International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Development of Autonomous NPC Worker Agents through a Modular GOAP Architecture in 3D Simulations

June Aurelius B. Jacinto[1], Mark Kevin S. Sanig[2], Sherillen Gavino Namuag[3], Edgardo D. Olmoguez II[4]

[1,2,3] Student, Jose Rizal Memorial State University Main Campus, Philippines
[4] Instructor, Jose Rizal Memorial State University Main Campus, Philippines

Abstract
The study investigates the optimization of autonomous agent labor cycles within 3D interactive simulations, addressing the persistent boring manual NPC management. The study proposes a modular architecture for Goal-Oriented Action Planning (GOAP) specifically designed to handle and automate constant directives to worker-class entities. Developed in Unity 3D, the framework uses a dependency-layered planner that treats environmental events and prerequisites as dynamic variables. By evaluating both global world states and internal agent needs, the system enables NPCs to autonomously prioritize and sequence complex tasks from resource acquisition and structural construction to base defense. Unlike traditional Finite State Machine (FSM) models, this modular approach separates high-level goals from procedural actions, enabling task shifting and interruptions. The empirical evaluation results from fifteen respondents, using a five-point Likert scale, scored the system's effectiveness with a functionality mean of 4.42, a reliability mean of 4.25, and an efficiency mean of 4.30, resulting in a cumulative weighted mean of 4.32. These findings confirm that integrating dependency-based logic into a modular GOAP framework enhances NPC agency and decision-making accuracy. The research provides a scalable technical blueprint for developers to implement self-sufficient autonomous agents, elevating the experience of 3D simulations while effectively reducing the cognitive load on players during complex simulation scenarios.

Keywords: Goal-oriented Action Planning, Game Artificial Intelligence, Autonomous Agents, Non-player Characters, Unity 3d

1. Introduction

The development of modern 3D simulations represents a multidisciplinary convergence of programming, art, structural mechanics, and narrative design. Within these complex interactive ecosystems, non-player Characters (NPCs) serve as fundamental elements across nearly all genres and are essential tools for atmosphere, realism, and world design. Whether acting as adversaries that challenge the user or as allies providing strategic support, NPCs are the primary agents through which a virtual environment achieves a sense of agency. The integration of artificial intelligence allows simulations to provide superior experiences by creating life-like situational developments that hold engagement through increasing complexity (Miyake, 2017). However, the implementation of such systems is rarely uniform; it requires a

profound understanding of specific algorithmic foundations specific to the unique requirements of the environment.

Despite the importance of AI in interactive media, current development tools often prioritize one-off implementations instead of scalable, generalized frameworks. This approach fails to leverage the inherent similarities between different simulation types, which lead to difficulties when handling complicated environments independently (Addoum et al., 2021). Furthermore, designing realistic NPCs that can automatically produce engaging experiences still remains a technical bottleneck. Traditional AI applications struggle with a lack of context outside of their training data, which can result in predictable or immersion-breaking behaviors (Feng & Tan, 2016). These limitations lead to a transition toward hybrid methodologies where foundational algorithms, such as Goal-Oriented Action Planning (GOAP), are modified and combined with awareness and environmental state and logic to produce more intelligent, autonomous agents (Kopel & Hajas, 2018).

The research addresses the "babysitting" problem in design, where users must provide repetitive, manual directives to worker-class NPCs. The objective is to determine whether an enhanced, modular GOAP model utilizing a novel Goal Dependency Theory can satisfy the demands of functionality, reliability, and efficiency in a 3D virtual space. The system architecture is implemented using Unity 3D, C#, and Blender 3D, following a seven-stage Software Development Life Cycle (SDLC) that ranges from data gathering to iterative documentation. By undergoing the decision-making process for NPCs through dependency-layered planning (Vlachopoulos et al., 2014), this paper provides a conceptual and practical rationale for autonomous agent design that handles high-level intelligence and hardware performance constraints of modern 3D engines.

2. Literature Review

The world of games has been in circles of interest for different researchers because the application of intelligence machine learning techniques is put in games with the aim of providing an entertaining and satisfying gaming experience for the human player. Ranjitha et al. (2020) compared different AI techniques applied in gaming environments to algorithms employing artificial neural networks. In algorithm-driven AI games, the algorithms used were minimax search tree and Monte Carlo Tree Search, while adaptive techniques used the Artificial Neural Network learning methods to determine moves and strategies. In digital games, artificial intelligence (AI) is used in terms of generating responsive, adaptive, or intelligent behaviors for the non-player characters (NPCs) similar to human-like intelligence. This also connects to the work of Miyake (2017), who explains that a video game character's AI, known as Character AI, must work together with Navigation AI and Meta AI to create a unified system that creates a dynamic user experience in expansive virtual 2D/3D worlds.

