UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MOISÉS OLIVEIRA COSTA

TESTES DE SOFTWARE AUTOMATIZADOS EM JOGOS DIGITAIS: UM ESTUDO BIBLIOGRÁFICO

RUSSAS

2024

MOISÉS OLIVEIRA COSTA

TESTES DE SOFTWARE AUTOMATIZADOS EM JOGOS DIGITAIS: UM ESTUDO BIBLIOGRÁFICO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Markos Oliveira Freitas.

RUSSAS

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C874t Costa, Moisés Oliveira.
Testes de software automatizados em jogos digitais : Um estudo bibliográfico / Moisés Oliveira Costa. – 2024.
62 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2024.
Orientação: Prof. Dr. Markos Oliveira Freitas.

1. Testes de Software. 2. Jogos Digitais. 3. Automação de testes. I. Título.

CDD 005

MOISÉS OLIVEIRA COSTA

TESTES DE SOFTWARE AUTOMATIZADOS EM JOGOS DIGITAIS: UM ESTUDO BIBLIOGRÁFICO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Markos Oliveira Freitas (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Rafael Fernandes Ivo
Universidade Federal do Ceará (UFC)

Profa. Dra. Jacilane de Holanda Rabelo
Universidade Federal do Ceará (UFC)

Dedico este trabalho à minha família, por sua capacidade de acreditar em mim e investir em mim. Aos meus pais, seus cuidados e dedicações foi que me deram, em alguns momentos, a esperança para seguir. A todos que, direta ou indiretamente, fizeram parte deste trabalho.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois Ele me concede coragem para nunca desistir.

À minha família e amigos, seu suporte inabalável, expresso minha profunda gratidão pelo apoio incondicional, estímulo constante e palavras de encorajamento, fundamentais para transpor os desafios. Aos colegas de faculdade, agradeço pelas enriquecedoras trocas de conhecimento, valiosa amizade e momentos inesquecíveis compartilhados.

Aos professores, que compartilharam seu conhecimento e contribuíram para o meu crescimento acadêmico e profissional. Em particular, desejo expressar minha profunda gratidão ao meu orientador, Markos, pela sua orientação sábia e paciência incansável. Seu papel foi essencial para o desenvolvimento bem-sucedido deste trabalho de pesquisa e aprendizado.

Agradeço também a todas as pessoas que, de alguma maneira, me incentivaram e estiveram ao meu lado ao longo dessa trajetória. Suas expressões de incentivo foram essenciais para superar os desafios. A todos vocês, minha mais sincera gratidão. Este trabalho é o resultado do apoio de pessoas extraordinárias que confiaram em mim. Sem vocês, nada disso seria possível.

"O único lugar onde o sucesso vem antes do trabalho é no dicionário." (Albert Einstein)

RESUMO

Avaliar a qualidade e o desempenho de software em jogos digitais é fundamental para garantir a satisfação do usuário e o sucesso comercial. À medida que a complexidade dos jogos aumenta, a necessidade de identificar e corrigir *bugs* de forma eficiente torna-se ainda mais importante. Este estudo fornece uma revisão da literatura sobre técnicas de automação teste de software em jogos digitais, destacando as tecnologias mais recentes e tendências emergentes na área. O estudo analisou artigos relevantes cobrindo questões como métodos de teste e estratégias de automação. Os resultados mostram que, embora tenham sido alcançados progressos significativos, ainda existem desafios que precisam ser superados. Com base nessas descobertas, este estudo fornece informações valiosas para pesquisadores e profissionais que buscam melhorar as práticas de teste de software em jogos digitais.

Palavras-chave: testes de software; jogos digitais; automação de testes.

ABSTRACT

To evaluate the quality and performance of software in digital games is fundamental in order to ensure user satisfaction and commercial success. As the complexity of games increases, the need to efficiently identify and fix bugs becomes even more important. This study provides a review of the literature on automatic software testing techniques in digital games, highlighting the latest technologies and emerging trends in the field. This study analyzes relevant articles covering issues such as testing methods and automation strategies. The results show that although significant progress has been made, there are still challenges that need to be overcome. Based on these findings, this study provides valuable information for researchers and professionals looking to improve software testing practices in digital games.

Keywords: software testing practices; digital games; test automation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3D	Three-dimensional/Tridimensional
AAA	*Triple A*/Triplo A
CCPT	*Curiosity-conditioned proximal trajectories*/Curiosidade-condicionada trajetórias proximais
DNN	*Deep Neural Network*/Rede Neural Profunda
EFSM	*Extended Finite State Machine model*/Máquina de Estados Finito
EvoMBT	*Evolutionary Model Based Testing*/Testes Baseados em Modelos Evolutivos
GANs	*Generative Adversarial Networks*/Redes Adversárias Generativas
GVG-AI	*General Video Game AI*/IA geral de jogos digitais
IA	Inteligência artificial
IL	*Imitation Learning*/Aprendizagem por imitação
LSTM	*Long/Short-Term Memory*/Memória de Longo e Curto Prazo
MCTS	*Monte Carlo Tree Search*/Busca em Árvore de Monte Carlo
MGP-IRL	*Maximum causal entropy inverse reinforcement learning*/Aprendizado por reforço inverso de entropia causal máxima
ML	*Machine learning*/Aprendizado de máquina
ReLU	*Rectified linear unit*/Unidade linear retificada
RL	*Reinforcement learning*/Aprendizagem por reforço
UCB1	*Upper Confidence Bound 1*/Limite superior de confiança 1
UI	*User interface*/Interface do usuário

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

Recentemente, a indústria dos jogos digitais tornou-se significativa dentro da indústria de entretenimento, buscando atingir o mesmo nível de impacto e popularidade indústrias cinematográfica e musical (KONVOY VENTURES, 2024). Essa expansão pode ser observada na Figura 1, que destaca o crescimento do mercado ao longo do tempo e prevê que o mercado de jogos atinja US$ 189,3 bilhões em 2024, com um crescimento de 2,9% em relação ao ano anterior (KONVOY VENTURES, 2024). Os jogos digitais são mais do que apenas *hobbies*, são plataformas complexas que empregam física avançada, inteligência artificial, gráficos e áudio para criar mundos tridimensionais e experiências interativas (MCGONIGAL, 2011).

Figura 1 – Os jogos digitais representam um mercado de US$ 189 em 2024

Fonte: KONVOY VENTURES (2024)

Segundo Zyda (2016), a complexidade desses sistemas traz desafios únicos no desenvolvimento de softwares. Ao contrário do software tradicional, os jogos exigem uma otimização minuciosa para garantir fluidez e resposta em tempo real. Cada componente, desde a renderização gráfica até a lógica dos jogos, deve ser dimensionado proporcionalmente para funcionar sem sacrificar o desempenho. Além disso, a necessidade de compatibilidade entre diferentes plataformas e dispositivos aumenta a dificuldade de desenvolvimento.

Neste caso, o teste de software é fundamental para identificar erros, melhorar a jogabilidade e garantir que os jogadores desfrutem de uma experiência ininterrupta neste caso. Testar software em jogos digitais, por outro lado, não é uma tarefa fácil. Muitas dificuldades são distintas devido à complexidade das interações entre narrativa, mecânicas de jogo e qualidade de software. Então, o objetivo deste trabalho é fornecer uma análise abrangente da literatura existente sobre o assunto, identificar tendências emergentes e resolver os problemas associados à automatização de testes em jogos digitais.

1.2 Objetivos

1.2.1 Objetivo geral

Realizar uma pesquisa sobre testes automatizados em jogos digitais, focando na avaliação da qualidade e nos desafios do desenvolvimento de jogos.

1.2.2 Objetivos específicos

- Identificar as práticas mais recentes para automatizar testes em jogos digitais;
- Identificar as principais vantagens e desvantagens ao utilizar automatização de testes em jogos digitais;
- Determinar como a automatização influencia a eficácia e a eficiência de testes em jogos digitais.

1.3 Organização

O trabalho está organizado da seguinte forma. No Capítulo 2, é fornecida a fundamentação teórica necessária, abordando conceitos como testes de software, inteligência artificial e aprendizado de máquina, além de descrever um trabalho que se assemelha a esse estudo. No Capítulo 3, são levantadas as questões de pesquisa que guiam essa pesquisa e são fornecidos os passos metodológicos realizados na execução deste trabalho. No Capítulo 4, são descritos os trabalhos relacionados que ajudaram no desenvolvimento da pesquisa. No Capítulo 5, são detalhadas e respondidas as questões de pesquisa, com base nos trabalhos analisados. Por fim, no Capítulo 6, são apresentadas as conclusões acerca do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Introdução

Neste capítulo, serão fornecidos os fundamentos necessários para a compreensão dos principais conceitos abordados neste estudo: testes de software e inteligência artificial. Testes de software são essenciais para garantir a qualidade do produto, identificando e corrigindo *bugs* antes do lançamento. A inteligência artificial é o ramo da ciência da computação que busca criar máquinas que simulam a inteligência humana. Ao explorar os fundamentos teóricos relevantes para o tema em discussão, pretende-se estabelecer uma premissa conceitual que permita ao leitor não só compreender os métodos utilizados, mas também determinar a relevância e o impacto das análises. Além disso, é apresentado um estudo relacionado ao presente trabalho, que traz uma revisão sobre testes automatizados de software em jogos digitais, o que é relevante para esta pesquisa.

2.2 Testes de software

Os testes de software são uma parte essencial da engenharia de software, pois ajudam a garantir a qualidade e a confiabilidade dos sistemas e aplicativos (PRESSMAN, 2014). Essa prática envolve a avaliação sistemática de um sistema ou aplicativo com o objetivo de identificar defeitos, erros e falhas (MYERS, 2011).

2.2.1 Classificação dos testes de software

Os testes de software podem ser classificados em várias categorias, dependendo do objetivo e do escopo do teste. Segundo Pressman (2014), os testes de software podem ser classificados em:

- Testes unitários: avaliam o comportamento de unidades individuais de código, como funções ou classes. Esses testes são importantes para garantir que cada componente do software esteja funcionando corretamente antes de ser integrado com outros componentes (MYERS, 2011);
- Testes de integração: avaliam como os componentes do sistema interagem entre si, verificando se eles estão funcionando corretamente juntos. Esses testes são importantes para garantir que os componentes do software estejam funcionando corretamente juntos

(PRESSMAN, 2014);

- Testes de sistema: avaliam o comportamento do sistema como um todo. Esses testes são importantes para garantir que o sistema esteja funcionando corretamente e atendendo aos requisitos do usuário (SOMMERVILLE, 2015);
- Testes de aceitação: avaliam se o sistema atende aos requisitos do usuário (PRESSMAN, 2014);
- Testes de desempenho: avaliam o desempenho do sistema em termos de velocidade, escalabilidade e capacidade de resposta (PRESSMAN, 2014);
- Testes de segurança: avaliam a segurança do sistema em relação a vulnerabilidades e ameaças (MYERS, 2011);
- Testes de usabilidade: avaliam a facilidade de uso e a experiência do usuário com o sistema (NIELSEN, 2000);
- Testes de compatibilidade: avaliam a compatibilidade do sistema com diferentes plataformas, navegadores e dispositivos (PRESSMAN, 2014).

2.2.2 Estratégias de teste

As estratégias de teste definem a abordagem geral para a execução de testes. Segundo Pressman (2014), existem várias estratégias de teste, incluindo:

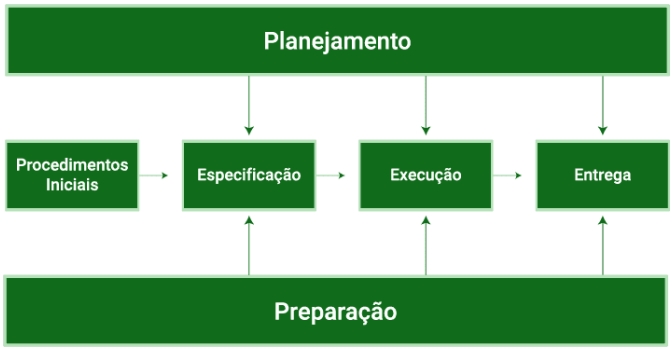
- Testes caixa-preta: avaliam o comportamento do sistema sem considerar seu funcionamento interno. Esses testes são importantes para garantir que o sistema esteja funcionando corretamente sem considerar os detalhes internos de implementação (BEIZER, 1990);
- Testes caixa-branca: avaliam o comportamento do sistema considerando seu funcionamento interno, incluindo a lógica de programação e a estrutura de dados. Esses testes são importantes para garantir que o sistema esteja funcionando corretamente e os componentes internos estejam trabalhando juntos corretamente (PRESSMAN, 2014);
- Testes caixa-cinza: uma combinação de testes caixa-preta e caixa-branca, que avaliam o comportamento do sistema considerando tanto o funcionamento interno quanto o externo (PRESSMAN, 2014);
- Testes caixa-verde: avaliam o comportamento do sistema em relação a requisitos específicos de desempenho, segurança ou usabilidade (MYERS, 2011).

2.2.3 Fases do ciclo de vida de teste

Os testes são conduzidos em diferentes fases do ciclo de vida de desenvolvimento de um software (Figura 2), desde a sua concepção até à sua manutenção. Cada fase tem seu próprio conjunto de atividades de teste e objetivos específicos (SOMMERVILLE, 2015). O ciclo de vida de desenvolvimento de software pode ser dividido em várias etapas, incluindo:

- Planejamento: definição dos objetivos e requisitos do projeto (PRESSMAN, 2014);
- Procedimentos iniciais: preparação do ambiente de desenvolvimento e definição das ferramentas e tecnologias a serem utilizadas (MCCONNELL, 2004);
- Especificação: definição detalhada dos requisitos e funcionalidades do software (ROBERTSON; ROBERTSON, 2013);
- Execução: desenvolvimento do software, incluindo a codificação e testes unitários (BECK, 2002);
- Entrega: entrega do software para os usuários finais Gilb (2005);
- Preparação: manutenção e atualização do software após a entrega (PIGOSKI, 1997).

Figura 2 – Fases do ciclo de testes.



