Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-25)
Demonstrations Track

Using Planning for Automated Testing of Video Games

Tomáš Balyo[1] , **Roman Barták**[1,2] , **Lukáš Chrpa**[1,3] , **Michal Červenka**[1] , **Filip Dvořák**[1] , **Stephan Gocht**[4] , **Lukáš Lipčák**[1] , **Viktor Macek**[1] , **Dominik Roháček**[1] , **Josef Ryzí**[1] , **Martin Suda**[1,3] , **Dominik Šafránek**[1] , **Slavomír Švancár**[1] , **G. Michael Youngblood**[1]
[1]Filuta AI, Inc., 1606 Headway Cir STE 9145, Austin, TX 78754, United States
[2]Faculty of Mathematics and Physics, Charles University, Prague, Czechia
[3]Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Czechia
[4]Stephan Gocht - AI Software Engineering, Meißen, Germany
tomas@filuta.ai, batak@ktiml.mff.cuni.cz, lukas.chrpa@cvut.cz, {mcervenka, filip}@filuta.ai, stephan@drgocht.com
{llipcak, viktor, drohacek, josef, msuda, dsafranek, slavo, michael}@filuta.ai

Abstract

In this demonstration, we present a system that automates regression testing for video games using automated planning techniques. Traditional test scripts are a common method for testing both video games and software in general. While effective, they require manual creation and frequent updates throughout development, making the process labor-intensive. Our system eliminates this burden by automatically generating and maintaining test scripts. The test engineer only needs to define the game's rules using the Planning Domain Definition Language (PDDL) and specify initial states and goals for individual test cases. This significantly reduces human effort while ensuring test scripts remain up to date. Additionally, our system integrates with game engine editors — supporting both Unity and Unreal to execute and evaluate test cases directly within the game. It collects detailed logs, telemetry data, and video recordings, allowing users to review test results efficiently.

1 Introduction

Game testing is crucial for identifying bugs, optimizing performance, and ensuring a high-quality user experience. It involves scenario testing, UI refinement, and compliance checks, especially for online games. While modern testing blends automation and manual efforts, fully integrating automation without disrupting development remains a challenge [Politowski *et al.*, 2022]. Our system expands automated testing with a planning-based approach.

We present an alternative to scripted testing. The most significant difference is that scripts, in conventional scripting, must be manually created and maintained for each test, whereas tests utilizing automated planning only require specifying the test goals. A plan can be used in the same way as test scripts in games. The planner creates an ad hoc path to the goal each time it is executed based on the game mechanics captured in the defined domain model. Furthermore, our system can react to unexpected events during the execution of a script and adjust it by re-planning from the current state and still possibly reaching the goal.

The presented system is being developed as a commercial product by Filuta AI, Inc. in collaboration with real game development studios as customers. It undergoes continuous evaluation and refinement based on their feedback to ensure it meets their needs.

2 Motivation

We focus on regression testing during game development, i.e., running frequent tests to ensure that recent changes did not introduce unintended glitches and the game still performs as expected. This is a particularly repetitive and dull task for a human tester but crucial for effective development. It is often the case that the longer the time difference between the introduction and discovery of the bug, the harder it is to fix.

2.1 What can be Tested with Planning

For regression testing, we can define a set of objectives that must be tested every time the game code is updated. These objectives can be translated into (planning) goals that need to be achieved by sequences of actions (plans) that can be generated automatically (by planners). The plans are then executed in the game in a similar fashion as (manually written) test scripts. On top of that, we can collect game performance telemetry data.

Besides regression testing, planning has the potential for investigating whether the game can be completed, i.e., each intended objective can be achieved from the starting state of each scenario, if the game has dead-ends, i.e., states that can be reached but it is impossible to complete the game from them. In modern game design, such situations are very undesirable and annoying for gamers. Additionally, planning can be also used to identify undesired shortcuts in the game.

The Planning Agent technology is a great fit for games that have random and non-deterministic nature due to its ability to adapt to changing conditions and the availability of re-planning (changing the game script) at any point. Planning

agents also inherently allow the automatic variation of the playthroughs by weighing the decision points (preferring the unexplored paths) so that the agent finds plans that are as different as possible from all the previous playthroughs while still achieving the goal. This principle is like Monte-Carlo Sampling, providing coverage of diverse gameplays even in infinitely large game worlds, where the only limitation is the number of hours budgeted for the explorative agents. This adaptability allows us to test objectives such as:

- Dialogue Choices / NPC[1] Interactions: Assuming the tree of narrative decisions is finite, the Planning Agents are expected to provide complete coverage of the narrative tree and capture anomalies such as suddenly inaccessible dialogue paths or unreachable story goals.
- Unscripted Event Responses: This can be a side effect of any other tests or a test by itself. Re-planning based on a changed world state is an integral part of Planning Agents and happens seamlessly. We can either detect deviations or re-plannings in all the other tests that can be used for this analysis, or we can design a test that covers a part of the game that changes dramatically and lets the Planning Agent reach a goal. We also measure the cognitive complexity of the dynamic changes and gameplay (how difficult the reasoning task about the dynamic change has been).