Planning video game algorithms for NPC behavior is a complex problem, as most solutions depend on hard-coded components. While various AI approaches can implement realistic NPCs, Pihlgren et al. (2016) explains that even in "triple A" titles, players encounter unrealistic behaviors because NPCs lack fundamental cognitive skills like memory. Some researchers focus on implementing AI for NPCs in 3D games using decision trees and hybrid methods (Kopel & Hajas, 2018), while others explore autonomous behavior learning (Feng & Tan, 2016). Despite this progress, FSM implementation remains the preferred

first choice for developers as it is the easiest to get running and provides a solid base to assess results (Pihlgren et al., 2016).

To achieve plausible human-like behavior, model-aided NPCs are applied to facilitate emergent behaviors and enhance behavioral animation. This involves human-like functionalities such as memory retention so the NPC senses the game environment and flawlessly exhibits more human-like playing performances (Perrie & Li, 2014). Furthermore, the industry is seeing a shift toward planning models that can handle complex goals. Vlachopoulos et al. (2014) investigated how STRIPS planning techniques can be used to enhance the behavior of worker units that are common in real-time strategy (RTS) games. By giving the human player, the capability of instructing the worker unit to achieve simple goals, "Smart Workers" are introduced into the planning domain to increase interaction. This is supported by the goal of finding a flexible way to define character behavior easily using architectures like GOAP (Long, 2007).

However, multi-behavior NPCs often have intelligence for selecting behavior themselves which sometimes is not appropriate to accomplish a team objective. To solve this problem, agents use combination methods between state machines and fuzzy coordinators to make decisions, especially in RTS games where coordination between team members is vital (Akbar et al., 2015). The fuzzy coordinator analyzes remaining time, NPC health, and the enemy to decide which NPC has to offense or defense in battle. Research into automated generation of diverse NPC-controlling FSMs using nondeterministic planning helps close the gap in creating credible behaviors (Coman & Munoz-Avila, 2021). While some models focus on drive-based utility maximization (Sloan, 2015) or decision-tree complexities (Mozo et al., 2013), the integration of these theories as discussed by Yashchenko (2014) provides the foundation for artificial personalities that recognize their environment and make dynamic decisions.

3. Theoretical Background

The theoretical foundation of this research is grounded in the principles of intelligent system creation, specifically the multidimensional organization of artificial intelligence (AI) consciousness and personality. As discussed by Yashchenko (2014), the general theory of AI encompasses the study of neural-like growing networks, memory systems, and the functional organization of the brain including consciousness and subconsciousness developed through training and education. In the context of game development, these concepts are often aligned to accommodate genres and hardware constraints. This study states that the most effective AI results are achieved through a hybrid methodology, combining diverse algorithms to balance intelligence with technical complexity.

Building upon these principles, this study is anchored in the findings of Long (2007) regarding the enhancement of non-player character (NPC) behavior through Goal-Oriented Action Planning (GOAP). The successful implementation of GOAP in contemporary titles, such as F.E.A.R. and S.T.A.L.K.E.R., demonstrates that real-time action planning significantly improves character behavior compared to traditional Finite State Machine (FSM) architectures. From a technical perspective, GOAP is superior due to its efficient memory management and CPU usage. Furthermore, it offers greater flexibility and reliability in managing complex agent states.

A central component of this framework is the dynamic evaluation of Goal Variables and World States. In this model, goals represent potential actions, while NPC and environmental states serve as live inputs used to calculate the weight of a current objective relative to all others. The GOAP planner acts as a heuristic traverser, identifying the least-cost path of actions to achieve a goal. This process allows for a dynamic approach that adds flexibility and realism to the simulation. By constantly looping to evaluate environmental changes or user inputs, the system ensures that NPCs commit to the most effective actions while maintaining a balance between behavioral intelligence and system performance factors such as graphical rendering and audio processing.

4. Methods

The implementation of intelligent NPC worker models is designed to be modular to facilitate their decision-making. The system uses an improved Goal-Oriented Action Planning (GOAP) algorithm, where agents pick from a list of possible goals based on a dynamic weighting system. As shown in the system diagram (Figure 1), each goal is given a base weight that is constantly modified by real-time events from the World and NPC states. The total weight determines the agent's priority, allowing the GOAP Planner to execute the most efficient sequence of actions. Because the environmental and agent states are always shifting, a dynamic approach is used, ensuring the NPC can pivot between tasks without manual player intervention.