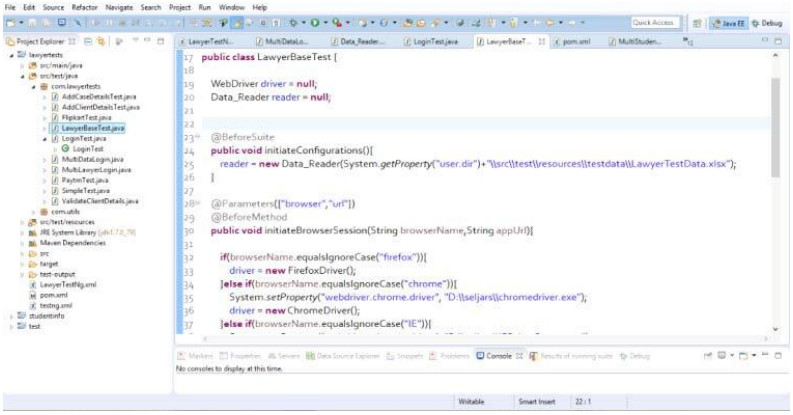
Fonte: Sofist (2023)

2.2.4 Ferramentas de teste

O uso de ferramentas de teste automatizadas é cada vez mais comum na indústria de software. Essas ferramentas ajudam a criar, executar e gerenciar casos de teste de maneira eficiente, acelerando o processo de teste (Kaner *et al.* (2002)). Por exemplo, o Selenium (Figura 3) é comumente empregado para realizar testes de interface de usuário (*User interface*/Interface do usuário (UI)) em aplicativos web, possibilitando a automatização de interações do usuário, como preenchimento de formulários e navegação. Isso resulta em economia de tempo e recursos

consideráveis durante o processo de teste.

Figura 3 – Exemplo de um caso de teste básico com o Selenium.



Fonte: Ramya *et al.* (2017)

2.3 Inteligência artificial e aprendizado de máquina

Desenvolver sistemas que imitem funções cognitivas humanas, como resolução de problemas e tomada de decisões, é um dos principais objetivos da Inteligência artificial (IA), que abrange diversas disciplinas (RUSSELL; NORVIG, 2011). O *Machine learning*/Aprendizado de máquina (ML) é um ramo da IA que se concentra em algoritmos que aprendem e melhoram a partir dos dados (MITCHELL, 1997).

2.3.1 Aprendizado de máquina

O ML é um ramo da inteligência artificial que automatiza o processo de aprendizagem e melhoria de desempenho de computadores sem ser explicitamente programado para isso. Em vez de seguir regras rígidas, os algoritmos de ML aprendem com os dados e a experiência que possuem e mudam seu comportamento para maximizar os resultados.

Existem três tipos de aprendizado de máquina(Figura 4):

- Aprendizagem supervisionada: os modelos de ML são treinados com um conjunto de dados previamente rotulados, onde cada dado possui uma etiqueta que define sua categoria, sua classificação, ou seu valor correto. O objetivo é que o modelo aprenda a mapear novos dados de entrada para as etiquetas correspondentes (RUSSELL; NORVIG, 2011);
- Aprendizagem não supervisionada: Ao contrário do aprendizado supervisionado, o aprendizado não supervisionado lida com dados não rotulados. Esses algoritmos podem ser usados

para identificar padrões e estruturas dentro dos próprios dados (RUSSELL; NORVIG, 2011);

– Aprendizagem por reforço: Esse tipo de aprendizado envolve um agente interagindo com seu ambiente. O agente recebe recompensas ou penalidades por suas ações e aprende a otimizar seu comportamento com base nessas recompensas (RUSSELL; NORVIG, 2011).

Figura 4 – Diagrama dos tipos de aprendizado em machine learning.

Fonte: Brasil (2020)

Outro conceito que ganhou força recentemente no campo do aprendizado de máquina é o das redes geradoras (GOODFELLOW *et al.*, 2014). As redes geradoras, como as *Generative Adversarial Networks*/Redes Adversárias Generativas (GANs), envolvem o uso de dois modelos: um gerador e um discriminador. O gerador cria dados falsos que se assemelham a dados reais, enquanto o discriminador tenta diferenciar entre dados reais e dados gerados. As GANs têm sido usadas em diversos campos, incluindo a criação de imagens e de vídeos sintéticos, o aprimoramento de imagens e até mesmo a geração de arte e de música, destacando seu potencial de inovação em aprendizado de máquina (GOODFELLOW *et al.*, 2014).

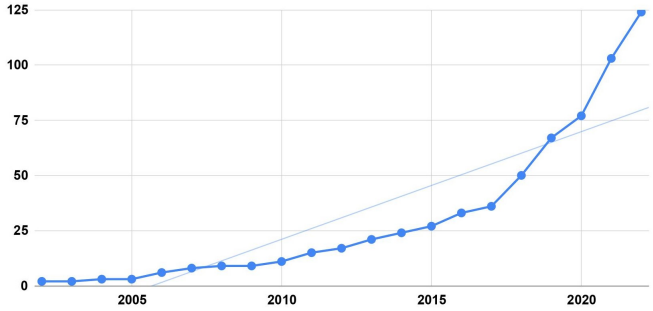
2.3.2 Aprendizado de máquina para geração de casos de teste em jogos digitais

Nos últimos anos, tem-se observado um aumento na utilização de aprendizado de máquina para gerar automaticamente casos de teste (Figura 5). Para jogos digitais, a fase de testes exige um grande número de casos de testes, e pode usufruir de vários benefícios dessa

automatização, como:

- Aumento da eficiência: O ML pode automatizar o processo de criação de casos de teste, liberando os testadores para se concentrarem em tarefas mais complexas (CHI; JONES, 2006);
- Melhoria da cobertura de código: O ML pode gerar casos de teste que cobrem mais código do que os casos de teste criados manualmente (HSIEH *et al.*, 2013);
- Detecção de falhas mais cedo: O ML pode gerar casos de teste que são mais propensos a encontrar falhas, o que permite que elas sejam detectadas e corrigidas mais cedo no ciclo de desenvolvimento (JUST *et al.*, 2014).

Figura 5 – Crescimento do uso de aprendizado de máquina na geração de testes desde 2002.



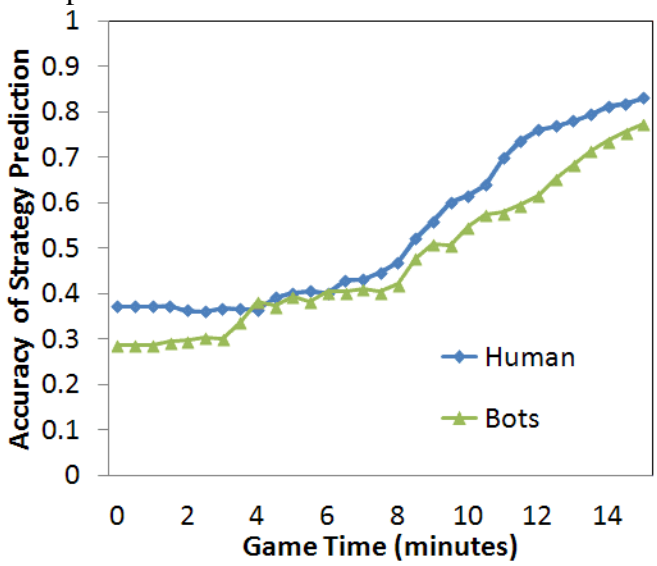
Fonte: Fontes e Gay (2023)

2.3.3 Bots com IA

A criação de casos de teste eficazes requer a simulação de diversos comportamentos dos jogadores. Os bots com IA podem enfrentar esse desafio, emulando o comportamento do jogador, podendo ser programados com diferentes níveis de habilidade, estratégias e algoritmos de tomada de decisão para imitar o comportamento de jogadores reais com diferentes níveis de experiência (MACHADO *et al.*, 2017). Isto ajuda a identificar desequilíbrios na jogabilidade, onde o jogo pode ser demasiadamente fácil ou difícil para grupos de jogadores específicos (Figura 6).

Além disso, em testes de estresse, ao implementar um grande número de bots atuando simultaneamente, a IA pode ser utilizada para testar a infraestrutura do jogo e identificar potenciais estrangulamentos ou limitações de recursos (MACHADO *et al.*, 2017).

Figura 6 – Comparação da precisão dos humanos e dos bots.



Fonte: Cho e Kim (2013)

2.4 Trabalho relacionado

Albaghajati e Ahmed (2023) analisam métodos para automatizar testes de jogos que incluem agentes baseados em IA para testar, equilibrar e melhorar a jogabilidade, bem como usar aprendizado de máquina e algoritmos A* para identificar *bugs*. Uma avaliação crítica e comparação destes métodos revela desafios que ainda precisam ser superados no campo da avaliação automatizada de jogos.

Os autores discutem em profundidade as várias abordagens para testes de jogos e identificam oito objetivos que devem ser alcançados durante os testes. Esses objetivos estão representados na Figura 7a. Albaghajati e Ahmed (2023) também descrevem estas abordagens como pertencentes a quatro categorias principais, através de um exame crítico da literatura existente. Essas classes estão listadas na Figura 7b.

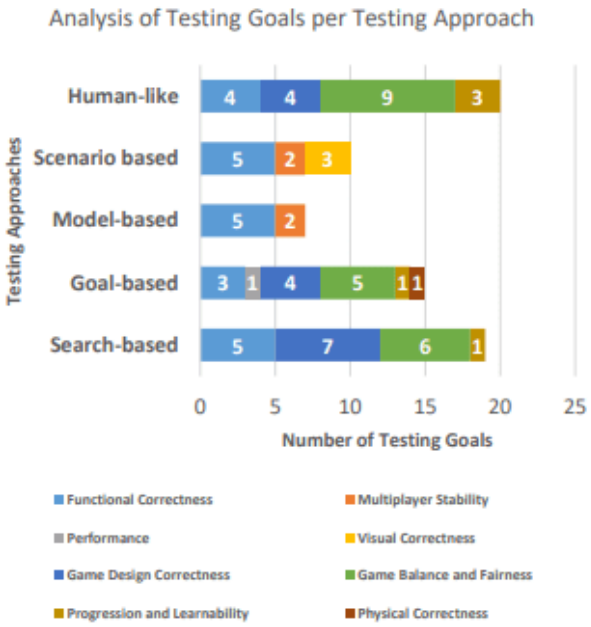
Os objetivos de testes automatizados identificados foram:

- Teste funcional: Garantir que o jogo execute suas funções sem erros;
- Teste de estabilidade multijogador: Verificar a estabilidade do jogo em ambientes multi-player;
- Teste de desempenho: Avaliar a capacidade do jogo de rodar em diferentes dispositivos sem problemas;
- Teste visual: Assegurar que os elementos visuais do jogo são exibidos corretamente;
- Teste do design do jogo: Testar se o jogo segue as regras de design estabelecidas;
- Teste de equilíbrio e equidade: Verificar se o jogo é equilibrado e justo para todos os

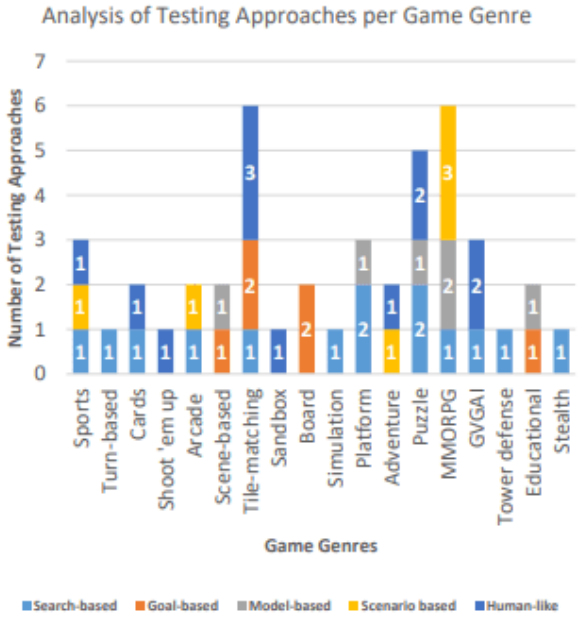
jogadores;

– Teste de progressão e capacidade de aprendizado: Testar se o jogo é fácil de aprender e progredir;
– Teste de física: Avaliar se o jogo segue as leis da física e comportamentos realistas;

Figura 7 – Abordagens para testes de jogos e objetivos:



(a) Análise das Abordagens de Teste por Abordagem de Teste.



(b) Análise das Abordagens de Teste por Gênero de Jogo.

Fonte: Albaghajati e Ahmed (2023)

As categorias de abordagens de testes automatizados descritos foram:

- Semelhante a um humano (*Human-like*): Agentes imitam o comportamento humano para testar aspectos centrados no jogador;
- Baseado em cenários (*Scenario-based*): Testes baseados em sequências de ações humanas pré-definidas ou simulações;
- Baseado em modelos (*Model-based*): Usam modelos para representar o comportamento do jogo e gerar testes;
- Baseado em objetivos (*Goal-based*): Concentram-se em atingir objetivos específicos dentro do jogo;
- Baseado em pesquisas (*Search-based*): Exploram o espaço de estados do jogo.

A investigação identificou múltiplas abordagens automatizadas para testar jogos, cada uma com suas próprias vantagens e desvantagens. No entanto, a pesquisa também revelou lacunas e áreas que deverão ser estudadas no futuro, incluindo a criação de um agente de teste de jogos que possa ser utilizado em qualquer tipo de jogo, a colaboração entre diferentes abordagens, a utilização de visão computacional para analisar componentes visuais que são dinâmicos e a consideração de jogos de realidade aumentada e virtual.

O trabalho concluiu que as análises automatizadas de jogos podem ter um impacto significativo na indústria de desenvolvimento de jogos, sentido na identificação de erros, na melhoria da jogabilidade e na otimização da experiência do jogador. Ao compreender as abordagens existentes e as áreas que são possíveis projetos de desenvolvimento, a investigação tenta direcionar pesquisas futuras na área de avaliação automática de jogos.

2.5 Considerações finais

Neste capítulo, foram apresentados os fundamentos teóricos necessários para o tema. Os conceitos de testes de software, inteligência artificial e aprendizado de máquina foram abordados, mostrando como eles se relacionam com o desenvolvimento e a avaliação de jogos de alta qualidade. Esses conceitos fornecem uma base conceitual que permite o leitor compreender os métodos utilizados nesta pesquisa, bem como determinar sua relevância. Ademais, foi explorado um trabalho relacionado ao tema testes automatizados de software em jogos digitais, com o objetivo de mapear a área e identificar as principais contribuições para o desenvolvimento da pesquisa. O estudo analisado aborda diferentes metodologias e perspectivas, fornecendo um panorama do tema.