2.2 Applicability

The classical planning-based testing approach is most suitable for games with many discrete causal interactions such as role-playing games (RPG), point-and-click adventures, strategy or city-building games, visual novels, or puzzles. It is less suitable for fast-paced action-oriented games such as sports games or driving simulations. For such games, reinforcement learning-based approaches, that have been studied in recent works, seem to be more suitable (see the Related Work Section of this paper). Overall, we observe that the reinforcement learning and the planning-based approaches can complement each other rather well.

3 Preliminaries

Automated planning concerns finding a plan — a sequence of actions that transform the world from an initial state to a goal state, i.e., a state where all goal conditions are satisfied. A planning problem instance consists of a domain definition and a task definition. The domain definition describes the environment and possible actions with their preconditions and effects. In STRIPS planning [Fikes and Nilsson, 1971], the environment is specified by predicates, and preconditions and effects of actions are conjunctions of predicates and their negations. The task specification consists of the description of the initial state and the goal conditions.

4 Related Work

Planning is rooted in search and, as such, has had a long history with games. At the beginning of AI, the focus was on games like chess [Newell *et al.*, 1972] and checkers [Samuel, 1959], which initially relied on the search for solutions. Planning continued to dominate in the 80s and 90s with checkers and chess with Deep Blue [Hsu *et al.*, 1990] and Chinook [Schaeffer *et al.*, 1996]. Agent architectures and production systems added value, and soon, planning started to add value in games like bridge [Smith *et al.*, 1998] and the class of Real-Time Strategy (RTS) games [Chung *et al.*, 2005]. In [Duarte *et al.*, 2020], they survey the history of planning and learning in games, covering the spectrum as well as diving into the lineage of planning from search, minimax and alpha-beta pruning, hierarchical task networks, and Monte Carlo Tree search, through classical planning, rapidly-exploring random trees, case-based planning, and behavior trees. Most of the work is focused on creating AI-driven opponents [Wurman *et al.*, 2022], which are sometimes used to play both sides for evaluation, AI training, and testing.

Automated testing with AI has been a rising research focus more recently with work that has focused on agent-based approaches that include navigation mesh path-finding [Shirzadehhajimahmood *et al.*, 2021], reinforcement learning agents for finding design and environmental defects [Ariyurek *et al.*, 2019; Ferdous *et al.*, 2022], reinforcement learning for load testing [Tufano *et al.*, 2022], modeling of user interaction for boundary testing [Owen *et al.*, 2016], search for test case generation [Ferdous *et al.*, 2021a], and search for automated play testing [Ferdous *et al.*, 2021b]. Despite planning uses in-game AI, we do not see its use in in-game testing more broadly beyond back to search. However, as evidenced by Bram Ridder's (AI Programmer for Rebellion) keynote talk at the 2021 AIIDE Conference on "Improved Automated Game Testing Using Domain-Independent AI Planning" [Riddler, 2021] and his 2021 GDC AI Summit talk "Automated Game Testing Using a Numeric Domain Independent AI Planner," planning techniques for game testing are beginning to be used in the games industry mixed in with calls for more AI automation of testing [Fray, 2023].

In [Volokh and Halfond, 2023] the authors proposed an automated approach for determining actions when conducting automated exploration for games. It is based on program analysis (slicing) of the game code. Although they are not using the usual planning formalism (like PDDL), they work with symbolic representation of states and actions and rely on SMT (satisfiability modulo theories) solvers to determine the set of applicable actions in a given state.

5 System Description

A *Planning Agent* is an entity that generates a plan to reach the test goal, executes the plan, and then reports the outcome of the plan execution (e.g. passed or failed) and provides telemetry data. A planning agent requires a domain model that captures the mechanics of the game and a description of the current game state and a goal that needs to be achieved. The domain model is provided in Planning Domain Definition Language (PDDL) [Ghallab *et al.*, 1998], while the initial and goal states are described using predicates. The planning agent utilizes the Unified Planning framework [Micheli *et al.*, 2025] to interface with state-of-the-art planners and find plans.

