the studies aimed at audio correctness testing objective.
- Studies that implemented visual debugging approaches focused on static elements on screen. Hence, we observe that more studies are needed for checking dynamic visual elements such as visual effects, animations, and particle systems.
- Most of the search-based approaches were exploring game states to check invalid states. However, none of the studies looked at deadlocks, livelocks, or conflicting and overlapping valid states where two or more valid states are occurring at the same time, but they should not be.
- The literature lacks automated testing approaches that verify procedurally generated content in games.
- With the increasing interest in virtual reality and augmented reality games [88] [89], we observe that one of the research areas that needs to be explored is to automatically test games developed with such technologies.

B. *Game testing general applicability*

- Building a general game testing agent that is not biased towards a certain game genre or game development environment.
- From Figure 3, we can see that some game genres were tested by some approaches and not by others. Moreover, some game genres were not tested, such as fighting games or first-person shooting games. Thus, more applications of automated game testing to different game genres is suggested.
- Most of the tested game genres had small state space complexity, which might not be challenging. It would be more beneficial if future research focuses on testing games with practical complexity and state space.
- Due to the lack of publicly available relevant artifacts, we suggested that open source projects/data should be encouraged in future works to support studies and advancements in the field.

VI. THREATS TO VALIDITY

As any survey study, the validity of the findings of this research faces some threats. We cannot ensure that we have studied all available approaches; there might be other approaches either in the literature or in the industry that we were not able to reach. We mitigated this by considering studies from most prominent literature databases specifically (Google Scholar, IEEE, ACM, Springer, Science Direct) using the same search string. Moreover, to ensure that we collected adequate studies, we utilized both forward and backward snowballing.

In this study we categorized automated game testing approaches studied from the literature, in addition to categorizing game testing objectives. However, the categorization and classification procedure was a creative, research and development process. Thus, the own subjective nature of creativity imposes an external validity risk.

The observations concluded from our study were based on using our developed comparison framework and its attribute list presented in Section III-D. These attributes were used to compare between found approaches in literature and to extract answers to the research questions. The comparison framework and its attribute list were developed to cover the important characteristics of automated game testing approaches. However, additional attributes may be introduced by other researchers to further study and analyze automated game testing approaches.

Approach	Implementations	Testing Objectives	Game Genres

Search-Based	• **Evolutionary Algorithms:** [16, 30, 31, 32, 39]
	• **Graph search:**
	  • Depth first: [41]
	  • A*: [22, 35]
	  • MCTS: [33, 34, 40]
	  • Cicero: [42]
	• **RRT:** [36, 37, 38]
		• Functional correctness
		• Game design correctness
		• Game balance and fairness
		• Progression and learnability
			Sports, Turn-based, Cards, Arcade, Simulation, Tile-matching, MMORPG, Puzzle, Platform, Tower defense, Stealth, GVGAI

Goal-Directed	• **Reinforcement learning:** [47, 48, 49, 50] • **Restricted heuristics:** [51, 52, 53]	• Functional correctness • Game design correctness • Game balance and fairness • Progression and learnability • Physics correctness • Performance	Scene-based, Board, Tile-matching, Educational

Human-Like	• **Machine learning:** [54, 55, 20, 56, 57]
	• **Restricted heuristics:** [58, 59, 60, 61, 62]
	• **Mixed algorithms:**
	  • MCTS and Stratabots: [63]
	  • MCTS and Evolutionary Algorithms: [64]
	  • MCTS and Machine Learning: [65]
		• Functional correctness
		• Game design correctness
		• Game balance and fairness
		• Progression and learnability
			Sports, Shoot 'em up, Tile-matching, Sandbox, Puzzle, GVGAI, Cards, Adventure

Scenario-Based	• **Record and Replay:** [66, 67] • **Game simulation:** [68, 69, 70] • **Visual debugging:** [71, 72, 73]	• Functional correctness • Game design correctness • Visual correctness • Multiplayer stability	Adventure, MMORPG, Sports, Arcade
Model-Based	• **Petri nets:** [74, 75] • **Unified Modeling Language:** [76, 77] • **Game Description Language:** [80] • **Linear Temporal Logic:** [81] • **ModelMMORPG:** [82, 83]	• Functional correctness • Multiplayer stability	Educational, Platform, Scene-based, MMORPG, Puzzle

TABLE I
SUMMARY OF THE STUDY FINDINGS

VII. CONCLUSION

In this study we investigated automated game testing approaches in the literature. We developed a framework based on a set of attributes identified as a result of an extensive survey of existing approaches. Accordingly, we answered our research questions using our assessment framework discussed. We analyzed approaches available in the literature against our framework. We classified and compared the approaches accordingly.