3 METODOLOGIA

3.1 Introdução

Neste capítulo, serão apresentadas as etapas específicas do processo de revisão que foram realizadas durante a execução deste trabalho (Tabela 1), utilizando as orientações de (PINHEIRO, 2023) como referência para planejar, conduzir e escrever os resultados. O objetivo da metodologia é definir um conjunto de procedimentos essenciais para a revisão, que especificarão os métodos usados nesta pesquisa.

Tabela 1 – Tabela de seleção de artigos

Etapa	**Descrição**	**Resultado**
1	Definição das questões de pesquisa	Identificação de artigos relevantes
2	Definição das palavras-chave	Seleção de artigos relevantes
3	Busca em bases de artigos	10.273 artigos encontrados
4	Aplicação dos critérios automáticos de inclusão e exclusão	120 artigos selecionados
5	Triagem dos artigos e aplicação dos critérios não-automáticos de exclusão	5 artigos identificados
6	Análise das referências bibliográficas	7 artigos adicionais identificados
7	Inclusão de artigos adicionais	Conjunto final de 12 artigos selecionados

3.2 Questões de pesquisa

O objetivo deste trabalho é realizar uma revisão da literatura sobre testes de software em jogos digitais. Na Tabela 2, três questões chave foram identificadas durante o processo de pesquisa, visando aprofundar a compreensão e exploração deste tema.

A seleção dessas questões foi baseada no interesse em investigar o potencial das técnicas de testes automatizados para garantir a qualidade e consistência do jogo digital.

Tabela 2 – Questões de pesquisa.

Código	Pergunta
Q1	Quais as práticas mais recentes para automatizar testes em jogos digitais?
Q2	Quais são as vantagens e desvantagens das diferentes abordagens de testes automatizados de jogos?
Q3	Como a inteligência artificial pode melhorar a eficácia e a eficiência dos testes automatizados em jogos digitais?

Fonte: Elaborado pelo Autor (2024)

3.3 Palavras-chave

Para encontrar os trabalhos que satisfizessem aos objetivos desse trabalho, foi criada uma estratégia de busca com base nas palavras-chave do objetivo e das questões de pesquisa. Foram usadas as seguintes palavras-chave: *game testing*, *game performance testing*, *game quality assurance*, *video game testing*, *gameplay testing*, *game usability testing* e *game testing tools*. Essas palavras-chave foram escolhidas com base na especificidade, para evitar resultados irrelevantes, mas não tão específicas que excluam resultados relevantes. A combinação de palavras e de alguns sinônimos foi utilizada para criar uma busca mais eficaz, com o intuito de capturar resultados que usam termos diferentes.

3.4 Busca em bases de artigos

Para realizar este estudo sobre o tema Testes Automatizados de Software em Jogos Digitais, o repositório Google Acadêmico foi escolhido por ser um repositório abrangente e de fácil acesso, incluindo artigos de diversas origens, bem como o repositório Scopus, por apresentar ferramentas e filtros de pesquisa que aumentam a eficiência da pesquisa.

3.5 Aplicação dos critérios automáticos de inclusão e exclusão

Para obter estudos de alta qualidade para avaliação e reduzir o viés do pesquisador na fase de seleção, foram estabelecidos dois conjuntos de critérios distintos: critérios de inclusão identificados por CI (Tabela 3) e critérios de exclusão identificados por CE (Tabela 4).

Na busca inicial, foram encontrados 10.273 artigos. Após a aplicação dos critérios automáticos de inclusão e exclusão, 120 artigos foram selecionados para a revisão de literatura.

Tabela 3 – Critérios de inclusão.

Critério	Descrição
CI.1	Artigos que citem testes automatizados de software em jogos digitais
CI.2	Artigos que estejam escritos em inglês ou em português
CI.3	Artigos que tenham sido publicados entre 2018 e 2024
CI.4	Artigos publicados em periódicos ou anais de conferências

Fonte: Elaborado pelo Autor (2024)

Tabela 4 – Critérios de exclusão.

Critério	Descrição
CE.1	Artigos que não se encontram disponíveis para leitura ou download
CE.2	Artigos que não abordem testes de software classificados como testes de sistema ou testes de aceitação
CE.3	Artigos cujos autores não tenham associação ou filiação a empresas de desenvolvimento de jogos

Fonte: Elaborado pelo Autor (2024)

3.6 Triagem

Para auxiliar na priorização da leitura dos artigos selecionados, foram considerados os seguintes critérios de preferência:

- Relevância: A relevância do artigo para o tema da pesquisa e a contribuição direta para o desenvolvimento do estudo foram os principais fatores considerados.
- Novidade: A novidade das informações e descobertas apresentadas no artigo foi considerada, priorizando estudos recentes com contribuições inovadoras para a área de estudo.
- Impacto: A contribuição potencial do artigo para o desenvolvimento do estudo e a resposta as questões de pesquisa foi avaliada.

Foi feita uma análise das informações apresentadas nos resumos dos artigos, o que possibilitou uma extração de dados relevantes para a pesquisa e uma construção de uma base para o estudo.

3.7 Análise das referências

A partir da aplicação dos critérios de inclusão e exclusão, 120 artigos foram selecionados para uma análise mais aprofundada. No entanto, para garantir a qualidade e a profundidade da análise, foi necessário aplicar os critérios não automáticos de exclusão e os critérios de prioridade para selecionar apenas os artigos mais relevantes e que melhor atendessem aos objetivos da

pesquisa, resultando em 5 artigos selecionados para uma leitura.

3.8 Inclusão de artigos adicionais

Para melhorar a busca por artigos, também foram analisadas as referências bibliográficas de artigos já considerados relevantes para a pesquisa, utilizando uma abordagem de "*snowballing*"("bola de neve"), buscando identificar outros trabalhos que abordam o tema de forma similar ou complementar, mas que não tenham aparecido na busca inicial. Aqueles que atenderam aos critérios, cerca de 7 artigos, foram adicionados ao conjunto de referências, expandindo a base de dados da busca.

Assim, o total de artigos analisados foi de 12 (5 artigos inicialmente selecionados + 7 artigos adicionais identificados nas referências bibliográficas). A seleção criteriosa dos artigos garantiu que a análise fosse objetiva e relevante, ao mesmo tempo em que forneceu uma base sólida para os resultados deste estudo.

3.9 Considerações finais

Neste capítulo, foi descrita a metodologia adotada para realizar uma revisão da literatura sobre testes automatizados de software em jogos digitais. Essa abordagem permitiu responder às questões de pesquisa propostas, mas também a obter uma visão geral do "estado da arte".

4 ANÁLISE DA LITERATURA

4.1 Introdução

Neste capítulo, serão abordados, de forma detalhada, os artigos relacionados aos testes automatizados em jogos. Serão exploradas as diversas técnicas e estratégias utilizadas em estudos e práticas, para assegurar a qualidade e a confiabilidade dos videogames, destacando a importância desses procedimentos na indústria de desenvolvimento de jogos.

4.2 Artigos analisados

4.2.1 Automated Playtesting With Procedural Personas Through MCTS With Evolved Heuristics

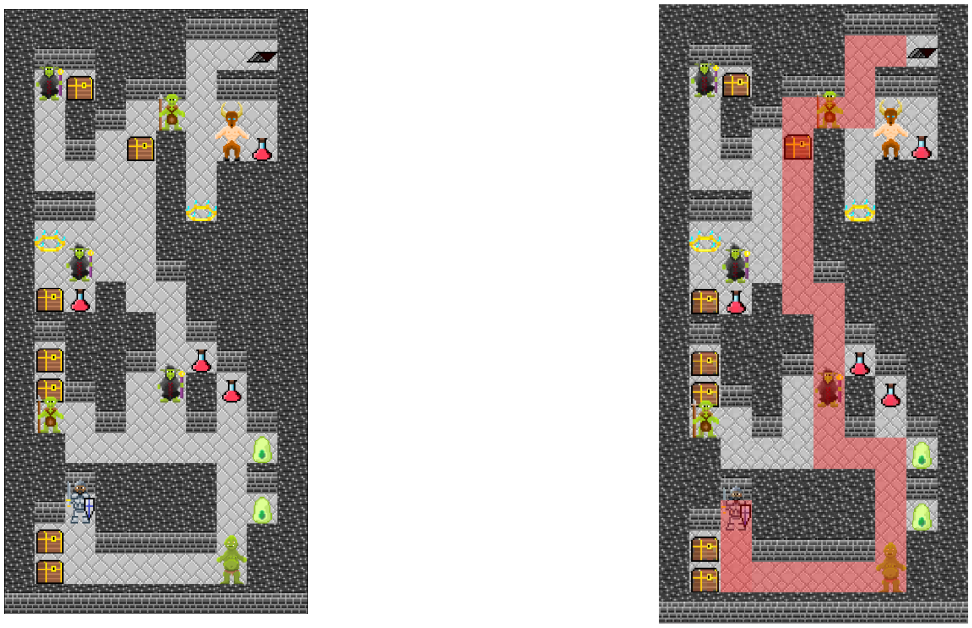
Holmgård *et al.* (2019) discutem uma abordagem inovadora para o teste automatizado de jogos digitais, com foco na criação e aplicação de personas procedurais, ou seja, modelos de jogadores artificiais criados com base em diferentes estilos de jogo. Estes modelos foram desenvolvidos com o objetivo de reproduzir o comportamento de jogadores reais e testar automaticamente o conteúdo dos jogos. Isso permite o uso de personas procedurais para prever e representar as ações de vários tipos de jogadores em uma variedade de cenários de jogos.

Holmgård *et al.* (2019) sugerem a implementação de uma variação do *Monte Carlo Tree Search*/Busca em Árvore de Monte Carlo (MCTS) (algoritmo de busca em árvore que combina a busca em largura com a busca em profundidade). Nessa variação, os critérios de seleção de nós são criados usando computação evolutiva em vez do critério padrão do MCTS *Upper Confidence Bound 1*/Limite superior de confiança 1 (UCB1). Essa variação permite que as heurísticas únicas de cada pessoa se desenvolvam. Isso permite a criação de modelos de jogadores que podem realizar testes mais precisos e adaptáveis em níveis de jogos complexos. Holmgård *et al.* (2019) demonstram como diferentes estilos de jogo podem ser efetivamente representados e analisados ao usar essas personas procedurais em um conjunto de níveis de jogos diversificado de níveis de jogos. Isso contribui para a melhoria e a robustez dos processos de desenvolvimento de jogos.

Os resultados mostram que personas procedurais permitem uma visualização rápida e eficaz das interações possíveis em um jogo (Figura 8a), fornecendo aos desenvolvedores informações úteis sobre como diferentes tipos de jogadores podem interagir com o conteúdo

do jogo (Figura 8b). Essa abordagem não apenas economiza tempo e recursos, mas também aumenta o escopo dos testes, permitindo a avaliação rápida de vários cenários e comportamentos de jogadores.

Figura 8 – Personas em execução no jogo MiniDungeons 2.



(a) Mapa 6 do jogo. (b) Mapas de calor do comportamento da persona no mapa 6.

Fonte: Holmgård *et al.* (2019)

Por fim, Holmgård *et al.* (2019) afirmam que esta técnica pode ser usada para coisas além do teste de jogos. As personas procedurais podem ser usadas como ferramentas interativas durante o desenvolvimento do jogo, o que permite uma análise contínua e iterativa. Alternativamente, elas podem ser usadas em sistemas de geração de conteúdo procedural, onde várias avaliações devem ser feitas rapidamente.

4.2.2 Augmenting Automated Game Testing with Deep Reinforcement Learning

Bergdahl *et al.* (2020) discutem a importância do uso de técnicas de aprendizagem por reforço (*Reinforcement learning*/Aprendizagem por reforço (RL)) para analisar e testar jogos atuais e argumentam a favor de testes automatizados utilizando simulações de Monte Carlo e modelos de jogadores construídos manualmente. Além disso, a aprendizagem supervisionada usando dados de jogos humanos foi aplicada com sucesso para testar jogos de quebra-cabeça 2D. O RL também é usado para garantia de qualidade, onde modelos baseados em Q-learning (um

algoritmo que aprende a realizar a melhor ação possível em um determinado estado com base nas recompensas recebidas) são treinados para explorar a interface gráfica de um aplicativo móvel.

Os agentes RL usam entradas contínuas para controlar suas ações, simulando controles reais de jogo. Não usar uma malha de navegação permite que os agentes explorem o ambiente mais abertamente. O agente observa o ambiente por meio de uma série de informações, incluindo posição relativa ao alvo, velocidade, rotação global, distância até o alvo, informações de escalada, contato com o solo, tempos de espera de salto, zerar temporizadores e uma matriz visual. As recompensas são usadas para encorajar o agente a se mover em direção ao objetivo e para punir o comportamento que se afasta do objetivo.

Na Figura 9a, as esferas amarelas indicam objetivos de navegação. A parede azul não tem uma malha de colisão o que permite que os agentes a atravessem. Os quadrados cor-de-rosa representam áreas onde o agente ficará preso ao entrar nelas. Além disso, na Figura 9b, as esferas amarelas indicam objetivos de navegação. A caixa azul escura representa uma parede escalável que é a única forma de alcançar os dois objetivos mais afastados da câmera. No ambiente de navegação dinâmica as 4 plataformas vermelhas se movem.

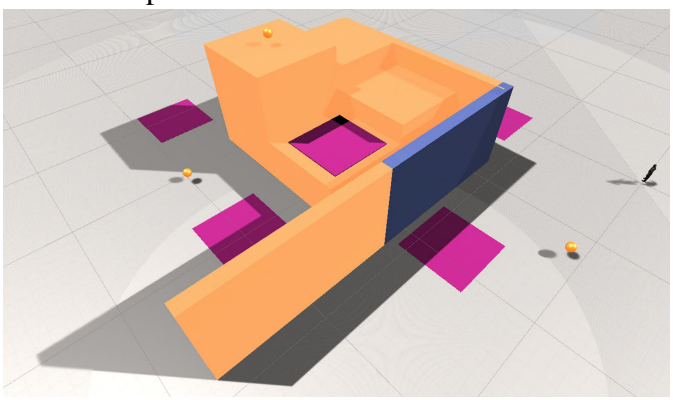
Fornecendo uma cobertura de teste mais abrangente, os agentes RL exibiram habilidades notáveis que superaram as dos agentes de *script* na detecção de explorações e *bugs*, conforme evidenciado na seção de resultados do artigo. Além disso, estes agentes demonstraram uma exploração mais diversificada do ambiente, mostrando o seu potencial excepcional. O domínio das tarefas de navegação em ambientes desafiadores ilustrou a sua capacidade de aprender tarefas complexas num período de tempo razoável.