[1]Non-player character

Planning Agents are engine-agnostic and can be integrated with any game engine. Our most advanced integration is with Unity and Unreal engines. For both we developed a unique Filuta Plugin that automatically generates the data required by the planning agent. Each plugin can hook onto the Unity and Unreal objects to automatically collect information about the game environment. Each plugin also allows the Planning Agent's output to be integrated back into the game to play it. Based on the environment configuration, it sets the game up into an initial state, creates a plan to fulfill the (user-specified) test goal and executes it. In most tests, the only configuration the user must specify is the environment (map) on which the test will be executed. Everything is configured and viewed on a web application called Planning Dashboard. It is also used by the user to inspect the results of the tests.

Filuta AI allows for unattended testing even in non-deterministic environments. In case of any unforeseen circumstances in a game, such as the targeted item is collected by a different game character, the Planning Agent can automatically adjust, i.e., the Planning Agent can stop the execution, evaluate the current world state, and produce a new plan to reach the goal.

5.1 Instrument Once, Test All Code Versions

The most labor-intensive part of integrating the Filuta AI Gaming QA solution is mapping the actions from the domain (move, kill, collect, . . .) back to the game code. Therefore we designed our plugins in a way that the game code itself does not need to be modified, and new code is only added on top of it. The plugins use reflection techniques to extract game state information. This allows the mapping needs to be developed only once, and unless there are significant changes to the game mechanics, it can remain the same throughout the whole game development process.

The original instrumentation of the real-time strategy game Silica [Bohemia Interactive, 2025] happened on a code version from September 2023. One of the initial groups of tests was to build all the available unit types in the game. The version from September 2023, however, did not implement some of the unit-building structures, so a player could not build some units yet, hence these goals were unachievable. When updating the code to a version from April 2024 (after 7 months of active development), no change was required to the Filuta AI instrumentation. All the tests that worked on the September 2023 version also work on the latest version without any amendments. Moreover, the structures to build the new units were implemented into the game and the Filuta Unity Plugin automatically registered them.

5.2 Addressing Hardness of PDDL Modeling

Automated Planning is not commonly utilized by game developers. A major challenge in its adoption lies in the authorial burden of writing domain and problem descriptions in PDDL [Ghallab *et al.*, 1998]. Crafting well-structured PDDL models is a specialized skill that develops over time, making efficient use of experienced modelers crucial. While Knowledge Engineering tools exist to support PDDL modeling [Simpson *et al.*, 2007; Vaquero *et al.*, 2013], the process remains somewhat of a "black art" [McCluskey *et al.*, 2017].

To address this challenge, we investigate the practical applicability of Automated Planning Action Model Learning (AML) techniques. AML aims to synthesize domain definitions directly from logs containing state sequences, reducing the need for manual modeling. Numerous algorithms have been developed for this task [Wang, 1996; Aineto *et al.*, 2019; Zhuo *et al.*, 2010; Yang *et al.*, 2007; Juba *et al.*, 2021], and the MacQ project [Callanan *et al.*, 2022] provides a recent overview along with new implementations of multiple AML techniques. Our research team is actively contributing to this field with ongoing work [Balyo *et al.*, 2024a; Balyo *et al.*, 2024b].

6 Demo Video

The demo video (available at https://youtu.be/ggRiXeQxMsE) showcases our system in action with two real game examples. The first is a real-time strategy game Silica by Bohemia Interactive [Bohemia Interactive, 2025] developed in the Unity Engine [Unity Technologies, 2025] and the second is the first-person shooter Lyra [Epic Games, 2025a]. Lyra is a sample tutorial project developed alongside Unreal Engine [Epic Games, 2025b] to serve as a starting point for learning about the engine and creating new games in it. The video features gameplay footage from the planning agent, along with behind-the-scenes insights into test setup and evaluation using our web application, the Filuta Planning Dashboard.

7 Conclusion

We introduced the Filuta AI planning agent, which automates regression testing for video games using PDDL-based automated planning. This approach eliminates the need for manual test scripts, reducing workload while maintaining accuracy and adaptability. Integrated with Unity and Unreal, it enables seamless test execution and detailed result analysis.

Our evaluation across multiple games shows that planning-based testing effectively verifies gameplay mechanics and objectives. Its ability to adapt and replan in dynamic environments enhances robustness.

However, challenges remain, particularly in the complexity of PDDL modeling. To address this, we explored action model learning (AML) to generate domain models from gameplay logs, making automated planning more accessible. Our research continues to refine these techniques and streamline integration.