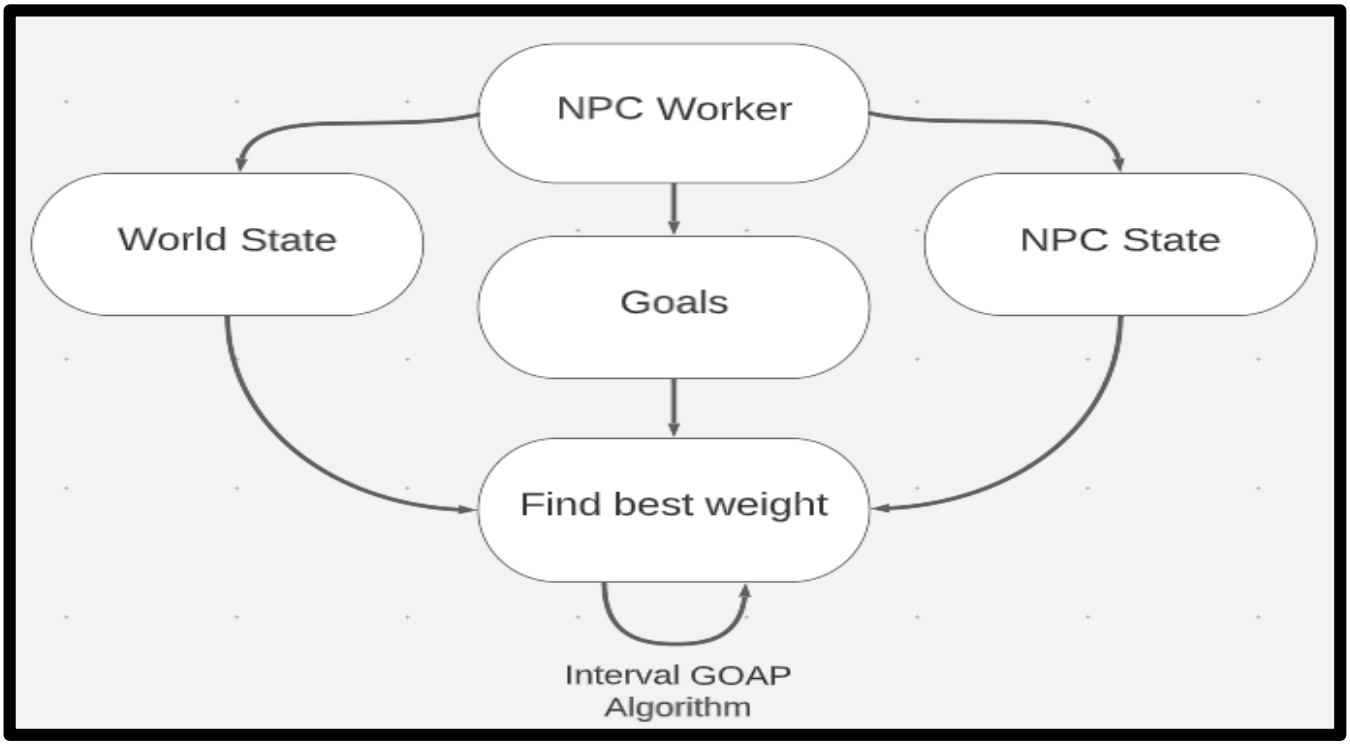

Figure 1. System Diagram

The Hierarchical Input Output Process (HIPO) diagram (Figure 2) shows the agent's decision-making ability. This involves a continuous feedback loop where self-awareness and world-awareness sections serve as sensory inputs and dynamic weight modifiers, constantly adjusting priorities based on real-time events such as resource scarcity or environmental threats. Specifically, the awareness section evaluates the active goal to return a normalized final weight, which the GOAP algorithm compares against the ultimate

goal set to identify the path of highest and lowest computational cost. Once the optimal goal is selected, the Planner creates an ordered list of actions that the agent traverses through; if a prerequisite state changes during execution, the system can reselect a sub-goal to resolve the dependency. A goal is only marked as "Complete" once this full traversal is executed without interruption, keeping the NPC worker models maintain high agency and adaptability as the 3D simulation environment evolves.