Our findings have shown that there is still a gap and future work required in this field. Moreover, we believe that automated game testing can improve the game development life cycle by finding errors, enhancing the gameplay and reporting game analysis in less time and effort supporting game designers and developers and improving their productivity.

ACKNOWLEDGMENT

The authors wish to acknowledge King Fahd University of Petroleum and Minerals (KFUPM) for providing the facilities to carry out this research. Many thanks are due to the anonymous referees for their detailed and helpful comments.

Aghyad Albaghajati M.S student in Software Engineering from King Fahd University of Petroleum and Minerals, Saudi Arabia. He earned his B.S in Software Engineering from King Saud University, Saudi Arabia, in 2016. His research interests are Software Testing, Software Construction, Software Architecture and Design and Game Development.

Moataz Ahmed received his PhD in computer science from George Mason University in 1997. Dr. Ahmed is currently a faculty member with the Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Saudi Arabia. He also severs as Adjunct/Guest Professor in a number of universities in the US and Italy. During his career, he worked as a software architect in several software houses. His research interest includes artificial intelligence and machine learning; and automated software engineering, especially, artificial intelligence based software testing, software reuse, and cost estimation. He has supervised a number of theses and published a number of scientific papers in refereed journals and conferences in these areas.

REFERENCES

[1] Blow, J., 2004. Game Development: Harder Than You Think, Queue, vol. 1, nro. 10, ss. 28–37, February 2004.

[2] Kanode, C. M. and Haddad, H. M., 2009. Software Engineering Challenges in Game Development, in Information Technology: New Generations (ITNG '09). Sixth International Conference on, ss. 260 – 265.

[3] E. Aarseth, "A narrative theory of games," in Proceedings of the international conference on the foundations of digital Games. ACM, 2012, pp. 129–133.

[4] Redavid, Claudio, and Adil Farid. "An overview of game testing techniques." Västerås: sn (2011).

[5] "Gamesindustry.Biz presents...the year in numbers 2018." [Online]. Available: https://www.gamesindustry.biz/articles/2018-12-17-gamesindustry-biz-presents-the-year-in-numbers-2018

[6] L. Levy and J. Novak, "Game Development Essentials: Game QA & Testing," Cengage Learning, 2009, pp.58-70.

[7] C. P. Schultz and R. D. Bryant, "Game Testing: All in One," Mercury Learning & Information, 2016, pp.125-128.

[8] V. R. Basili and R. W. Selby, "Comparing the effectiveness of software testing strategies," IEEE transactions on software engineering, no. 12, pp. 1278–1296, 1987.

[9] Klaib, Mohammad FJ. "A parallel tree based strategy for 3-way interaction testing." Procedia Computer Science 65 (2015): 377-384.

[10] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," IEEE transactions on software engineering, vol. 30, no. 6, pp. 418–421, 2004.

[11] Redavid, Claudio, and Adil Farid. "An overview of game testing techniques." Västerås: sn (2011).

[12] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams," Proceedings of The Future of Software Engineering at ICSE 2007, pp. 85-103, 2007.

[13] Starr, K. Testing Video Games Can't Possibly Be Harder Than an Afternoon With Xbox, Right? Seattle Weekly (July 2007).

[14] Lewis, Chris, and Jim Whitehead. "The whats and the whys of games and software engineering." Proceedings of the 1st international workshop on games and software engineering. 2011.

[15] J. P. Davis, K. Steury, and R. Pagulayan, "A survey method for assessing perceptions of a game: The consumer playtest in game design," Game Studies, vol. 5, no. 1, pp. 1–13, 2005.

[16] B. Chan, J. Denzinger, D. Gates, K. Loose and J. Buchanan, "Evolutionary behavior testing of commercial computer games," Proceedings of the 2004 Congress on Evolutionary Computation, Jun. 2004.

[17] J. Ortega, N. Shaker, J. Togelius and G. N. Yannakakis, "Imitating human playing styles in Super Mario Bros," Entertainment Computing, vol. 4, pp. 93-104, Apr. 2013.