7.1 Future Work

Moving forward, we aim to extend our system's applicability by testing it across a broader range of game genres. We're working closely with several video game studios to meet their real-world needs. Additionally, we plan to develop intuitive tools for domain modeling, such as graphical or no-code interfaces inspired by Unreal Engine's Blueprint system, to further lower the barrier for game developers. By enhancing usability and expanding coverage, we hope to drive wider adoption of automated planning in game testing, ultimately improving game quality and development efficiency.

Acknowledgements

Roman Barták and Lukáš Chrpa are supported by the project 25-18003S of the Czech Science Foundation.

References

[Aineto *et al.*, 2019] Diego Aineto, Sergio Jiménez Celorrio, and Eva Onaindia. Learning action models with minimal observability. *Artificial Intelligence*, 275:104–137, 2019.

[Ariyurek *et al.*, 2019] Sinan Ariyurek, Aysu Betin-Can, and Elif Surer. Automated video game testing using synthetic and humanlike agents. *IEEE Transactions on Games*, 13(1):50–67, 2019.

[Balyo *et al.*, 2024a] Tomáš Balyo, Martin Suda, Lukáš Chrpa, Dominik Šafránek, Stephan Gocht, Filip Dvořák, Roman Barták, and G Michael Youngblood. Planning domain model acquisition from state traces without action parameters. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 21, pages 812–822, 2024.

[Balyo *et al.*, 2024b] Tomáš Balyo, G Michael Youngblood, Filip Dvořák, and Roman Barták. On automating video game testing by planning and learning. *arXiv preprint arXiv:2402.12393*, 2024.

[Bohemia Interactive, 2025] Bohemia Interactive. Silica game. https://silicagame.com/, 2025. Accessed: 2025-05-15.

[Callanan *et al.*, 2022] Ethan Callanan, Rebecca De Venezia, Victoria Armstrong, Alison Paredes, Tathagata Chakraborti, and Christian Muise. Macq: a holistic view of model acquisition techniques. *arXiv preprint arXiv:2206.06530*, 2022.

[Chung *et al.*, 2005] Michael Chung, Michael Buro, and Jonathan Schaeffer. Monte carlo planning in rts games. In *Proceedings of the 2005 IEEE Symposium on Computational Intelligence and Games (CIG05)*. IEEE, 01 2005.

[Duarte *et al.*, 2020] Fernando Fradique Duarte, Nuno Lau, Artur Pereira, and Luis Paulo Reis. A survey of planning and learning in games. *Applied Sciences*, 10(13):4529, 2020.

[Epic Games, 2025a] Epic Games. Lyra starter game - unreal engine marketplace. https://www.unrealengine.com/marketplace/learn/lyra, 2025. Accessed: 2025-05-15.

[Epic Games, 2025b] Epic Games. Unreal engine. https://www.unrealengine.com/en-US, 2025. Accessed: 2025-05-15.

[Ferdous *et al.*, 2021a] Raihana Ferdous, Fitsum Kifetew, Davide Prandi, I. S. W. B. Prasetya, Samira Shirzadehhajimahmood, and Angelo Susi. Search-based automated play testing of computer games: A model-based approach. In *Search-Based Software Engineering: 13th International Symposium, SSBSE 2021, Bari, Italy, October 11–12, 2021, Proceedings*, page 56–71, Berlin, Heidelberg, 2021. Springer-Verlag.

[Ferdous *et al.*, 2021b] Raihana Ferdous, Fitsum Kifetew, Davide Prandi, ISWB Prasetya, Samira Shirzadehhajimahmood, and Angelo Susi. Search-based automated play testing of computer games: A model-based approach. In *International Symposium on Search Based Software Engineering*, pages 56–71. Springer, 2021.

[Ferdous *et al.*, 2022] Raihana Ferdous, Fitsum Kifetew, Davide Prandi, and Angelo Susi. Towards agent-based testing of 3d games using reinforcement learning. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–8, 2022.

[Fikes and Nilsson, 1971] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[Fray, 2023] Andrew Fray. Automated testing roundtables gdc 2023. https://autotestingroundtable.com/, 2023. (Accessed on 12/12/2023).

[Ghallab *et al.*, 1998] M. Ghallab, A. Howe, C. Knoblock, D. Mcdermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL—The Planning Domain Definition Language. 1998.

[Hsu *et al.*, 1990] F-h Hsu, Thomas S Anantharaman, Murray S Campbell, and Andreas Nowatzyk. Deep thought. In *Computers, Chess, and Cognition*, pages 55–78. Springer, 1990.

[Juba *et al.*, 2021] Brendan Juba, Hai S. Le, and Roni Stern. Safe Learning of Lifted Action Models. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, pages 379–389, 11 2021.