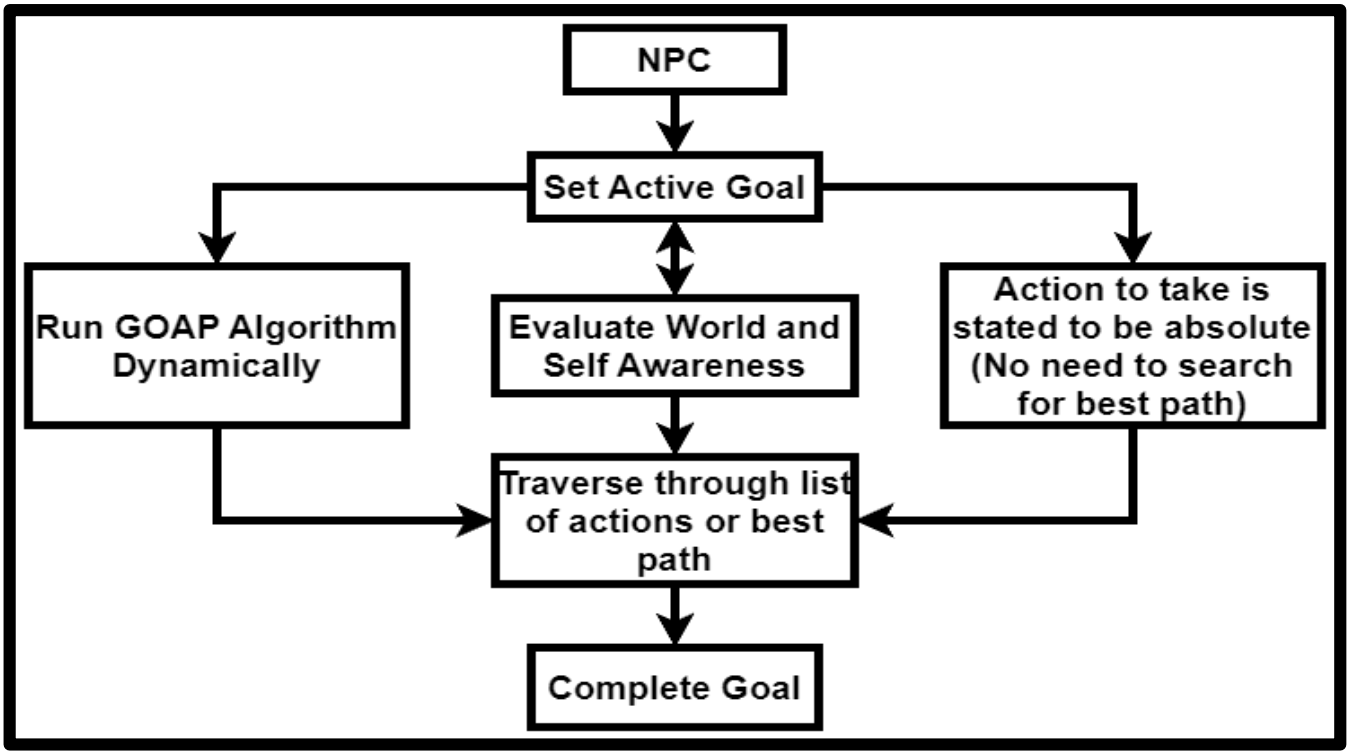

Figure 2. Hierarchical Input Process Output Diagram

The Class Diagram (Figure 3) shows how the NPC Worker object is dependent on the GOAPController, a centralized component that handles the agent's functional limits through an attribute known as traversalIntensity. This attribute defines the recursive depth of the planning process. The GOAPController serves as the manager for all potential objectives/targets, while the Planner component evaluates these goals by performing a comparison of modified weights. These weights are influenced by the WorldAwareness and NPCAwareness states, which map environmental data into numerical modifiers.

The CharacterAction is an abstract base that facilitates a variety of concrete child-class implementations such as gathering, defending, or patrolling without requiring modifications to the core planner logic. Each derived action defines its own unique baseWeight and PerformAction() algorithm, providing a clear separation of concerns between high-level planning and low-level execution. Furthermore, the GetWeight() method ensures that real-time environmental modifiers are accurately retrieved into the final logic. This structure allows the system to remain highly extensible, allowing developers to introduce new NPC behaviors simply by inheriting from the base class, thereby ensuring the framework's long-term manageability and technical efficiency.