[18] A. Khalifa, A. Isaksen, J. Togelius, and A. Nealen, "Modifying mcts for human-like general video game playing." in IJCAI, 2016, pp. 2514–2520.

[19] Yarwood, Jack. "A Look at How Different-Sized Studios Approach the Challenges of QA." Sep 2020. [Online]. Available: www.gamasutra.com/view/news/359240/A_look_at_how_differentsized_studios_approach_the_challenges_of_QA.php.

[20] Gudmundsson, Stefan Freyr, et al. "Human-like playtesting with deep learning." 2018 IEEE Conference on Computational Intelligence and Games (CIG). IEEE, 2018.

[21] Thompson, Tommy. "The Secret AI Testers inside Tom Clancy's The Division." Sep 2020. [Online]. Available: www.gamasutra.com/blogs/TommyThompson/20200304/359028/The_Secret_AI_Testers_inside_Tom_Clancys_The_Division.php.

[22] De Mesentier Silva, Fernando, Borovikov, Igor, Kolen, John, Aghdaie, Navid, AND Zaman, Kazi. "Exploring Gameplay With AI Agents" AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (2018).

[23] Southey, Finnegan, et al. "Semi-Automated Gameplay Analysis by Machine Learning." AIIDE. 2005.

[24] Roohi, Shaghayegh, et al. "Review of intrinsic motivation in simulation-based game testing." Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. 2018.

[25] Zarembo, Imants. "Analysis of Artificial Intelligence Applications for Automated Testing of Video Games." Proceedings of the 12th International Scientific and Practical Conference. Volume II. Vol. 170. 2019.

[26] B. Kitchenham, "Procedures for performing systematic reviews," Keele, UK, Keele University, vol. 33, no. 2004, pp. 1–26, 2004.

[27] M. O. Riedl and A. Zook, "Ai for game production," in 2013 IEEE Conference on Computational Inteligence in Games (CIG). IEEE, 2013, pp. 1–8.

[28] Wohlin, Claes. "Guidelines for snowballing in systematic literature studies and a replication in software engineering." Proceedings of the 18th international conference on evaluation and assessment in software engineering. 2014.

[29] "Background the origins of game genres." [Online]. Available: https://www.gamasutra.com/view/feature/132463/

[30] C. Salge, C. Lipski, T. Mahlmann, and B. Mathiak, "Using genetically optimized artificial intelligence to improve gameplaying fun for strategical games," in Proceedings of the 2008 ACM SIGGRAPH symposium on Video games. ACM, 2008, pp. 7–14.

[31] P. Garcia Sanchez, A. Tonda, A. M. Mora, G. Squillero, and J. J. Merelo, "Automated playtesting in collectible card games using evolutionary algorithms: A case study in hearthstone," Knowledge-Based Systems, vol. 153, pp. 133–146, 2018.

[32] Tan, Tse Guan, et al. "Automated Evaluation for AI Controllers in Tower Defense Game Using Genetic Algorithm." International Multi-Conference on Artificial Intelligence Technology. Springer, Berlin, Heidelberg, 2013.

[33] O. Keehl and A. M. Smith, "Monster carlo: An mcts-based framework for machine playtesting unity games," in 2018 IEEE Conference on Computational Intelligence and Games (CIG), Aug 2018.

[34] Zook, Alexander, Brent Harrison, and Mark O. Riedl. "Monte-Carlo Tree Search for Simulation-based Strategy Analysis."

[35] Hoyt, Andrew, et al. "Integrating Automated Play in Level Co-Creation." arXiv preprint arXiv:1911.09219 (2019).

[36] Zhan, Zeping, Batu Aytemiz, and Adam M. Smith. "Taking the scenic route: Automatic exploration for videogames." arXiv preprint arXiv:1812.03125 (2018).

[37] Chang, Kenneth, Batu Aytemiz, and Adam M. Smith. "Reveal-more: Amplifying human effort in quality assurance testing using automated exploration." 2019 IEEE Conference on Games (CoG). IEEE, 2019.

[38] Tremblay, Jonathan, Pedro Andrade Torres, and Clark Verbrugge. "An algorithmic approach to analyzing combat and stealth games." 2014 IEEE Conference on Computational Intelligence and Games. IEEE, 2014.

[39] 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.

[40] Isaksen, Aaron, Daniel Gopstein, and Andrew Nealen. "Exploring Game Space Using Survival Analysis." FDG. 2015.