Bergdahl *et al.* (2020) concluem que o RL pode ser uma ferramenta valiosa para aprimorar os testes de jogos, especialmente em cenários complexos de movimentação e navegação. Ele complementa os métodos tradicionais de *script* e oferece a capacidade de aprendizado e exploração, o que os torna eficazes na detecção de *bugs* e abusos. No entanto, também é enfatizado que o RL é mais adequado para tarefas bem definidas e específicas, onde desafios como simular comportamento humano, detectar abusos e avaliar dificuldades são bastante significativos.

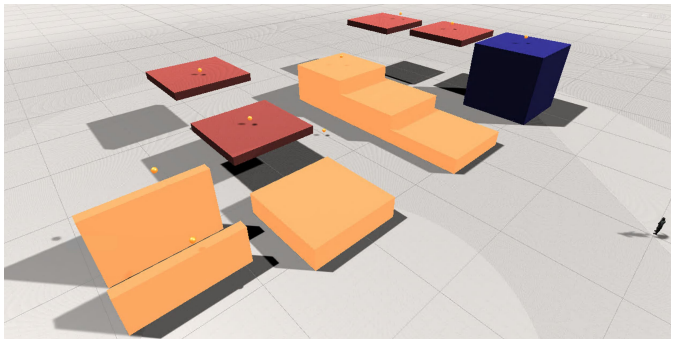
4.2.3 A Long/Short-Term Memory Based Automated Testing Model to Quantitatively Evaluate Game Design

Chen *et al.* (2020) propõem uma abordagem inovadora para avaliar o design de jogos por meio de um modelo de teste automatizado chamado LSTM-Testing, que aborda esse

Figura 9 – Ambientes utilizados para testes.



(a) *Exploit* e *Sand-box* do jogador preso.



(b) *Sand-box* de navegação e navegação dinâmica.

Fonte: Bergdahl *et al.* (2020)

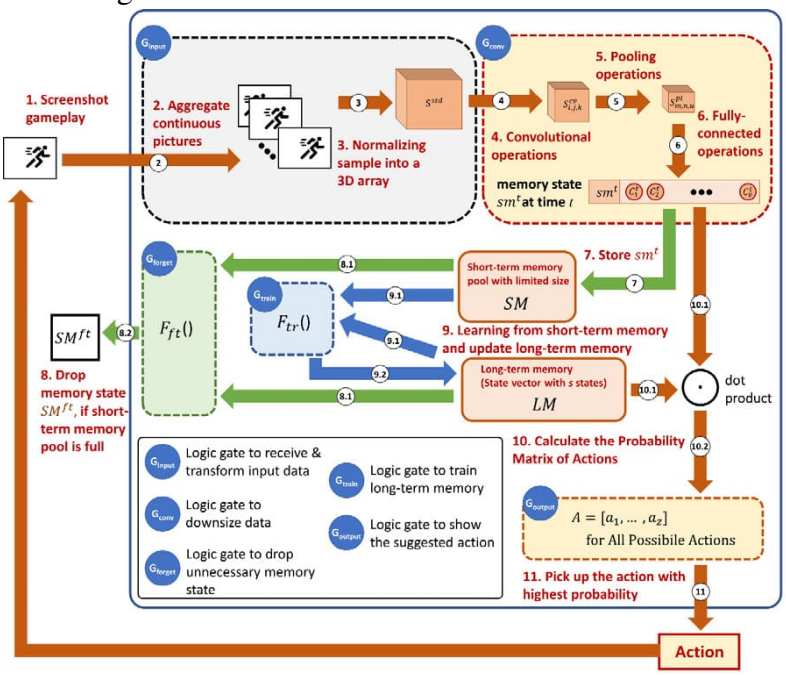
problema utilizando uma rede de LSTM para imitar o comportamento de jogadores reais. Isso facilita a capacidade do modelo de jogar repetidamente e aprender com suas experiências sem a necessidade de qualquer intervenção humana.

Na Figura 10, é mostrada a proposta do LSTM. A principal funcionalidade do LSTM-Testing é fazer capturas de tela do jogo durante o jogo e, em seguida, usar uma *Deep Neural Network*/Rede Neural Profunda (DNN) para converter as capturas de tela em equações lineares. O modelo então emprega LSTM para imitar o comportamento dos jogadores humanos, incluindo o armazenamento a longo prazo do conhecimento que já foi aprendido e o armazenamento a curto prazo do conhecimento que é relevante para as situações atuais. Ao tomar decisões e observar como o jogo responde, o LSTM-Testing pode avaliar a dificuldade do jogo e identificar áreas que precisam ser melhoradas.

Existem três vantagens do LSTM-Testing:

– Fornece uma abordagem neutra e quantitativa para avaliar o design de jogos. Isso facilita aos designers de jogos superar suas opiniões pessoais e tomar decisões com base em dados;
– Reduz o tempo e as despesas associadas ao teste manual. O teste manual é um problema

Figura 10 – LSTM-Testing framework.



Fonte: Chen *et al.* (2020)

significativo no processo de desenvolvimento de jogos, e o LSTM-Testing automatiza esse processo, liberando tempo e recursos para outras atividades;

– Libera os desenvolvedores de jogos para utilizar o seu tempo com os aspectos técnicos dos jogos, como o desenvolvimento de propriedade intelectual. A automação do teste permite que os designers se concentrem na criatividade e na inovação do design do jogo.

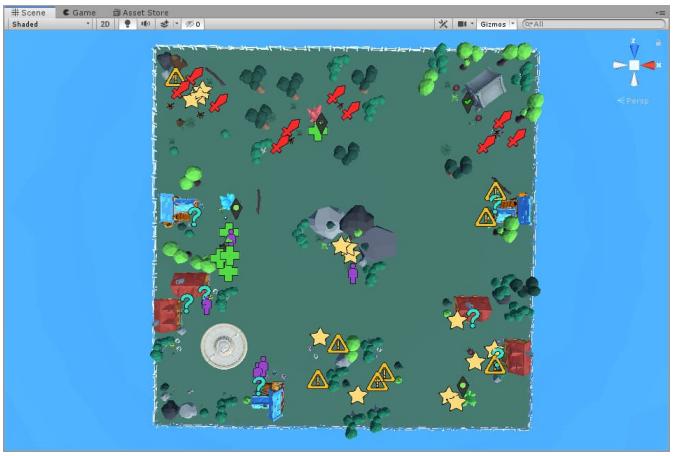
Portanto, o LSTM-Testing representa um avanço significativo na avaliação do design de jogos. Ao utilizar o aprendizado de máquina para replicar o comportamento de jogadores humanos, o LSTM-Testing fornece aos designers de jogos um mecanismo poderoso para avaliar objetivamente a dificuldade e a justiça de seus jogos. Isso pode resultar na criação de jogos mais divertidos e equilibrados para os jogadores.

4.2.4 Artificial Players in the Design Process: Developing an Automated Testing Tool for Game Level and World Design

Stahlke *et al.* (2020) descrevem o PathOS (Figura 11), uma ferramenta criada para simplificar o processo de avaliação de níveis e mundos de jogos. Para fazer isso, o PathOS simula a experiência do jogador por meio do uso de agentes artificiais controlados por um modelo comportamental.

Esses agentes não são bots aleatórios, pois eles imitam jogadores humanos por

Figura 11 – Uma captura de tela do PathOS.



Fonte: Stahlke *et al.* (2020)

meio de um modelo comportamental. Este modelo se inspira na taxonomia de Bartle, que agrupa os jogadores em quatro categorias com base em como eles jogam e como se comportam: exploradores, conquistadores, socializadores e assassinos. Os designers podem adaptar o PathOS ao seu jogo específico ajustando o peso dado a diferentes motivações, como exploração ou conquista. Por exemplo, um designer de jogos de quebra-cabeça pode criar agentes com forte foco na exploração, incentivando-os a se aprofundar no mundo do jogo.

Os agentes PathOS vão além do movimento básico (Figura 12). Eles estão equipados com uma câmera de "ponto de vista do jogador", permitindo-lhes perceber o mundo do jogo de forma semelhante aos jogadores humanos. Este modelo de percepção é crucial para que os agentes identifiquem objetos e naveguem pelo ambiente de forma eficaz.

Figura 12 – Exibição do mapa de memória espacial do agente e da câmera POV do agente é mostrada.

Fonte: Stahlke *et al.* (2020)

Portanto, o trabalho de Stahlke *et al.* (2020) oferece uma economia de tempo significativa para os desenvolvedores de jogos. Ao automatizar aspectos do teste de jogabilidade,

permite uma iteração mais rápida e a identificação de problemas de design no início do desenvolvimento. Além disso, a ferramenta se integra perfeitamente a projetos existentes e não requer conhecimento de programação dos designers.

4.2.5 Automated Video Game Testing Using Synthetic and Human-Like Agents

Ariyurek *et al.* (2021) apresentam uma abordagem diferente para testar videogames usando testadores automatizados. Existem duas categorias de testadores disponíveis: os sintéticos e os análogos a humanos. Os testadores sintéticos são programados para encontrar falhas nos jogos, procurando por erros nos cenários criados a partir dos jogos. Por outro lado, os testadores análogos a humanos empregam objetivos de teste derivados das ações de testadores humanos usando um algoritmo chamado aprendizado de reforço múltiplo guloso (*greedy multiple reinforcement learning*).

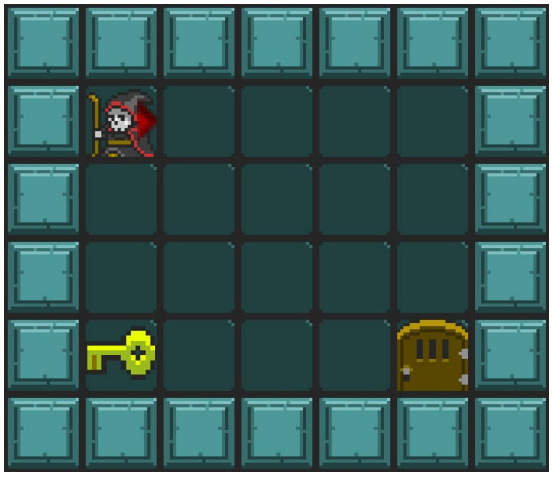
Os pesquisadores usaram a estrutura *General Video Game AI*/IA geral de jogos digitais (GVG-AI) para criar três jogos de tamanhos e complexidades variados, chamados Jogo A, Jogo B e Jogo C (Figura 13), onde cada jogo tem objetivos e layouts diferentes.

No jogo, 45 erros (como remoção de linhas do conjunto de interação, alteração da ordem ou nome das tags atribuídas a diferentes objetos ou personagens nas interações e adição de interações incorretas) foram introduzidos intencionalmente no *script* de descrição do jogo para avaliar a experiência do jogo e a capacidade do agente de identificar problemas.

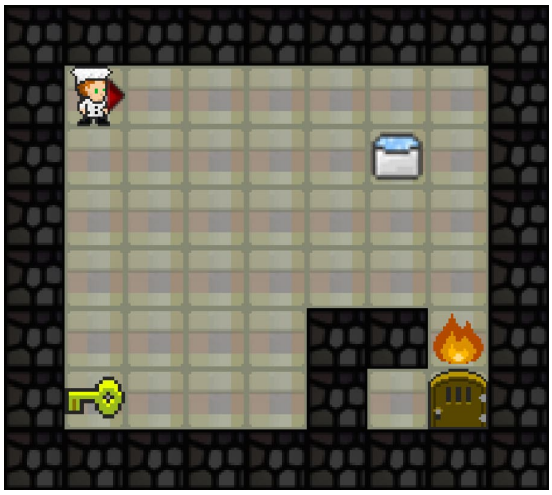
Testadores humanos, compostos por 15 participantes com diversas experiências em jogos e testes, foram solicitados a jogar sem instruções específicas e coletar trajetórias de jogo usando a estrutura GVG-AI. Uma trajetória é uma sequência de ações que um jogador realiza enquanto explora o jogo. Além disso, o estudo introduz "estados de teste" para capturar e executar comportamentos de testadores humanos. Os resultados dos experimentos revelaram várias descobertas interessantes:

- Agentes sintéticos, com um limiar de probabilidade (um parâmetro usado para controlar a semelhança entre o comportamento dos agentes semelhantes a humanos e os testadores humanos) de 0.0, foram os mais semelhantes ao comportamento humano e encontraram todos os *bugs* no jogo mais simples (Jogo A). Um limiar de probabilidade de 0.0 indica que esses agentes estão totalmente comprometidos em seguir o comportamento humano, deixando pouco espaço para a exploração de novos comportamentos.
- No Jogo B, os agentes análogos a humanos superaram os agentes sintéticos, e o agente de

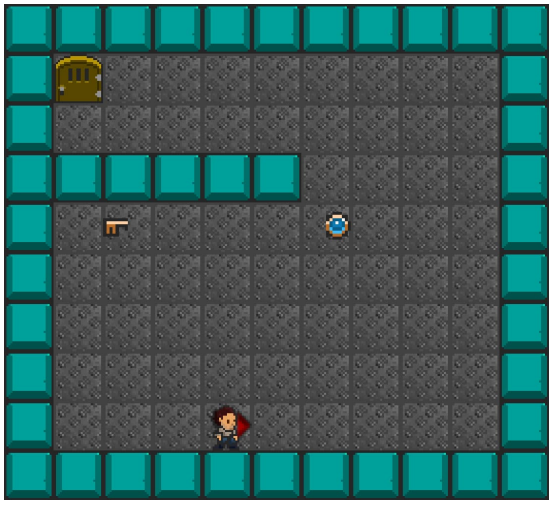
Figura 13 – Representando o início do jogos.



(a) Jogo A - Fase 1.



(b) Jogo B - Fase 1.



(c) Jogo C - Fase 1.

Fonte: Ariyurek *et al.* (2021)

referência (um agente simples que realiza ações básicas no jogo e é usado como um ponto de referência para comparar o desempenho de outros agentes) teve um bom desempenho, sugerindo que alguns testadores humanos não conseguiram cobrir todos os caminhos pretendidos.

– No Jogo C, o desempenho dos testadores humanos superou o dos agentes análogos a humanos, mas os agentes análogos a humanos com uma probabilidade de semelhança de 0.0 tiveram um desempenho semelhante ao humano.