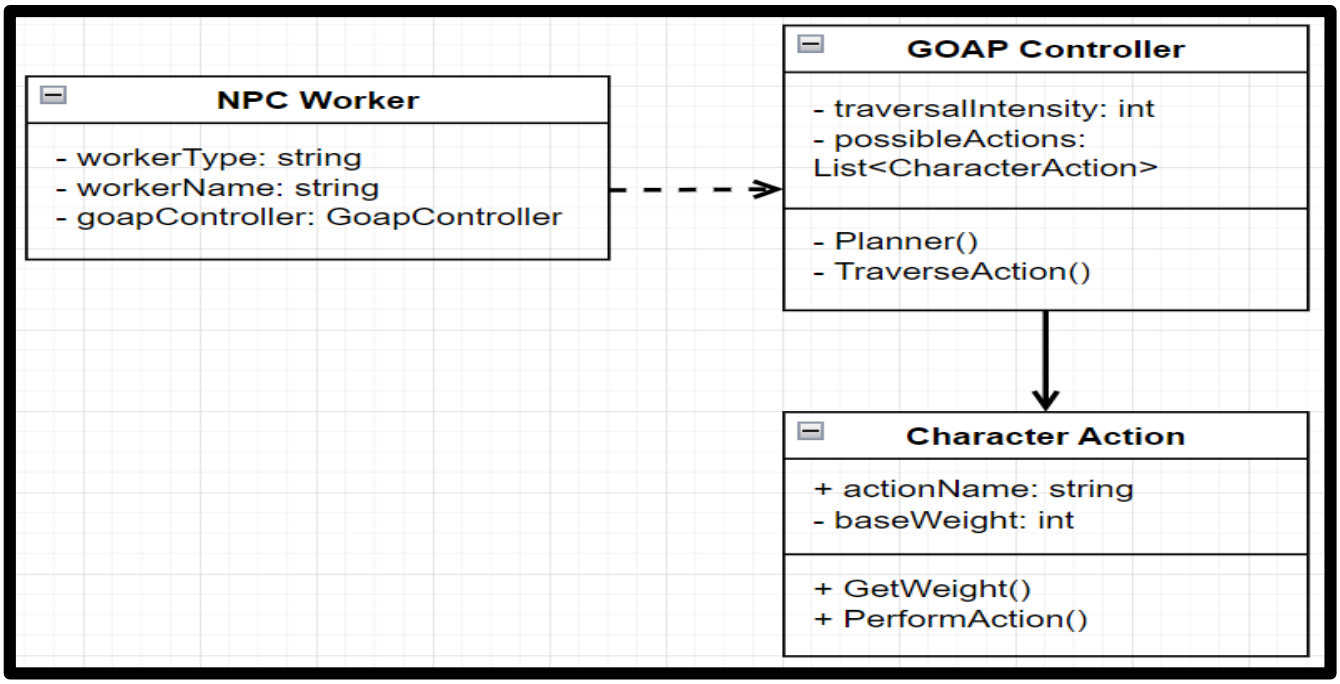

Figure 3. Class Diagram

The flow and interactions are mapped via a Sequence Diagram (Figure 4) and an Activity Diagram (Figure 5). The interaction begins when an NPC Worker receives a goal either through automatic awareness or manual player input. This goal is processed by the Awareness section to retrieve relevant functions and weights. The GOAP algorithm uses these data points to find the optimal path, which the NPC Worker then requests and executes. Furthermore, the Activity Diagram distinguishes the responsibilities between the Sender/Executor (the NPC), the Evaluator (Awareness section), and the Core Planner (GOAP Algorithm). This ensures that the NPC can efficiently request specific awareness weights for simple tasks or flags without always triggering the full complexity of the GOAP planning cycle, optimizing system performance.