[41] Shaker, Noor, Mohammad Shaker, and Julian Togelius. "Evolving playable content for cut the rope through a simulation-based approach." Ninth Artificial Intelligence and Interactive Digital Entertainment Conference. 2013.

[42] Machado, Tiago, et al. "Ai-assisted game debugging with cicero." 2018 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2018.

[43] J. L. G. S´anchez, F. M. Simarro, N. P. Zea, and F. L. G. Vela, "Playability as extension of quality in use in video games." in I-USED, 2009.

[44] L. Nacke, A. Drachen, K. Kuikkaniemi, J. Niesenhaus, H. J. Korhonen, W. M. Hoogen, K. Poels, W. A. IJsselsteijn, and Y. A. De Kort, "Playability and player experience research," in Proceedings of digra 2009: Breaking new ground: Innovation in games, play, practice and theory. DiGRA, 2009.

[45] J. K¨ucklich and M. C. Fellow, "Play and playability as key concepts in new media studies," STeM Centre, Dublin City University, 2004.

[46] J. L. G. S´anchez, N. P. Zea, and F. L. Guti´errez, "From usability to playability: Introduction to player-centred video game development process," in International Conference on Human Centered Design. Springer, 2009, pp. 65–74.

[47] Pfau, Johannes, Jan David Smeddinck, and Rainer Malaka. "Automated game testing with icarus: Intelligent completion of adventure riddles via unsupervised solving." Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play. 2017.

[48] Napolitano, Nicholas. "Testing match-3 video games with Deep Reinforcement Learning." arXiv preprint arXiv:2007.01137 (2020).