Esta pesquisa dá uma contribuição significativa ao campo de testes de jogos, fornecendo conhecimentos valiosos sobre como usar agentes para melhorar o processo de testes de videogames. No futuro, os pesquisadores planejam explorar o uso de aproximadores de funções em agentes de aprendizagem por reforço, melhorar o algoritmo *Maximum causal entropy inverse rein-*

forcement learning/Aprendizado por reforço inverso de entropia causal máxima (MGP-IRL) para lidar com a aleatoriedade e estender esta abordagem a jogos Three-dimensional/Tridimensional (3D).

4.2.6 Improving Playtesting Coverage via Curiosity Driven Reinforcement Learning Agents

Gordillo *et al.* (2021) propõem uma nova solução que utiliza agentes RL para automatizar o processo de exploração e teste. A ideia central envolve treinar agentes de RL para explorar um ambiente de jogo da forma mais abrangente possível. Esses agentes são recompensados por realizar ações que os levam a áreas novas e desconhecidas. Essa curiosidade intrínseca motiva os agentes a explorar ativamente e descobrir problemas que os testadores humanos podem perder.

Os autores descrevem um cenário específico (Figura 14) para avaliar a eficácia da abordagem baseada em RL. Este cenário é um ambiente 3D complexo com mecânicas de navegação desafiadoras, como saltos, objetos escaláveis e elevadores. Os resultados demonstram que os agentes de RL curiosos superam as técnicas de exploração tradicionais em termos de cobertura e identificação de problemas. Os agentes navegam pelo ambiente complexo e coletam dados valiosos para identificar falhas, *bugs* e falhas de design.

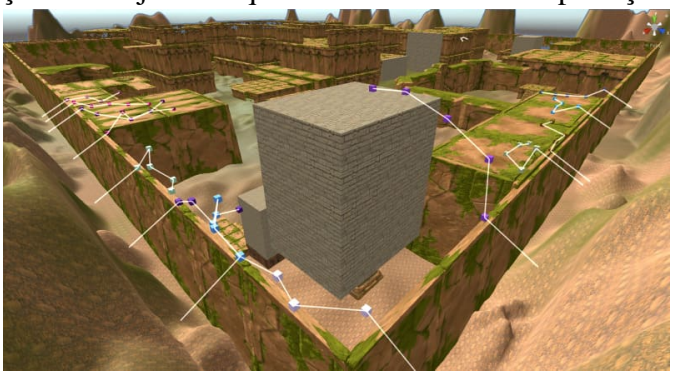
Figura 14 – O ambiente com desafios de navegação complexos.

Fonte: Gordillo *et al.* (2021)

Além da abordagem central de RL, o trabalho explora técnicas de visualização para analisar os dados coletados pelos agentes (Figura 15). Essas visualizações ajudam os designers de jogos a entender como os jogadores interagem com o ambiente e identificam áreas que requerem refinamento adicional. Essa abordagem pode melhorar significativamente a eficiência e o rigor dos testes, levando ao desenvolvimento de jogos de maior qualidade.

Além dos benefícios mencionados, o uso de agentes de RL para testes também pode

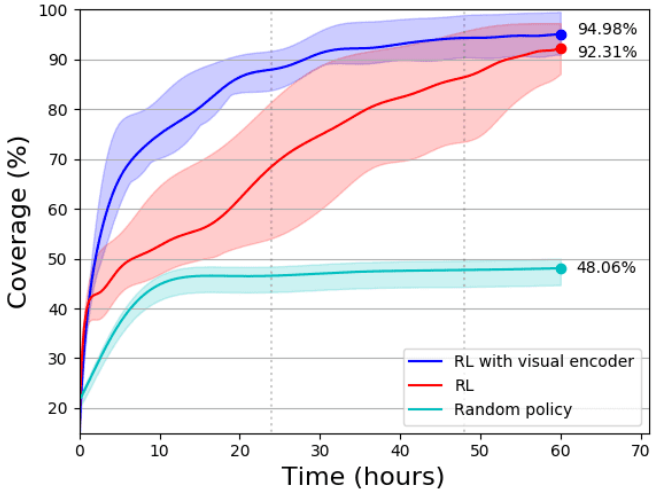
Figura 15 – Visualização das trajetórias que saem do limite de exploração.



Fonte: Gordillo *et al.* (2021)

reduzir os custos gerais de desenvolvimento (Figura 16). Ao identificar problemas no início do processo, os desenvolvedores podem corrigi-los de forma mais rápida e econômica, evitando retrabalhos dispendiosos posteriormente no ciclo de desenvolvimento. No entanto, a abordagem baseada em RL também apresenta desafios. Treinar agentes de RL eficazes pode exigir uma quantidade significativa de dados e poder computacional. Além disso, projetar funções de recompensa adequadas para incentivar a exploração valiosa pode ser complexo.

Figura 16 – O gráfico mostra a média e a variação do desempenho de diferentes políticas em 5 execuções diferentes.



Fonte: Gordillo *et al.* (2021)

Apesar desses desafios, Gordillo *et al.* (2021) afirmam que o uso de agentes de RL para testes automatizados de videogames é uma técnica promissora com grande potencial para melhorar a indústria de jogos.

4.2.7 CCPT: Automatic Gameplay Testing and Validation with Curiosity-Conditioned Proximal Trajectories

Sestini *et al.* (2022) discutem a importância do teste de jogabilidade na produção de jogos de próxima geração, propondo o algoritmo *Curiosity-conditioned proximal trajectories*/Curiosidade-condicionada trajetórias proximais (CCPT), que pode identificar automaticamente problemas no design de jogos.

O algoritmo CCPT consiste em quatro componentes principais. O primeiro deles é uma plataforma de treinamento para agentes de exploração e navegação. A plataforma de treinamento é um ambiente de navegação 3D que simula cenários de jogos complexos com diversos obstáculos e desafios (Figura 17). Os agentes são treinados para explorar o ambiente e identificar falhas de design, como falta de hitboxes e falhas.

Figura 17 – Visão geral do ambiente proposto.

(a) Foto do ambiente.

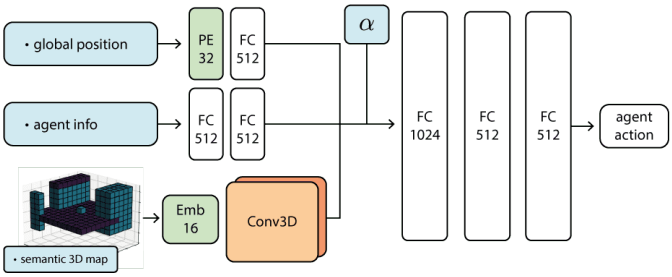
(b) Visão de cima.

Fonte: Sestini *et al.* (2022)

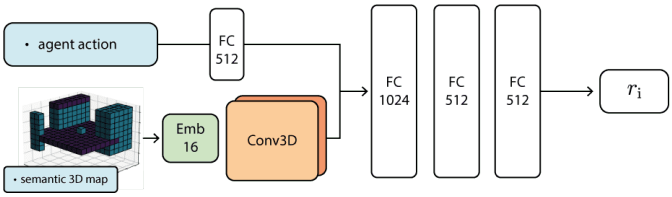
O segundo componente é uma arquitetura de rede neural para agentes de navegação e exploração (Figura 18). As arquiteturas de redes neurais são projetadas para processar dados de entrada do ambiente e gerar ações que maximizem a curiosidade e a exploração do agente. As redes neurais consistem em três partes:

– Navegação: Recebe o estado atual do ambiente, gerando ação de movimento do agente por meio de camadas conectadas com ativação *Rectified linear unit*/Unidade linear retificada (ReLU) e uma camada de processo gaussiano (Figura 18a).
– Imitação: Utiliza demonstrações de especialistas para gerar uma política imitativa. Processa o mapa de ocupação 3D do ambiente com uma rede neural convolucional e informações do agente com camadas totalmente conectadas. A saída é passada por uma camada de processo gaussiano (Figura 18b).
– Curiosidade: Com base no estado do ambiente e interno do agente, gera uma recompensa de curiosidade. Isso é realizado por meio de camadas de incorporação posicional e camadas totalmente conectadas, seguidas por uma camada linear para a recompensa de curiosidade (Figura 18c).

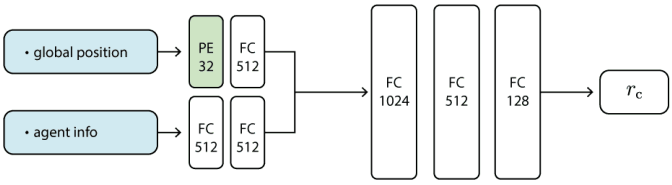
Figura 18 – Visão geral das arquiteturas de módulo utilizadas neste trabalho.



(a) Módulo de Navegação.



(b) Módulo de Imitação.



(c) Módulo de Curiosidade.

Fonte: Sestini *et al.* (2022)

O terceiro componente é o uso de novos algoritmos de exploração demonstrados por especialistas (Figura 19). Essas demonstrações são utilizadas para orientar a exploração do agente e garantir que ele explore as partes mais relevantes do ambiente. Este algoritmo é um componente chave do algoritmo CCPT, pois permite ao agente explorar conscientemente a proximidade de uma trajetória desejada.

Figura 19 – Algoritmo da ferramenta CCPT.

Algorithm 1 Training with CCPT

input: M dataset of expert demonstrations, ϵ filter threshold
$\pi \leftarrow$ initialize policy
$D \leftarrow$ initialize AMP discriminator
$\tilde{\phi} \leftarrow$ initialize RND target network
$\phi \leftarrow$ initialize RND predictor network
$G \leftarrow$ initialize external dataset

while not done **do** ▷ models training
  for $i = 1, ..., m$ **do**
    $B \leftarrow$ initialize policy experience dataset
    $\alpha_i \sim [0, 1]$ ▷ sample exploration value
    $\theta_i \sim \pi$ ▷ sample trajectory
    for $t = 1, ..., T$ **do**
      $r_i^t = \max[0, 1 - 0.25(D(s_t, a_t) - 1)^2]$
      $r_e^t = ||(\tilde{\phi}(s_{t+1}) - \phi(s_{t+1}))^2||$
      $R^t = \alpha_i \cdot r_e^t + (1 - \alpha_i) \cdot r_i^t + r_s^t$
      Store R^t in θ_i
    end for
    Store θ_i in B
    Store (θ_i, α_i) in G ▷ store all trajectories in the external dataset
  end for
  Update D with $\mathcal{L}^{\text{AMP}}$ with samples from B
  Update ϕ with $\mathcal{L}^{\text{RND}}$ with samples from B
  Update π with $\mathcal{L}^{\text{PPO}}$ with samples from B
end while

$\Theta \leftarrow$ initialize set
for trajectory $\theta_i \in G \mid \alpha_i >= 0.5$ **do** ▷ filtering results
  $\bar{r}_c^i = \frac{\sum_{t=0}^{T} r_c(s_t)}{T}$
  if $\bar{r}_c^i(\theta_i) > \epsilon$ **then**
    Store θ_i in Θ
  end if
end for
return Θ ▷ return set of broken trajectories

Fonte: Sestini *et al.* (2022)

O quarto componente é um conjunto de métricas de avaliação usadas para medir o desempenho do agente. Segundo Sestini *et al.* (2022), o CCPT supera esses métodos em termos de precisão e eficiência, conforme evidenciado por métricas como cobertura, *bugs* encontrados e *bugs* destacados (Tabela 5).

Tabela 5 – Resultados quantitativos comparados com bases de referência:

	Cobertura	***Bugs* encontrados**	***Bugs* destacados**
CCPT	1.91	**13**	**13**
Combinação linear	1.14	7	7
Apenas por imitação	0.84	1	0
Apenas por curiosidade	**2.39**	10	3

Fonte: Sestini *et al.* (2022)

Sestini *et al.* (2022) também descrevem detalhadamente o processo de treinamento do algoritmo CCPT e como ele combina curiosidade e aprendizagem por imitação para treinar agentes. O treinamento do algoritmo CCPT requer a execução de 10 agentes em paralelo na mesma máquina e consiste nas seguintes etapas:

1. Primeiro, especialistas fornecem demonstrações para delimitar a área de teste.
2. O algoritmo combina aprendizado por imitação e curiosidade para explorar trajetórias próximas às demonstradas, guiado por uma função de recompensa intrínseca. Isso estimula a exploração em áreas não abrangidas pelas demonstrações.
3. Durante o treinamento, o agente aprende a navegar no ambiente e detectar problemas. Também aprende a identificar trajetórias problemáticas para designers de jogos.
4. O algoritmo é treinado com um conjunto de hiperparâmetros escolhidos após experimentos preliminares.

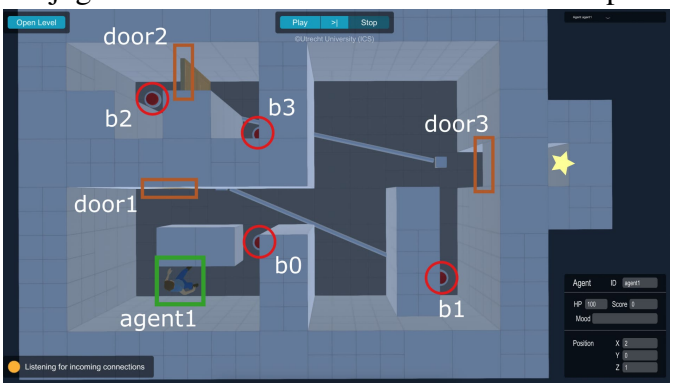
Em resumo, o algoritmo CCPT é um método promissor para automatizar o processo de teste de design de jogos. Os resultados demonstram que o algoritmo CCPT tem potencial para melhorar significativamente a qualidade do jogo, identificando falhas de design e melhorando a experiência geral do jogo.