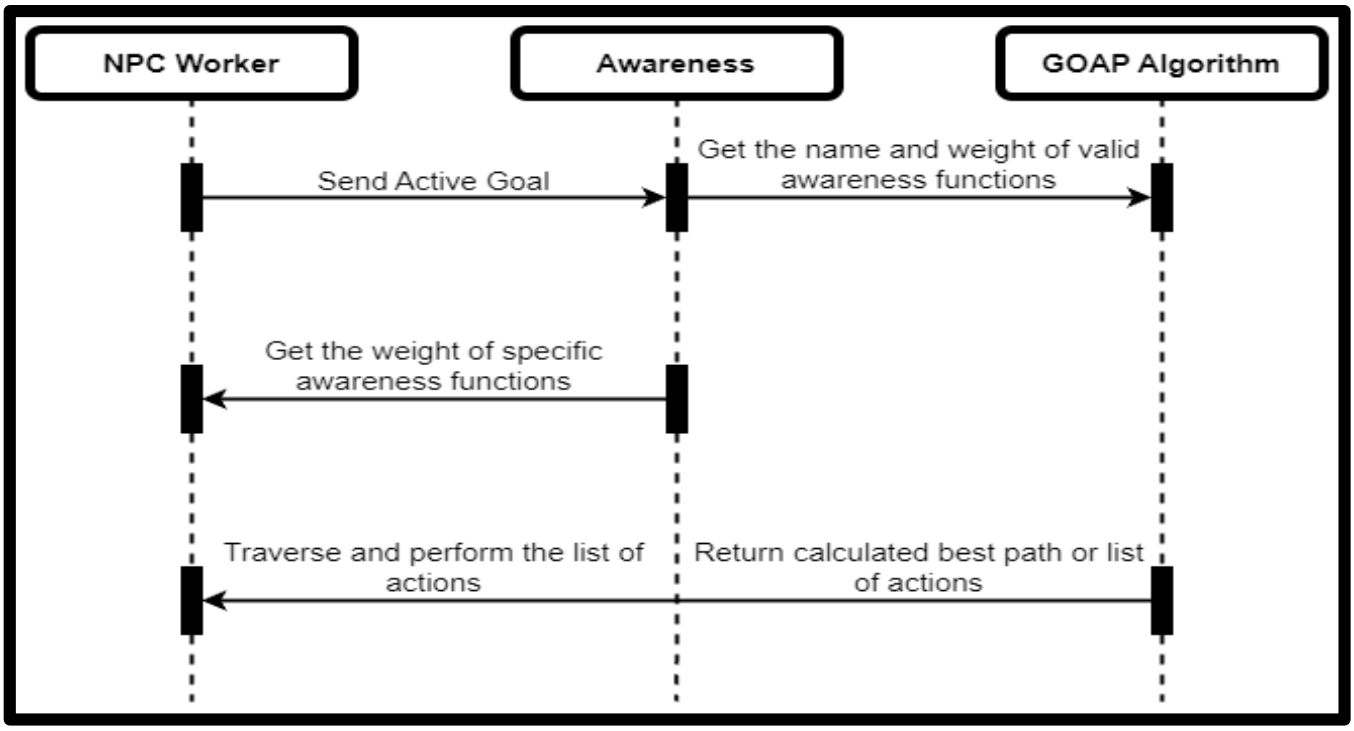

Figure 4. Sequence Diagram

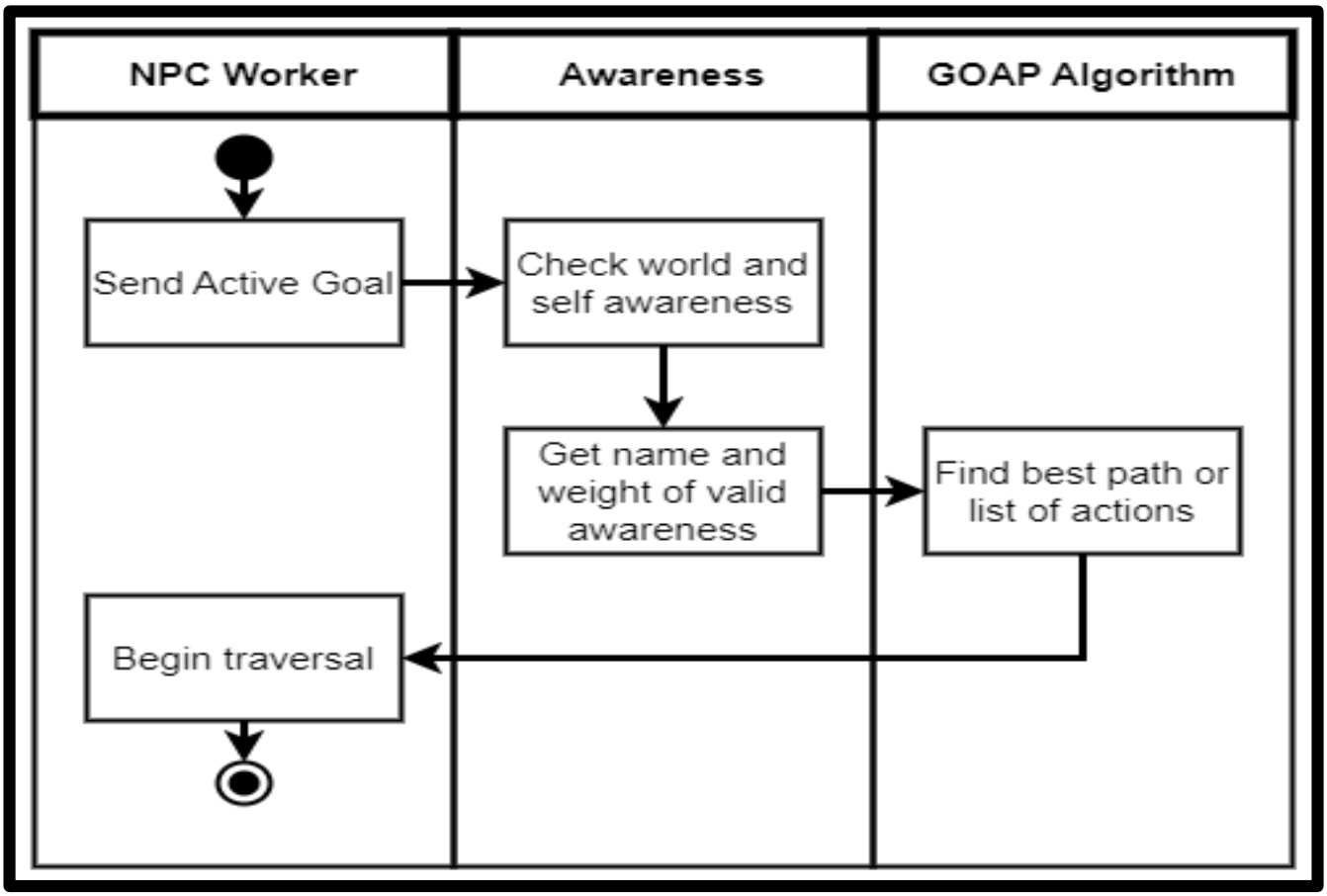

Figure 5. Activity Diagram

5. Results and Analysis

The empirical validation of the GOAP-enhanced NPC worker models was conducted through a technical evaluation involving a diverse panel of 15 respondents, comprising five IT professionals and ten IT students. Participants attempt testing sessions to assess the system across three primary software quality dimensions: functionality, reliability, and efficiency. Each criterion was evaluated on a five-point Likert scale, with the resulting weighted means providing a quantitative measure of the implementation's effectiveness.

Table 1. Respondents' Rating of the Proposed System in terms of Functionality

Criteria	**Weighted Mean**	**Description**
NPC Workers are able to choose different actions	4.53	Very Much Functional
NPC Workers properly perform their specified task	4.6	Very Much Functional
Different systems in game are interacting properly	4.4	Very Much Functional
Game provide NPCs that give more life to the game	4.47	Very Much Functional
NPC Workers don't break the game	4.6	Very Much Functional
Game mechanics are compliant of its user needs	3.87	Much Functional
Intended use of the software	4.47	Very Much Functional
Average Weighted Mean	**4.42**	**Very Much Functional**

The assessment of functionality focused on the NPC's capacity to select and execute tasks within the interactive environment. As shown in Table 1, the system achieved high performance, particularly in the areas of task execution and game stability, both receiving a weighted mean of 4.60. Respondents noted that the different game systems interacted properly without breaking the logic of the environment. The average weighted mean of 4.42 suggests that the application of the GOAP algorithm provided NPCs with the decision-making capabilities to enhance the realism and "life" of the game world.