[49] Joakim Bergdahl, Camilo Gordillo, Konrad Tollmar, Linus Gisslén, Augmenting Automated Game Testing with Deep Reinforcement Learning, IEEE Conference on Games (CoG), 2020
[50] Shin, Yuchul, et al. "Playtesting in Match 3 Game Using Strategic Plays via Reinforcement Learning." IEEE Access 8 (2020): 51593-51600.
[51] F. de Mesentier Silva, S. Lee, J. Togelius, and A. Nealen, "Ai as evaluator: Search driven playtesting of modern board games," 2017.
[52] de Mesentier Silva, Fernando, et al. "AI-based playtesting of contemporary board games." Proceedings of the 12th International Conference on the Foundations of Digital Games. 2017.
[53] Jaffe, Alexander, et al. "Evaluating competitive game balance with restricted play." Eighth Artificial Intelligence and Interactive Digital Entertainment Conference. 2012.
[54] Southey, Finnegan, et al. "Machine learning for semi-automated gameplay analysis." Proceedings of the 2005 Game Developers Conference (GDC. 2005).
[55] Zook, Alexander, Eric Fruchter, and Mark O. Riedl. "Automatic Playtesting for Game Parameter Tuning via Active Learning."
[56] Borovikov, Igor, et al. "Winning isn't everything: Training agents to playtest modern games." AAAI Workshop on Reinforcement Learning in Games. 2019.
[57] Pfau, Johannes, et al. "Dungeons & Replicants: Automated Game Balancing via Deep Player Behavior Modeling."
[58] Devlin, Sam, et al. "Combining Gameplay Data with Monte Carlo Tree Search to Emulate Human Play." AIIDE. 2016.
[59] Ariyurek, Sinan, Aysu Betin-Can, and Elif Surer. "Automated Video Game Testing Using Synthetic and Human-Like Agents." IEEE Transactions on Games (2019).
[60] Ariyurek, Sinan, Aysu Betin-Can, and Elif Surer. "Enhancing the Monte Carlo Tree Search Algorithm for Video Game Testing." arXiv preprint arXiv:2003.07813 (2020).
[61] Mugrai, Luvneesh, et al. "Automated playtesting of matching tile games." 2019 IEEE Conference on Games (CoG). IEEE, 2019.
[62] Stahlke, Samantha, Atiya Nova, and Pejman Mirza-Babaei. "Artificial Playfulness: A Tool for Automated Agent-Based Playtesting." Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems. 2019.
[63] Horn, Britton, et al. "A Monte Carlo approach to skill-based automated playtesting." Proceedings. AAAI Artificial Intelligence and Interactive Digital Entertainment Conference. Vol. 2018. NIH Public Access, 2018.
[64] Holmgård, Christoffer, et al. "Automated playtesting with procedural personas through MCTS with evolved heuristics." arXiv preprint arXiv:1802.06881 (2018).
[65] Keehl, Oleksandra, and Adam M. Smith. "Monster Carlo 2: Integrating Learning and Tree Search for Machine Playtesting." 2019 IEEE Conference on Games (CoG). IEEE, 2019.
[66] Ostrowski, Michail, and Samir Aroudj. "Automated Regression Testing within Video Game Development." GSTF Journal on Computing 3.2 (2013).
[67] Bécares, Jennifer Hernández, Luis Costero Valero, and Pedro Pablo Gómez Martín. "An approach to automated videogame beta testing." Entertainment Computing 18 (2017): 79-92.
[68] Jung, YungWoo, et al. "VENUS: The online game simulator using massively virtual clients." Asian Simulation Conference. Springer, Berlin, Heidelberg, 2004.
[69] Cho, Chang-Sik, et al. "Online game testing using scenario-based control of massive virtual users." 2010 The 12th International Conference on Advanced Communication Technology (ICACT). Vol. 2. IEEE, 2010.
[70] Cho, Chang-Sik, et al. "Scenario-based approach for blackbox load testing of online game servers." 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. IEEE, 2010.
[71] Nantes, Alfredo, Ross Brown, and Frederic Maire. "A Framework for the Semi-Automatic Testing of Video Games." AIIDE. 2008.
[72] Mozgovoy, Maxim, and Evgeny Pyshkin. "Unity application testing automation with appium and image recognition." International Conference on Tools and Methods for Program Analysis. Springer, Cham, 2017.
[73] Tuovenen, J., Mourad Oussalah, and Panos Kostakos. "MAuto: Automatic Mobile Game Testing Tool Using Image-Matching Based Approach." The Computer Games Journal 8.3-4 (2019): 215-239.
[74] Yessad, Amel, et al. "Have You Found the Error? A Formal Framework for Learning Game Verification." European Conference on Technology Enhanced Learning. Springer, Cham, 2014.
[75] Reuter, Christian, Stefan Göbel, and Ralf Steinmetz. "Detecting structural errors in scene-based Multiplayer Games using automatically generated Petri Nets." Foundations of Digital Games, Pacific Grove, USA (2015).
[76] Schaefer, Christopher, Hyunsook Do, and Brian M. Slator. "Crushinator: A framework towards game-independent testing." 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2013.
[77] Iftikhar, Sidra, et al. "An automated model based testing approach for platform games." 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS). IEEE, 2015.
[78] Jensen, Kurt, and Lars M. Kristensen. Coloured Petri nets: modelling and validation of concurrent systems. Springer Science & Business Media, 2009.
[79] J. R. Horgan, S. London, and M. R. Lyu, "Achieving software quality with testing coverage measures," Computer, vol. 27, no. 9, pp. 60–69, 1994.
[80] Haufe, Sebastian, Stephan Schiffel, and Michael Thielscher. "Automated verification of state sequence invariants in general game playing." Artificial Intelligence 187 (2012): 1-30.
[81] Varvaressos, Simon, et al. "Automated Bug Finding in Video Games: A Case Study for Runtime Monitoring." Proceedings of the 2014 IEEE International Conference on Software Testing, Verification, and Validation. 2014.
[82] Schatten, Markus, et al. "Towards an agent-based automated testing environment for massively multi-player role playing games." 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2017.
[83] Schatten, Markus, et al. "Automated MMORPG Testing–An Agent-Based Approach." International conference on practical applications of agents and multi-agent systems. Springer, Cham, 2017.
[84] MonsterCarlo. Sep 2020. [Online]. Available: https://github.com/saya1984/Monster-Carlo
[85] MonsterCarlo2. Sep 2020. [Online]. Available: https://github.com/saya1984/MonsterCarlo2
[86] ModelMMORPG. Sep 2020. [Online]. Available: https://github.com/tomicic/ModelMMORPG
[87] BeepBeep. Sep 2020. [Online]. Available: https://github.com/kimlavoie/BeepBeepPingus
[88] Lee, D., et al. "A development of virtual reality game utilizing Kinect, Oculus Rift and smartphone." International Journal of Applied Engineering Research 11.2 (2016): 829-833.
[89] Koutromanos, George, and Lucy Avraamidou. "The use of mobile games in formal and informal learning environments: a review of the literature." Educational Media International 51.1 (2014): 49-65.