4.2.8 EvoMBT: Evolutionary model based testing

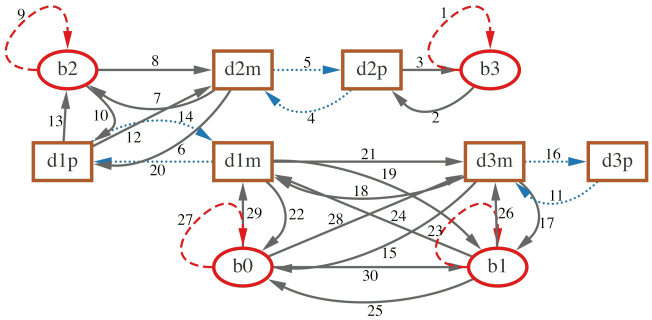
Ferdous *et al.* (2023) abordam um problema comum no desenvolvimento de software: testar sistemas complexos que tenham uma interação direta com o próprio sistema. Isso pode ser atribuído à natureza complexa do sistema, a questões de segurança ou à incapacidade de criar um ambiente de teste com uma configuração específica. O *Evolutionary Model Based Testing*/Testes Baseados em Modelos Evolutivos (EvoMBT) entra em ação com a mesma abordagem de teste empregada nos testes baseados em modelo (MBT). Aqui, é criada uma representação simplificada do sistema, chamada *Extended Finite State Machine model*/Máquina de Estados Finito (EFSM). Este modelo captura os possíveis estados do sistema e as transições entre eles, desencadeadas por vários eventos.

EvoMBT se destaca por sua capacidade de utilizar este modelo para criar casos de teste. Ele emprega vários métodos de busca para explorar o EFSM e identificar sequências de eventos (caminhos) que o sistema pode localizar (Figura 20). Esses caminhos representam cenários potenciais que podem ser seguidos para identificar erros no comportamento real do sistema.

Figura 20 – Um nível do jogo Lab Recruits e o modelo EFSM correspondente.



(a) Um exemplo de um nível em Lab Recruits (um jogo 3D utilizado como plataforma de testes no contexto de pesquisa de testes de software). Neste nível, há quatro botões e três portas.



(b) Modelo EFSM do nível Lab Recruits. Os estados representam botões ou lados das portas, as transições representam movimento ou ação (pressionar um botão).

Fonte: Ferdous *et al.* (2023)

Além de suas funcionalidades primárias, o EvoMBT tem um histórico documentado de sucesso em cenários do mundo real. Ele foi fundamental nos testes do *Lab Recruits*, um jogo 3D destinado ao uso em experimentos de IA. Nele, o EvoMBT produz labirintos e cria automaticamente instâncias que podem percorrer, isso garante a funcionalidade do jogo nesses ambientes virtuais.

As habilidades do instrumento vão além do domínio dos jogos. Na plataforma experimental *Cyber-Physical Systems*, a EvoMBT participou de um concurso para criar cenários virtuais de estradas que desafiam a capacidade dos carros autônomos de manter seu curso. Ao criar estas complexas configurações de estradas e avaliá-las numa simulação de condução, o EvoMBT teve um impacto significativo na segurança e confiabilidade dos veículos autônomos.

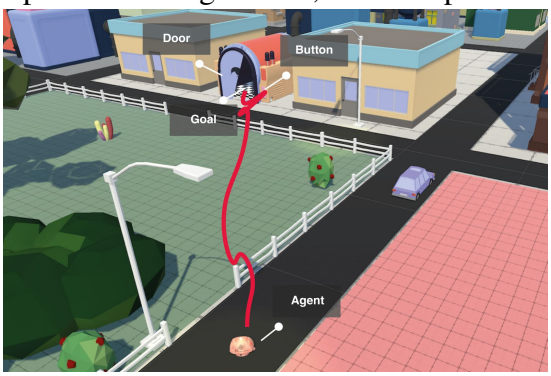
Em conclusão, o EvoMBT se torna uma ferramenta versátil e poderosa que aborda os problemas associados ao teste de sistemas complexos. Sua funcionalidade diversificada e seus atributos de natureza de código aberto tornam-no um recurso significativo para pesquisadores,

desenvolvedores e qualquer pessoa interessada em garantir a estabilidade de sistemas de software complexos.

4.2.9 Towards Informed Design and Validation Assistance in Computer Games Using Imitation Learning

Sestini *et al.* (2023) propõem uma nova abordagem para validação e teste automatizados de jogos usando *Imitation Learning*/Aprendizagem por imitação (IL). A IL permite que um agente aprenda observando e imitando o comportamento de um especialista humano, como um jogador habilidoso. O agente IL treinado pode então testar o jogo, imitando as ações do humano e identificando problemas de jogabilidade (Figura 21).

Figura 21 – Experimento com o agente IL. Na Figura 21a, é visto o processo para validar as modificações no design de um sistema, se elas atendem aos requisitos e funcionam conforme esperado. Na Figura 21b, é visto o processo para validar a navegação no sistema.



(a) Captura de tela do caso de uso de *Validation Design Changes*.

(b) Captura de tela do caso de uso de *Navigation*.

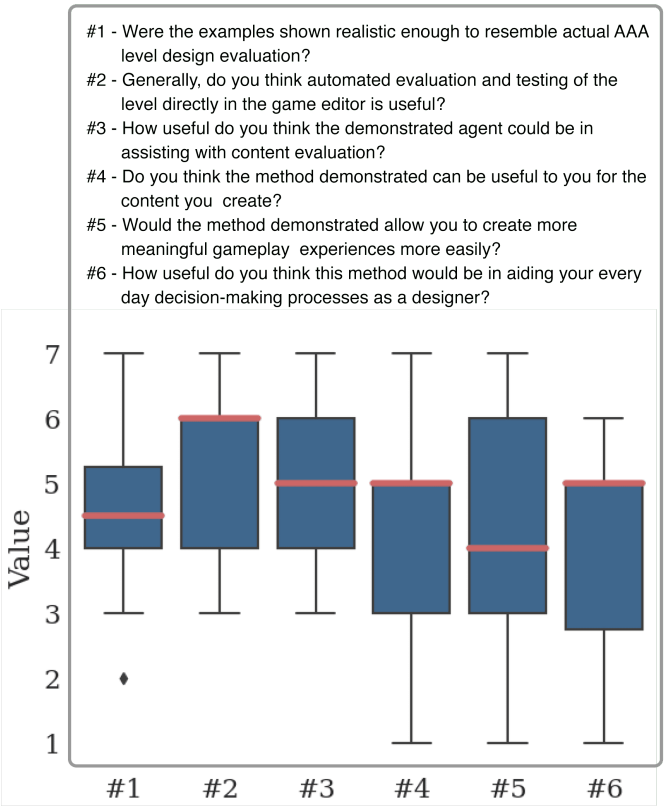
Fonte: Sestini *et al.* (2023)

A IL oferece várias vantagens sobre as abordagens tradicionais. Primeiramente, a IL é eficiente, eliminando a necessidade de criação e manutenção manual de testes. Segundo Sestini *et al.* (2023), a IL oferece controle, pois o comportamento do agente IL pode ser facilmente ajustado observando o especialista humano. Terceiro, a IL é robusta, capturando uma quantidade mais ampla de problemas de jogabilidade em comparação aos métodos de teste tradicionais, imitando a exploração e a tomada de decisão de um jogador humano.

Um estudo com especialistas da indústria validou a abordagem baseada em IL, e demonstrou que ela tem potencial para reduzir o esforço e melhorar a qualidade dos testes de jogabilidade modernos (Figura 22).

No entanto, existem desafios que precisam ser superados com a IL. Primeiro, garantir

Figura 22 – Boxplot de algumas das perguntas mais importantes da pesquisa.



Fonte: Sestini *et al.* (2023)

que o agente IL possa adaptar seu conhecimento a vários cenários de jogo além daqueles em que foi treinado é crucial para que seja amplamente aplicado. Segundo Sestini *et al.* (2023), compreender as decisões tomadas pelo agente IL é importante para depurar e corrigir problemas identificados durante o teste. A falta de interpretabilidade pode dificultar a identificação da causa raiz dos problemas. Terceiro, tornar o agente IL resiliente a jogadores mal-intencionados que tentam enganá-lo é importante para garantir a validade do processo de teste.

Apesar desses desafios, a IL tem o potencial de revolucionar o teste de jogos reduzindo significativamente o tempo e o custo envolvidos. Pesquisas futuras se concentrarão em superar esses desafios e refinar a utilidade da IL na validação de jogos.

4.2.10 Technical Challenges of Deploying Reinforcement Learning Agents for Game Testing in AAA Games

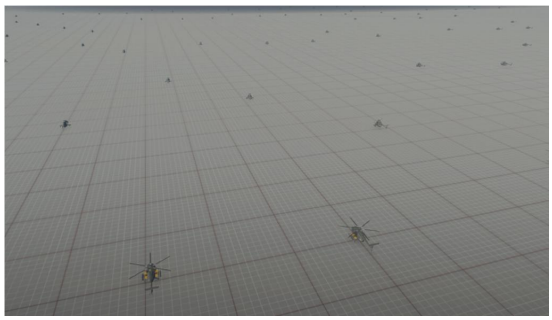
Gillberg *et al.* (2023) exploram os desafios técnicos da implementação de agentes RL para testes de jogos *Triple A*/Triplo A (AAA) (Alta qualidade, Alta produção, Alta audiência) (Figura 23). Embora os agentes de RL tenham potencial para aumentar a cobertura dos testes e identificar falhas na jogabilidade, vários problemas precisam ser resolvidos antes de sua adoção

generalizada.

Outro desafio é a "caixa preta"da tomada de decisão pelos agentes RL. Muitas vezes, as razões por trás de suas ações não são facilmente compreendidas. Isso pode levar a problemas em cenários de segurança crítica em jogos AAA. Por exemplo, a tomada de decisões que afetam negativamente a experiência do jogador ou a integridade do jogo exige o desenvolvimento de métodos para tornar o processo de tomada de decisão mais transparente.

Figura 23 – Exemplo de configuração de ambiente que vai do protótipo à produção no Battlefield 2042: três ambientes de treinamento de RL diferentes diferentes.

(a) Ambiente de protótipo.



(b) Ambiente da faixa de teste.

(c) Ambiente de produção.

Fonte: Gillberg *et al.* (2023)

Em comparação com os agentes RL, os bots baseados em scripts se mostram inadequados para tarefas que exigem exploração e adaptação. A rigidez dos scripts os torna incapazes de lidar com situações inesperadas ou ambientes em constante mudança, características comuns em jogos AAA.

Além disso, os agentes RL podem precisar de quantidades de dados de treinamento maiores do que as atualmente viáveis para coleta em jogos AAA. O processo de obtenção e rotulagem de grandes volumes de dados para treinar esses agentes de forma eficiente ainda representa um obstáculo significativo.

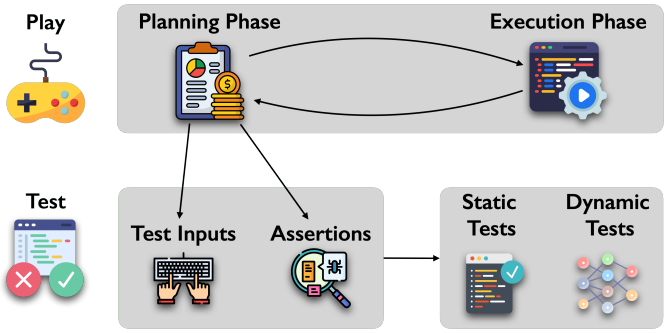
Apesar dos desafios, Gillberg *et al.* (2023) concluem que os agentes RL não devem ser vistos como substitutos para bots scriptados, mas sim como ferramentas complementares.

A combinação de ambas as abordagens, juntamente com o desenvolvimento de soluções para os desafios técnicos identificados, pode contribuir para testes de jogos AAA mais eficientes, abrangentes e eficazes.

4.2.11 PlayTest: A Gamified Test Generator for Games

Feldmeier *et al.* (2023) descrevem um gerador de testes gamificado para jogos que utiliza crowdsourcing para criar testes (Figura 24). Ele alcança isso separando o processo de geração de testes em dois módulos: Jogar e Testar. O módulo Jogar utiliza uma abordagem baseada em jogo onde dois jogadores competem entre si para identificar o máximo possível de variações do programa (mutantes). Isso é alcançado por meio de duas fases: planejamento e execução.

Figura 24 – Visão geral do PlayTest.



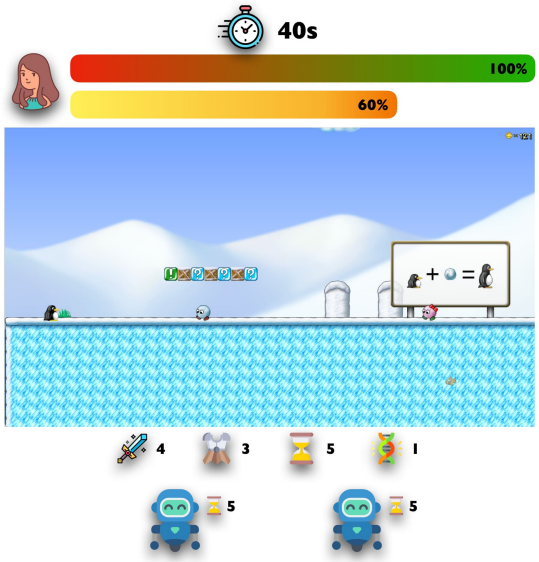
Fonte: Feldmeier *et al.* (2023)

Durante a fase de planejamento (Figura 25), os jogadores gravam rastros de jogabilidade jogando o jogo por um tempo limitado. Esses rastros são então usados para gerar asserções, que são essencialmente condições que precisam ser atendidas durante o jogo. Os jogadores podem comprar asserções usando pontos de ação ganhos ao longo do jogo. Os pontos de ação também podem ser usados para melhorar vários atributos do jogador, como poder de ataque ou armadura.

Após o término da fase de planejamento, a fase de execução começa (Figura 26). Durante esta fase, o jogo executa os rastros de jogabilidade gravados junto com as asserções em várias versões modificadas do jogo. Se as asserções detectarem com sucesso uma mutação, o jogador sobrevive; caso contrário, o jogador perde pontos de vida.

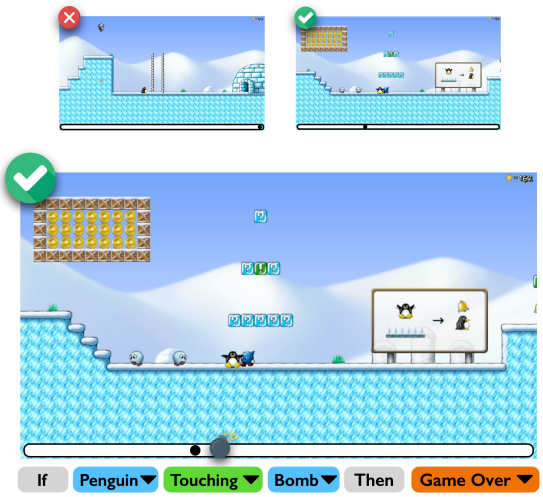
O módulo Testar, trabalhando em segundo plano, aproveita os rastros de jogabilidade e asserções coletados durante o módulo Jogar para gerar casos de teste. Esses casos de teste

Figura 25 – Interface de usuário da fase de planejamento.