Table 2. Respondents' Rating of the Proposed System in terms of Reliability

Criteria	**Weighted Mean**	**Description**
NPC Workers perform their task without fail	4.27	Very Much Reliable
Processes performed by the game is accurate	4.33	Very Much Reliable
NPC Workers are reliable in choosing different actions	4.47	Very Much Reliable
Clear difference between two AI approach (GOAP more superior)	4.47	Very Much Reliable
Fitness of Game Mechanics	4.0	Much Reliable
Accuracy of NPC Workers result	3.87	Much Reliable
Completeness of the algorithm	4.33	Very Much Reliable
Average Weighted Mean	**4.25**	**Very Much Reliable**

Reliability was measured to determine the consistency of the algorithm in various world states. The data presented in Table 2 indicates a strong preference for the autonomous approach, with a mean of 4.47 for the perceived superiority of GOAP over traditional manual AI. While "Fitness of Game Mechanics" and "Accuracy of Result" received slightly lower scores (4.0 and 3.87, respectively), the overall average weighted mean of 4.25 confirms that the processes performed by the game are accurate and consistent. This reliability is critical for ensuring that autonomous agents can properly manage background tasks without constant player supervision.

Criteria	**Weighted Mean**	**Description**
NPC Workers perform current action smoothly	4.13	Much Efficient
GOAP Algorithm can integrate any kind of actions and still perform at peak condition	4.2	Much Efficient

GOAP Algorithm efficiently finds the best goal or action	4.27	Highly Efficient
Different systems in the game are working together smoothly	4.6	Highly Efficient
GOAP Algorithm doesn't affect game performance	4.4	Highly Efficient
NPC and World Awareness are smooth in evaluation	4.2	Much Efficient
GOAP Algorithm merge efficiently with other algorithms	4.33	Highly Efficient
Average Weighted Mean	**4.3**	**Highly Efficient**

Table 3. Respondents' Rating of the Proposed System in terms of Efficiency

Efficiency, as measured by the complexity of the GOAP planner, did not negatively impact engine performance. Evaluators found that the system performed smoothly, with the GOAP algorithm successfully merging with other awareness algorithms without causing frame-rate degradation. As detailed in Table 3, the highest score in this category (4.60) was given to the smooth interaction between different in-game systems. The overall efficiency mean of 4.30 suggests that the dynamic evaluation of world and NPC states is highly optimized for real-time applications.

Table 4. Overall Results of the Project Evaluation

Software Quality Factor	**Average Weighted Mean**	**Description**
Functionality	4.42	Very Much Functional
Reliability	4.25	Very Much Reliable
Efficiency	4.3	Highly Efficient
Overall	**4.32**	**Very Much Acceptable**

The overall score of the technical evaluation highlights the robustness of the Goal Dependency framework. Table 4 provides the overall results, showing that the system maintains high standards across all quality factors. With a final cumulative weighted mean of 4.32, the implementation of intelligent NPC worker models using the GOAP algorithm is classified as "Very Much Acceptable." These findings validate the

theoretical premise that a dependency-layered planning architecture can significantly reduce player "babysitting" while maintaining high technical and performance efficiency.

6. Discussion

The results across functionality, reliability, and efficiency show that a dependency-layered GOAP architecture significantly enhances the perceived autonomy of NPC worker models. The functionality mean of 4.42 confirms that the algorithm successfully transitions between complex world states, such as moving from a "defend" state to a "resource gathering" state without breaking the game's flow. This level of performance indicates that the implementation effectively meets the design requirements of interactive environments, where NPCs are expected to be more than static background elements.

The reliability score of 4.25 shows the unpredictability inherent in real-time game engines. The ability of the NPC workers to handle varying inputs and produce consistent results suggests that the Goal Dependency framework provides a stable foundation for autonomous behavior. Unlike traditional state-machine approaches that often fail when an expected environmental variable is missing, the enhanced GOAP model's ability to resolve its own prerequisites ensures a predictable experience for the user. This reliability is a critical factor for developers seeking to implement sophisticated AI in environments where stability is as important as complexity.

Furthermore, the efficiency mean of 4.30 shows that high-level planning algorithms are too computationally expensive for real-time applications. The findings demonstrate that the modified planner can quickly identify and execute the least-cost path of actions without affecting system performance. This efficiency makes the GOAP algorithm a viable tool not just for high-end hardware, but also for scalable game development where resource management is needed.

The overall weighted mean of 4.32 establishes the proposed system as an effective software framework. The acceptance among both IT professionals and students suggests that the transition from a "command-response" model to an automatic "dependency-resolution" model is both feasible and superior. By reducing the "babysitting" burden on the player, this implementation of the GOAP algorithm represents a measurable improvement in the development of intelligent, self-sufficient game agents.