[McCluskey *et al.*, 2017] Thomas Leo McCluskey, Tiago Stegun Vaquero, and Mauro Vallati. Engineering knowledge for automated planning: Towards a notion of quality. In Óscar Corcho, Krzysztof Janowicz, Giuseppe Rizzo, Ilaria Tiddi, and Daniel Garijo, editors, *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*, pages 14:1–14:8. ACM, 2017.

[Micheli *et al.*, 2025] Andrea Micheli, Arthur Bit-Monnot, Gabriele Röger, Enrico Scala, Alessandro Valentini, Luca Framba, Alberto Rovetta, Alessandro Trapasso, Luigi Bonassi, Alfonso Emilio Gerevini, Luca Iocchi, Felix Ingrand, Uwe Köckemann, Fabio Patrizi, Alessandro Saetti, Ivan Serina, and Sebastian Stock. Unified planning: Modeling, manipulating and solving ai planning problems in python. *SoftwareX*, 29:102012, 2025.

[Newell *et al.*, 1972] Allen Newell, Herbert Alexander Simon, et al. *Human problem solving*, volume 104:9. Prentice-hall Englewood Cliffs, NJ, 1972.

[Owen *et al.*, 2016] V Elizabeth Owen, Gabriella Anton, and Ryan Baker. Modeling user exploration and boundary testing in digital learning games. In *Proceedings of the 2016 conference on user modeling adaptation and personalization*, pages 301–302, 2016.

[Politowski *et al.*, 2022] Cristiano Politowski, Yann-Gaël Guéhéneuc, and Fabio Petrillo. Towards automated video game testing: still a long way to go. In *Proceedings of the 6th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation*, pages 37–43, 2022.

[Riddler, 2021] Bram Riddler. Improve automated game testing using domain independent ai planning - youtube. https://www.youtube.com/watch?v=2KXmxuCjjCw, 2021. (Accessed on 12/12/2023).

[Samuel, 1959] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[Schaeffer *et al.*, 1996] Jonathan Schaeffer, Robert Lake, Paul Lu, and Martin Bryant. Chinook the world man-machine checkers champion. *AI magazine*, 17(1):21–21, 1996.

[Shirzadehhajimahmood *et al.*, 2021] Samira Shirzadehhajimahmood, ISWB Prasetya, Frank Dignum, Mehdi Dastani, and Gabriele Keller. Using an agent-based approach for robust automated testing of computer games. In *Proceedings of the 12th International Workshop on Automating TEST Case Design, Selection, and Evaluation*, pages 1–8, 2021.

[Simpson *et al.*, 2007] Ron M. Simpson, Diane E. Kitchin, and Thomas Leo McCluskey. Planning domain definition using GIPO. *Knowl. Eng. Rev.*, 22(2):117–134, 2007.

[Smith *et al.*, 1998] Stephen J Smith, Dana Nau, and Tom Throop. Computer bridge: A big win for ai planning. *AI magazine*, 19(2):93–93, 1998.

[Tufano *et al.*, 2022] Rosalia Tufano, Simone Scalabrino, Luca Pascarella, Emad Aghajani, Rocco Oliveto, and Gabriele Bavota. Using reinforcement learning for load testing of video games. In *Proceedings of the 44th International Conference on Software Engineering*, pages 2303–2314, 2022.

[Unity Technologies, 2025] Unity Technologies. Unity game engine. https://unity.com/, 2025. Accessed: 2025-05-15.

[Vaquero *et al.*, 2013] Tiago Stegun Vaquero, José Reinaldo Silva, Flavio Tonidandel, and J. Christopher Beck. itsimple: towards an integrated design system for real planning applications. *Knowl. Eng. Rev.*, 28(2):215–230, 2013.

[Volokh and Halfond, 2023] Sasha Volokh and William G.J. Halfond. Automatically defining game action spaces for exploration using program analysis. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 19(1):145–154, Oct. 2023.

[Wang, 1996] Xuemei Wang. *Learning planning operators by observation and practice*. PhD thesis, Citeseer, 1996.

[Wurman *et al.*, 2022] Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion gran turismo drivers with deep reinforcement learning. *Nat.*, 602(7896):223–228, 2022.

[Yang *et al.*, 2007] Qiang Yang, Kangheng Wu, and Yunfei Jiang. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence*, 171(2-3):107–143, 2007.

[Zhuo *et al.*, 2010] Hankz Hankui Zhuo, Qiang Yang, Derek Hao Hu, and Lei Li. Learning complex action models with quantifiers and logical implications. *Artificial Intelligence*, 174(18):1540–1569, 2010.