Fonte: Feldmeier *et al.* (2023)

Figura 26 – Interface de usuário da fase de execução.



Fonte: Feldmeier *et al.* (2023)

podem ser estáticos, o que significa que seguem um conjunto predeterminado de instruções, ou dinâmicos, o que significa que podem se adaptar ao comportamento aleatório do jogo.

Os autores afirmam que o PlayTest é capaz de gerar casos de teste "novos e de alta qualidade". No entanto, eles também reconhecem que o PlayTest pode citar casos de teste já "existentes"em alguns casos. Isso ocorre porque os rastros de jogabilidade usados para gerar os casos de teste podem conter comportamentos que já foram testados anteriormente. Nesses casos, o PlayTest pode simplesmente repetir os testes que já foram realizados.

No geral, PlayTest é uma abordagem inovadora para gerar casos de teste para jogos que aproveita o poder da gamificação e do *crowdsourcing*. Ao separar o processo de geração de teste em módulos distintos e utilizar uma abordagem baseada em jogo, a ferramenta oferece uma

solução envolvente e eficaz para melhorar a qualidade dos jogos.

4.3 Considerações finais

A Tabela 6 resume quais tipos de testes e de agentes foram utilizados em cada artigo analisado neste trabalho.

Tabela 6 – Parâmetros de Avaliação.

Artigos	**Tipos de teste**	**Tipos de agente**
Holmgård *et al.* (2019)	Teste de jogabilidade(*Playtesting*), Teste automatizado, Testes com personas	Personas procedurais
Bergdahl *et al.* (2020)	Testes de navegação, Testes com RL)	Agentes RL com controles contínuos, Agentes RL para detecção de *exploits*, Agentes RL para tarefas de navegação
Chen *et al.* (2020)	Teste manual, Teste automatizado, Teste com memória de longo/curto prazo (LSTM-Testing)	Agentes LSTM (*Long Short-Term Memory*)
Stahlke *et al.* (2020)	Testes de navegação de jogadores, Testes de fluxo de usuário	Agentes com modelo de percepção, Agentes com modelo de memória, Agentes baseados em modelos de aprendizado imitativo
Ariyurek *et al.* (2021)	Testes baseados em cenários (*scenario-based testing*), Testes baseados em modelos (*model-based testing*), Testes beta automatizados, Testes de aventura automatizados	Agentes sintéticos, Agentes análogos a humanos

Tabela 6 – Parâmetros de Avaliação (continuação)

Artigo	**Tipos de teste**	**Tipos de agente**
Gordillo *et al.* (2021)	Testes automatizados com agentes, Testes com personas	Agentes de aprendizado por reforço baseados em curiosidade, Agentes com exploração baseada em contagem, Agentes usando demonstrações humanas
Sestini *et al.* (2022)	Testes de navegação e exploração, Testes de curiosidade	Agentes MCTS, Agentes Sarsa
Ferdous *et al.* (2023)	Testes baseados em modelos (*Model-Based Testing - MBT*), Testes baseados em busca (*search-based software testing*), Testes de jogabilidade	Agentes humanos, agentes artificiais
Sestini *et al.* (2023)	Testes baseados em cenários (*scenario-based testing*), Testes automatizados	Agentes de exploração, Agentes treinados com IL(*Imitation Learning - IL*) e RL
Gillberg *et al.* (2023)	Testes automatizados com bots scriptados, Testes baseados em cenários (*scenario-based testing*)	Agentes RL, Agentes de aprendizado por imitação (*Imitation Learning - IL*)
Feldmeier *et al.* (2023)	Testes automatizados baseados em ações dos jogadores, Testes de mutação, Testes crowdsourcing	Agentes neuroevolutivos, Agentes de crowdsourcing

Pode-se perceber com base nas análises e nos resultados que o uso do aprendizado de máquina pode melhorar os processos de desenvolvimento e qualidade dos jogos digitais. Os trabalhos analisados mostram que as vantagens superam as desvantagens, apesar dos obstáculos

e problemas, como a necessidade de ajustes minuciosos nos hiperparâmetros e a complexidade dos algoritmos de IA. Além disso, a integração dessas ferramentas pode aumentar a cobertura de teste e identificar erros que podem passar despercebidos nos testes manuais.

5 RESULTADOS

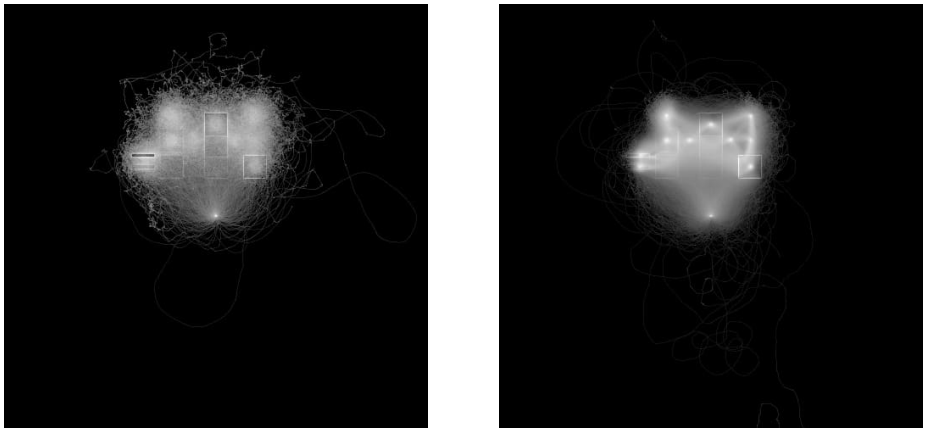
5.1 Introdução

Neste capítulo, serão apresentadas detalhadamente as questões de pesquisa que nortearam este estudo e será feita uma análise aprofundada dos resultados obtidos através da revisão da literatura.

5.2 Quais as práticas mais recentes para automatizar testes em jogos digitais?

Nos trabalhos analisados, o uso de técnicas de aprendizagem por reforço (RL), conforme discutido em Bergdahl *et al.* (2020), Gordillo *et al.* (2021), Sestini *et al.* (2022), Gillberg *et al.* (2023), é um procedimento bastante comum. A utilização desses métodos permite que agentes de RL inspecionem o ambiente do jogo para encontrar explorações e erros que os métodos convencionais podem ter dificuldade em identificar (Figura 27). Além disso, Ariyurek *et al.* (2021) enfatizam o uso de agentes que imitam o comportamento humano para testar aspectos centrados no jogador, aumentando a relevância dos testes para a experiência real do usuário.

Figura 27 – Mapas de calor gerados durante o treino na *Dynamic Navigation sand-box*.



(a) Agente após 20 passos (b) Agente treinado

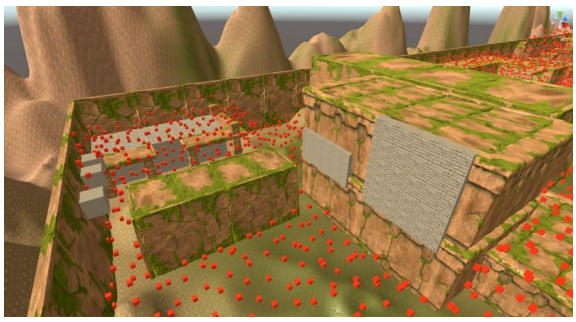
Fonte: Bergdahl *et al.* (2020)

Conforme discutido em Chen *et al.* (2020), um método eficaz para avaliar quantitativamente o design do jogo é o uso de modelos de memória de longo e curto prazo (LSTM). Este modelo ajuda a encontrar problemas de design e de jogabilidade que possam prejudicar a experiência do jogador.

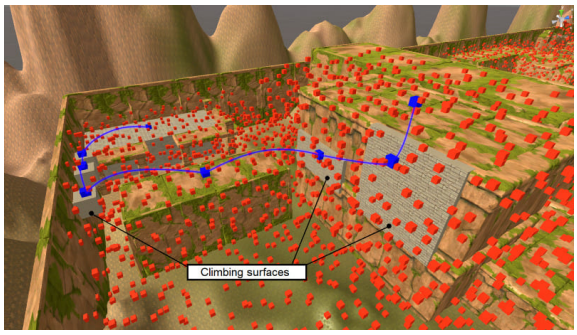
Além disso, como descrito em Gordillo *et al.* (2021), Sestini *et al.* (2022), o uso

de agentes de aprendizado por reforço baseados em curiosidade tem o potencial de aumentar a cobertura dos testes ao incentivar os agentes a explorar áreas do jogo que normalmente não seriam testadas, fazendo com que isso possa resultar na descoberta de novos *bugs* e melhorias na jogabilidade (Figura 28).

Figura 28 – Um dos desafios de navegação espalhados pelo mapa e a solução encontrada pelos agentes de RL de treinamento.



(a) Desempenho de exploração de uma estratégia de exploração aleatória, quando foram feitos desafios de navegação.



(b) Os agentes conseguiram chegar ao topo da figura da esquerda pulando e escalando uma série de obstáculos.

Fonte: Gordillo *et al.* (2021)

De acordo com Holmgård *et al.* (2019), a técnica de personas procedurais com MCTS permite a simulação automatizada de vários estilos de jogo, o que facilita a detecção de problemas ou desequilíbrios no design de jogos. Essa técnica também é capaz de replicar comportamentos diferentes dos jogadores, cobre vários cenários de jogo e garante que várias interações e estratégias sejam testadas de forma eficaz.

Por fim, de acordo com Ariyurek *et al.* (2021), Ferdous *et al.* (2023), a evolução dos testes baseados em modelos (EvoMBT), que cria casos de teste automaticamente usando algoritmos evolutivos, permite uma abordagem dinâmica e flexível, que é essencial para o desenvolvimento contínuo e a crescente complexidade dos jogos digitais.

5.3 Quais são as vantagens e desvantagens das diferentes abordagens de testes automatizados de jogos?

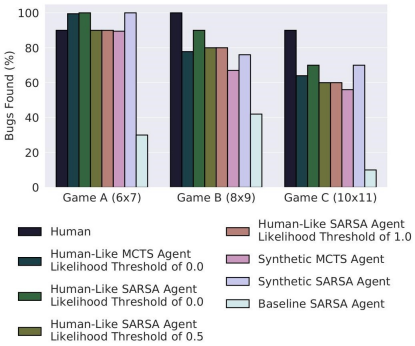
Os trabalhos analisados apontaram diversas vantagens em utilizar as diferentes abordagens de testes. Conforme Bergdahl *et al.* (2020), os agentes podem aprender a explorar os mecanismos do jogo com base em um sinal de recompensa definido pelo usuário, aumentando assim a cobertura do teste e descobrindo mecanismos de jogo não intencionais, explorações

e *bugs* em vários tipos de jogos. Ainda segundo Stahlke *et al.* (2020), em comparação com testes manuais, os testes automatizados permitem uma cobertura de testes mais ampla. Isso inclui a capacidade de testar rapidamente uma variedade de cenários e variações para descobrir problemas que podem ter sido esquecidos.

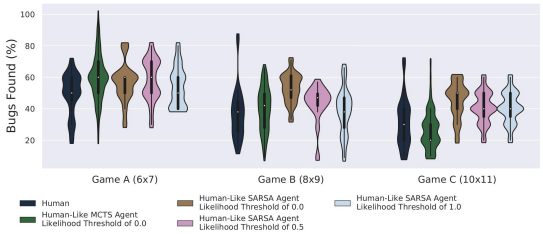
De acordo com Gordillo *et al.* (2021), ao eliminar a variabilidade humana, testes automatizados garantem que os mesmos passos sejam executados da mesma maneira toda vez. Isso aumenta a consistência da detecção de defeitos.

Em Ariyurek *et al.* (2021), os agentes sintéticos aceleram o processo de teste real e os agentes humanos proporcionam uma perspectiva humana (Figura 29). Conforme Ferdous *et al.* (2023), a automação dos testes acelera o processo de desenvolvimento e reduz drasticamente o tempo necessário para testar repetidamente. Isso é essencial em ciclos de desenvolvimento rápidos, onde é necessário ser rápido para introduzir correções e atualizações.

Figura 29 – Comparações de resultados de *bugs* detectados.



(a) Porcentagem de *bugs* detectados por testadores humanos e por agentes gerados para cada jogo em teste.



(b) Porcentagem de *bugs* detectados por testadores humanos e agentes análogos a humanos para cada jogo testado.

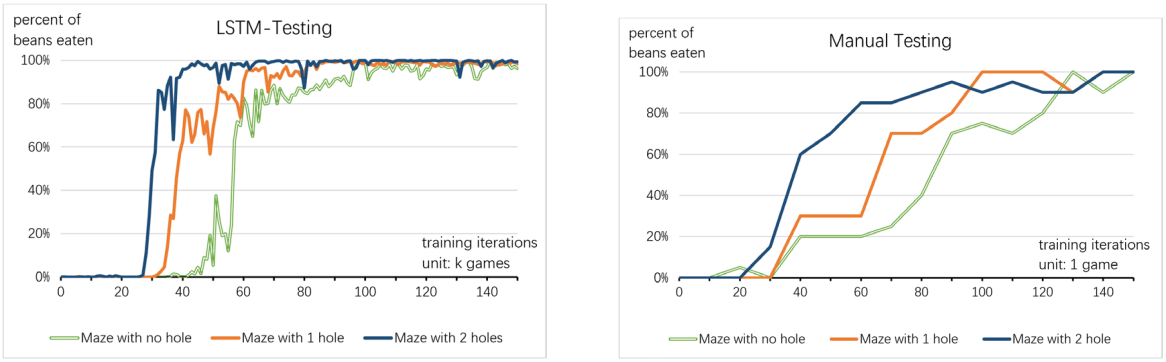
Fonte: Ariyurek *et al.* (2021)

Finalmente, Chen *et al.* (2020) afirmam que, embora a instalação de testes automatizados tenha um custo inicial elevado, no final, eles economizam recursos ao reduzir a necessidade de envolvimento humano constante (Figura 30). Isso permite que os testadores se concentrem em tarefas mais complicadas e criativas.