7. Conclusion

The study demonstrated a method for reducing the manual intervention required to manage worker-class NPCs in interactive environments. The implementation, developed within the Unity 3D engine using C#, successfully implemented beyond traditional, state-machine logic by allowing agents to independently resolve environmental prerequisites. The findings indicate that while the GOAP algorithm provides a strong foundation for planning, the added dependency logic is essential for enabling agents to transition between primary goals and necessary sub-tasks, such as resource gathering, without breaking the continuity of the simulation. The study concludes that the strategic utilization and modification of goal-driven behavior can significantly elevate the quality of game AI, offering a more immersive experience by reducing the babysitting headaches often placed on the player.

Based on these outcomes, it is recommended that developers of games such as RPGs and strategy simulations consider the integration of dependency-based planning. While the utilized GOAP model is highly versatile, future work should explore expanding these enhancements to include multi-agent coordination and social intelligence. This research contributes a refinement to the existing field of Game AI, providing a practical blueprint for designers and programmers looking to implement more autonomous and reliable NPC systems through the calculated improvement of established planning algorithms.

References

1. Addoum, M. A., Mekhaemar, J., Rouffet, M., & Jacopin, E. (2021). Khaldun: GOAP both Procedural Level Generation and NPC Behaviors. 2021 IEEE Conference on Games (CoG). https://doi.org/10.1109/CoG52621.2021.9619062
2. Akbar, M. A., Hariadi, M., Praponco, W., & Supeno, M. S. N. (2015). Multi behavior NPC coordination using Fuzzy coordinator and Gaussian distribution. 2015 International Seminar on Intelligent Technology and Its Applications (ISITIA), 203-208. https://doi.org/10.1109/isitia.2015.7219946
3. Coman, A., & Munoz-Avila, H. (2021). Automated Generation of diverse NPC-controlling FSMs using Nondeterministic Planning Techniques. Proceedings of the AAAI Artificial Intelligence and Interactive Digital Entertainment Conference, 9(1), 121–127. https://doi.org/10.1609/aiide.v9i1.12666
4. Feng, S., & Tan, A.-H. (2016). Towards Autonomous Behavior Learning of non-player characters in Games. Expert Systems with Applications, 56, 89–99. https://doi.org/10.1016/j.eswa.2016.02.043
5. Francisco, K. N., Ruste, J. P. L., Sagum, R. A., Song, Y. S., & Sy, D. S. G. (2013). Incongruity Theory Applied in Dynamic Adaptive Game Artificial Intelligence. International Journal of Future Computer and Communication, 2(5), 499–504. https://doi.org/10.7763/ijfcc.2013.v2.214
6. Kopel, M., & Hajas, T. (2018). Implementing AI for non-player characters in 3D video games. Intelligent Information and Database Systems, 610–619.
7. https://doi.org/10.1007/978-3-319-75417-857
8. Long, E. (2007). Enhanced NPC Behaviour using Goal Oriented Action Planning. Research Gate. https://www.researchgate.net/publication/254155611_Enhanced_NPC_Behaviour using_Goal_Oriented_Action_Planning
9. Miyake, Y. (2017). Current Status of Applying Artificial Intelligence in Digital Games. Handbook of Digital Games and Entertainment Technologies, 253–292.
10. https://doi.org/10.1007/978-981-4560-50-4_70
11. Mozo, C. V., Manreal, M. L., & Saludar, F. L. J. (2013). The Implementation of the Complexities of the Decision-Tree Learning algorithm in the Game Whiz. CLOUD, 1(1), 1–1. https://ejournals.ph/article.php?id=6097
12. Perrie, J., & Li, L. (2014). Building a Dynamic Social Community with Non Playable Characters. IEICE Transactions on Information and Systems, E97-D(8), 1965–1973. https://doi.org/10.1587/transinf.E97.D.1965

13. Pihlgren, G. G., Nilsson, M., Larsson, M., Olsson, O., Foughman, T., & Gustafsson, V. (2016). Realistic NPCs in Video Games using different AI approaches. Chalmers Publication Library.
14. https://odr.chalmers.se/items/f2c23a1d-b9a9-4148-b92f-183907970e2d
15. Ranjitha, M., Nathan, K., & Joseph, L. (2020). Artificial Intelligence Algorithms and Techniques in the computation of Player-Adaptive Games. Journal of Physics: Conference Series, 1427(1), 012006. https://doi.org/10.1088/1742-6596/1427/1/012006
16. Sloan, C. (2015). Drive-Based Utility-Maximizing Computer Game Non-Player Characters. Dublin Institute of Technology. https://doi.org/10.21427/D7KK57
17. Vlachopoulos, I., Vassos, S., & Koubarakis, M. (2014). Flexible Behavior for Worker Units in Real-Time Strategy Games Using STRIPS Planning. Artificial Intelligence: Methods and Applications, 555–568. https://doi.org/10.1007/978-3-319-07064-3_48
18. Yashchenko, V. (2014). Artificial intelligence theory (basic concepts). 2014 IEEE International Integrated Reliability Workshop (IIRW), 134-137. https://doi.org/10.1109/IIRW.2014.6918230