Por outro lado, também há desvantagens em automatizar os teste. Segundo Ariyurek *et al.* (2021), Chen *et al.* (2020), Holmgård *et al.* (2019), a implementação de testes automatizados requer um investimento inicial considerável em ferramentas, infraestrutura e treinamento. Isso pode ser um problema para equipes ou projetos com orçamentos pequenos.

Conforme Sestini *et al.* (2022), com o passar do tempo, os testes automatizados

Figura 30 – Resultados do LSTM-Testing e do teste manual em um jogo de labirinto



(a) Teste manual. (b) *LSTM-Testing*.

Fonte: Chen *et al.* (2020)

podem se deteriorar e, por isso, requerem manutenção contínua. Os scripts de teste podem ser danificados por alterações no código do jogo, o que significa que são necessárias atualizações regulares.

Por último, de acordo com Chen *et al.* (2020), embora testes automatizados sejam úteis para verificar cenários pré-definidos, eles têm limitações quando se trata de testes exploratórios, onde a criatividade e a intuição humanas são necessárias para encontrar novos *bugs*.

5.4 Como a inteligência artificial pode melhorar a eficácia e a eficiência dos testes automatizados em jogos digitais?

É fundamental distinguir entre eficiência e eficácia no teste de software, especialmente em jogos digitais. A eficácia se refere a capacidade do processo de teste de atingir os objetivos pretendidos, como detectar e corrigir erros e garantir que o software funcione conforme esperado. A eficiência, por outro lado, se refere ao uso otimizado dos recursos, como tempo e esforço, para realizar os testes. Assim, enquanto a eficácia se concentra nos resultados finais, a eficiência se refere ao processo e as técnicas que foram empregadas para atingir esses resultados.

Por meio de uma variedade de técnicas avançadas que aumentam a cobertura e a precisão dos testes, a inteligência artificial (IA) tem o potencial de aumentar significativamente a eficácia dos testes automatizados em jogos digitais. Primeiramente, a IA pode automatizar a criação de uma infinidade de cenários de teste que cobrem uma variedade de combinações de ações e estados do jogo (STAHLKE *et al.*, 2020). Quando se trata de situações raras ou complexas que seriam difíceis de prever manualmente, isso é particularmente útil para encontrar

bugs.

Outro aspecto importante para aumentar a eficácia é que a IA tem a capacidade de simular comportamentos complexos de jogadores, o que permite que os testes analisem uma variedade mais ampla de potenciais interações (GORDILLO *et al.*, 2021). Isso ajuda a garantir que o jogo funcione para vários estilos de jogo e preferências dos usuários.

Além disso, os algoritmos de aprendizado de máquina podem analisar os resultados dos testes e descobrir padrões nos dados de falhas (FERDOUS *et al.*, 2023). Isso ajuda a otimizar os casos de teste e priorizar as áreas do jogo mais importantes que precisam de mais atenção, melhorando a eficiência na fase de teste de software.

Essas abordagens não só aumentam a eficácia e a eficiência dos testes, mas também melhoram a experiência de jogo, tornando-a mais forte e sem falhas. Isso aumenta a satisfação do jogador e a qualidade do produto final.

5.5 Considerações finais

Neste capítulo, foram apresentadas as respostas às questões de pesquisa que orientaram este estudo, com base na análise dos artigos revisados. Os resultados mostraram que existem diversas técnicas e ferramentas para testar software em jogos digitais, cada uma com suas vantagens e suas desvantagens, que podem aumentar a eficácia e a eficiência de testes de software.

Apesar de possivelmente ter um custo alto de treinamento e implantação, a IA se destaca como uma ferramenta fundamental para melhorar a eficácia e a eficiência dos testes automatizados em jogos digitais, permitindo a criação de cenários de teste mais complexos e precisos, a simulação de comportamentos de jogadores e a análise de padrões nos dados de falhas. A incorporação da IA, em particular, da aprendizagem por reforço, nos testes automatizados pode revolucionar a forma como os jogos digitais são testados, tornando-os mais robustos e sem falhas, e melhorando a experiência do jogador. Portanto, é fundamental que os desenvolvedores de jogos digitais considerem a utilização da IA como uma ferramenta essencial para melhorar a qualidade e a eficiência dos testes de software.

6 CONCLUSÃO

6.1 Principais contribuições

A análise literária realizada sobre o tema dos testes de software em jogos digitais revela um profundo entendimento dos desafios e complexidades envolvidos nesse processo crucial no desenvolvimento de jogos. A revisão abrangeu diversas fontes, proporcionando uma visão completa das práticas atuais, metodologias e ferramentas utilizadas nessa área.

Os estudos analisados ressaltaram a importância crítica dos testes de software em jogos digitais não apenas para assegurar a qualidade técnica, mas também para melhorar a experiência do usuário. Desafios específicos como o constante avanço tecnológico e as demandas por lançamentos rápidos foram identificados como elementos que exigem abordagens inovadoras e adaptáveis nos processos de teste.

6.2 Trabalhos futuros

Com base na análise literária, alguns tópicos podem ser considerados para pesquisas futuras. O primeiro seria uma expansão da revisão, que pode ser expandida para incluir mais bases de dados. Isso pode revelar novas tendências e tecnologias emergentes no campo dos testes de software em jogos digitais.

Outra melhoria seria o desenvolvimento de novas ferramentas e técnicas. com o avanço contínuo da tecnologia (UNIVERSITY, 2021) e a crescente complexidade dos jogos (DEBRUSK, 2022), há uma necessidade de desenvolver novas ferramentas e técnicas de teste que possam lidar efetivamente com esses desafios.

Por último, seria interessante analisar testes de acessibilidade em jogos. À medida que a indústria de jogos se torna mais inclusiva (JERRETT, 2022; KOLAKOWSKI, 2023; MOREIRA, 2023), os testes de acessibilidade se tornam cada vez mais importantes para garantir que os jogos sejam acessíveis a todos os jogadores, independentemente de suas habilidades ou deficiências.

REFERÊNCIAS

ALBAGHAJATI, A.; AHMED, M. Video game automated testing approaches: An assessment framework. **IEEE Transactions on Games**, v. 15, n. 1, p. 81–94, 2023.

ARIYUREK, S.; BETIN-CAN, A.; SURER, E. Automated video game testing using synthetic and humanlike agents. **IEEE Transactions on Games**, v. 13, n. 1, p. 50–67, 2021.

BECK, K. **Test-Driven Development: By Example**. [*S. l.*]: Addison-Wesley, 2002.

BEIZER, B. **Software Testing Techniques**. 2. ed. [*S. l.*]: Van Nostrand Reinhold, 1990. ISBN 978-0442008703.

BERGDAHL, J.; GORDILLO, C.; TOLLMAR, K.; GISSLéN, L. Augmenting automated game testing with deep reinforcement learning. In: **2020 IEEE Conference on Games (CoG)**. [*S. l.*: *s. n.*], 2020. p. 600–603.

BRASIL, D. S. **Introdução ao Machine Learning**. 2020. https://dataat.github.io/introducao-ao-machine-learning/introdu%C3%A7%C3%A3o.html. Acesso em [data de acesso].

CHEN, L.-K.; CHEN, Y.-H.; CHANG, S.-F.; CHANG, S.-C. A long/short-term memory based automated testing model to quantitatively evaluate game desig. **Applied Sciences**, 2020.

CHI, H.; JONES, E. L. Computational investigations of quasirandom sequences in generating test cases for specification-based tests. In: **Proceedings of the 2006 Winter Simulation Conference**. [*S. l.*: *s. n.*], 2006. p. 975–980.

CHO, H.-C.; KIM, K.-J. Comparison of human and ai bots in starcraft with replay data mining. In: **2013 IEEE Conference on Computational Inteligence in Games (CIG)**. [*S. l.*: *s. n.*], 2013. p. 1–2.

DEBRUSK, C. **Technology Complexity And Its Impact On Innovation**. 2022. Disponível em: https://www.oliverwyman.com/our-expertise/insights/2022/jan/technology-complexity-and-its-impact-on-innovation.html.

FELDMEIER, P.; STRAUBINGER, P.; FRASER, G. Playtest: A gamified test generator for games. **ACM Digital Library**, Association for Computing Machinery, New York, NY, USA, p. 47–51, 2023. Disponível em: https://doi.org/10.1145/3617553.3617884.

FERDOUS, R.; HUNG, C. kang; KIFETEW, F.; PRANDI, D.; SUSI, A. Evombt: Evolutionary model based testing. **Science of Computer Programming**, v. 227, p. 102942, 2023. ISSN 0167-6423. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167642323000242.

FONTES, A.; GAY, G. **The Integration of Machine Learning into Automated Test Generation: A Systematic Mapping Study**. 2023. Disponível em: https://arxiv.org/abs/2206.10210.

GILB, T. **Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage**. [*S. l.*]: Elsevier, 2005.

GILLBERG, J.; BERGDAHL, J.; SESTINI, A.; EAKINS, A.; GISSLéN, L. Technical challenges of deploying reinforcement learning agents for game testing in aaa games. In: **2023 IEEE Conference on Games (CoG)**. [*S. l.*: *s. n.*], 2023. p. 1–8.

GOODFELLOW, I. J.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. **Generative Adversarial Networks**. 2014.

GORDILLO, C.; BERGDAHL, J.; TOLLMAR, K.; GISSLéN, L. Improving playtesting coverage via curiosity driven reinforcement learning agents. In: **2021 IEEE Conference on Games (CoG)**. [*S. l.*: *s. n.*], 2021. p. 1–8.

HOLMGåRD, C.; CERNY, M. G.; LIAPIS, A.; TOGELIUS, J. Automated playtesting with procedural personas through mcts with evolved heuristics. **IEEE Transactions on Games**, v. 11, n. 4, p. 352–362, 2019.

HSIEH, C.-Y.; TSAI, C.-H.; CHENG, Y. C. Test-duo: A framework for generating and executing automated acceptance tests from use cases. In: **2013 8th International Workshop on Automation of Software Test (AST)**. [*S. l.*: *s. n.*], 2013. p. 89–92.

JERRETT, A. **How 'GamerGate' led the gaming industry to embrace more diverse and caring values**. 2022. Disponível em: https://theconversation.com/how-gamergate-led-the-gaming-industry-to-embrace-more-diverse-and-caring-values-190068.

JUST, R.; JALALI, D.; ERNST, M. D. Defects4j: A database of existing bugs and test cases. In: IEEE. **2014 IEEE International Conference on Software Testing, Verification and Validation (ICST)**. [*S. l.*], 2014. p. 437–440.

KANER, C.; FALK, J.; NGUYEN, H. Q. **Testing Computer Software**. [*S. l.*]: Wiley, 2002.

KOLAKOWSKI, N. **GDC 2023 State of the Game Industry: Devs Weigh in on the Metaverse, Player Toxicity, and More**. 2023. Disponível em: https://www.dice.com/career-advice/has-the-video-game-industry-embraced-diversity-and-inclusion.

KONVOY VENTURES. **Gaming Industry Report - Q2 2024**. 2024. Acesso em: 14 ago. 2024. Disponível em: https://www.konvoy.vc/content/gaming-industry-report-q2-2024.

MACHADO, M. C.; BELLEMARE, M. G.; TALVITIE, E.; VENESS, J.; HAUSKNECHT, M.; BOWLING, M. **Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents**. 2017. Disponível em: https://arxiv.org/abs/1709.06009.

MCCONNELL, S. **Code Complete: A Practical Handbook of Software Construction**. 2. ed. [*S. l.*]: Microsoft Press, 2004.

MCGONIGAL, J. **Reality is Broken: Why Games Make Us Better and How They Can Change the World**. [*S. l.*]: Penguin Books, 2011.

MITCHELL, T. **Machine Learning**. [*S. l.*]: McGraw-Hill, 1997.

MOREIRA, C. **Increased number of women in a growing games industry**. 2023. Disponível em: https://www.womeningames.org/overcoming-challenges-and-building-a-more-inclusive-industry/.

MYERS, G. J. **The Art of Software Testing**. [*S. l.*]: John Wiley & Sons, 2011.

NIELSEN, J. **Designing Web Usability**. [*S. l.*]: New Riders Publishing, 2000.

PIGOSKI, T. M. **Practical Software Maintenance: Best Practices for Managing Your Software Investment**. [*S. l.*]: Wiley, 1997.

PINHEIRO, M. J. O. Desafio na utilização de técnicas de renderização não fotorrealistas com transferência de estilo com características do cartoon: uma revisão sistemática da literatura. 2023.

PRESSMAN, R. S. **Software Engineering: A Practitioner's Approach**. [*S. l.*]: McGraw-Hill, 2014.

RAMYA, P.; SINDHURA, V.; SAGAR, P. V. Testing using selenium web driver. In: **2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)**. [*S. l.*: *s. n.*], 2017. p. 1–7.

ROBERTSON, S.; ROBERTSON, J. **Mastering the Requirements Process: Getting Requirements Right**. 3ª ed.. ed. [*S. l.*]: Addison-Wesley, 2013.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. [*S. l.*]: Pearson Education, 2011.

SESTINI, A.; BERGDAHL, J.; TOLLMAR, K.; BAGDANOV, A. D.; GISSLéN, L. Towards informed design and validation assistance in computer games using imitation learning. In: **2023 IEEE Conference on Games (CoG)**. [*S. l.*: *s. n.*], 2023. p. 1–8.

SESTINI, A.; GISSLéN, L.; BERGDAHL, J.; TOLLMAR, K.; BAGDANOV, A. D. **CCPT: Automatic Gameplay Testing and Validation with Curiosity-Conditioned Proximal Trajectories**. 2022.

SOFIST. **Automação de testes: o que é, como fazer e quais os benefícios**. 2023. Disponível em: https://www.sofist.com.br/blog/automacao-de-testes-o-que-e-como-fazer-e-quais-os-beneficios/.

SOMMERVILLE, I. **Software Engineering**. 10th. ed. [*S. l.*]: Addison-Wesley, 2015.

STAHLKE, S.; NOVA, A.; MIRZA-BABAEI, P. Artificial players in the design process: Developing an automated testing tool for game level and world design. **ACM Digital Library**, Association for Computing Machinery, New York, NY, USA, p. 267–280, 2020. Disponível em: https://doi.org/10.1145/3410404.3414249.

UNIVERSITY, M. **Future of Video Games: Trends, Technology, and Types**. 2021. Disponível em: https://online.maryville.edu/blog/future-of-video-games/.

ZYDA, M. **Game Design: Concepts and Principles**. [*S. l.*]: Elsevier, 2016.