Faculty of Mathematics and Computer Science

Multiagent systems course (MAS)

Exploring the integration of BDI agents in games and virtual environments

Liviu-Ștefan Neacșu-Miclea

Department of Computer Science, Babeș-Bolyai University
1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania
E-mail: liviu.neacsu@stud.ubbcluj.ro

Abstract

The Belief-Desire-Intention (BDI) model stands as a prominent cognitive architecture for the development of rational agents, providing a framework grounded in the concepts of Beliefs, representing an agent's knowledge of the world; Desires, embodying the agent's goals and motivations; and Intentions, signifying the agent's chosen plans to achieve these goals. This model has proven particularly valuable in simulating human-like reasoning and decision-making within complex environments, finding significant applications in the realms of games and simulations. This report aims to compare and contrast the utilization of the BDI model across four distinct research endeavors: the work of Adamatti et al. (2009) on virtual players in the GMABS methodology, the framework proposed by Davies et al. (2006) for implementing deliberative agents in computer games, the multiagent system architecture presented by Barreto et al. (2014) in Orphibs II, and the exploration of BDI agents for Real Time Strategy games by Dallatana (2012).

Keywords:
Multi-Agent Systems, Belief-Desire-Intention, Games, Simulation

1. Introduction

Multi-Agent Systems (MAS) has seen successful applications in communication, administration and industry and gained a notable popularity in entertainment, namely in the form of video games AI bots [2]. A common critique to traditional AI-driven virtual players is the plain absence of human-like behavior, negatively felt by the real players interacting with such technology [5].

One proposed solution to this issue is building agents upon the Belief–Desire–Intention (BDI) model, which is recognized for its effectiveness in approximating human approach in problem solving, which generally works by setting goals and making plans to achieve them based on belief on the environment [5] [10] [1]. While it was proven that BDI agents can interact well with real world players in games requiring more complex tasks, like storytelling

[7], counter-examples of this model reportedly under-performing are offered, and unsurprisingly a not too dissimilar model, with roots in psychology as well, successfully replaces it [9]. This only cements the emerging importance of designing agents capable of emulating human behavioral patterns as accurate as possible.

This paper is structured in multiple sections. Section 2 argues the importance of studying BDI models for games and simulations. Section 3 provides a comprehensive literature review on Multi-Agent Systems (MAS) and their application in various domains and introduces the core concepts of MAS and BDI, specifying their key components and functionalities. Section 4 describes the chosen works for debate, providing information on tools and architecture used, communication systems, the experimental setup, evaluation metrics, and results obtained from implementing the framework. Section 5 compares the approaches and discusses the implications of the results, potential limitations, and avenues for future research. Finally, Section 6 concludes the paper by summarizing the key findings and contributions.

2. Relevance of the topic

In game development, BDI is successful at efficiently [2] animating the virtual world (be it a shooter [5], a role-playing [1] or a real-time strategy game [4]) with deliberative agents in a realistic way thus enhancing players' experience and entertainment. However, a sub-genre of games, namely simulators, poses an attractive point of interest with implications way beyond player-base satisfaction. Considering a game as a simplified model of one or more real world aspects, then agents interacting with the virtual environment and with each other can, on one hand, teach us facts about the evolution of the system, like in population dynamics [3] or resource management [1]. On the other hand, agents designed in virtual environments may be reused in other similar contexts [9], or even serve as prototype for real world autonomous robots capable of interacting with objects around [5] or agents responsible for decision making and integrating in social contexts [1].

The potential of games becoming a playground for multi-agent research justifies the effort put into developing and analyzing such systems and designing suitable architectures. Moreover, a simulated environment allows for automation of processes implying running, observing and testing the agents' behavior in contexts where otherwise the presence of a human interacting actor may be necessary, which is one of the core use-cases with a BDI model.

3. Related work

3.1. Multi-agent systems

A multi-agent system (MAS) is an artificial structure populated by interacting agents [8]. Literature has yet to establish a common definition of an agent due to the variety of ways researchers have interpreted and applied this term [8]. However, there exist a list of commonly recognized features an agent should possess. The signature treat of an agent is autonomy [5], being able to act independently without external guidance, having control both over its internal state and its behavior [8]. They are aware to the changes in environment and respond to triggers accordingly, thus being reactive [5], and they are also pro-active, being able to identify potential ways to fulfill their goal and actively take action towards reaching it [8]. Moreover, agents interact with other actors in the system, be they human controlled or other agents themselves, featuring some sense of sociability which allow them to communicate in order to synchronize their activities [8].

In the context of artificial life simulations (A-Life), MAS is seen as a framework for a system which models the entities that live in the simulated world [3].

The integration of MAS in games is an topic of interest in human-computer interaction (HCI) [2] since it needs to take user experience into account. This research emphasis the ability of the agent to manifest human-like behavior as assurately as possible [1] and aim to keep players entertained rather than master the game or play robot-like movements [1] [5].

3.2. The BDI Architecture

The Belief-Desire-Intention (BDI) model in agent systems is an adaptation to computer science of Bratman's philosophical theory of practical reasoning [6]. Based on the currently known state of the environment, called the

belief, the deliberative agent is able to decide on its own at each moment, which action to perform that brings it closer to fulfill its goals, or desires [10]. Out of a set of possible actions, the agent chooses one of them, representing the intention [10]. It is argued that BDI is an efficient computational model for emulating human decision psychology, the idea being that, while the agent commits itself to reach a state that satisfies its desire until it becomes impractical or impossible, the intentions help dispose information that is not relevant or consistent to the specific goal, thus reducing the computational cost [5].

BDI agents are capable of planning and deciding which plan to use given the current goal and a set of beliefs [6] [10].

Due to its connections and fidelity to human psychology [5], as well as its intuition and ease of usage [5], BDI solves the previously raised issue with game-human agent interaction and has become an established approach in game AI development [1].

4. Recent approaches

This section describes some implementations of BDI agents in virtual worlds such as games and simulations. A short description of each game will be provided for context, followed by the specification of agent architectures and agent-specific tools used in their development, communication with the environment, and evaluation of the results from the authors' viewpoint.

4.1. ViP-JogoMan [1]

Jogoman is a role playing game built upon ideas from natural resources management aiming to model the difficult situation of studying water resource quality in a certain region of Brazil [1]. It is a board-like game with cities and lands where players assume diverse roles and try to work together in order to find the water quality and quantity in the cities outskirts [1]. ViP-Jogoman is an extension of this game which runs on computerized media in the form of an interactive simulation, as opposed to its previous physical pen-on-paper variant, and supports remote multiplayer and virtual players, or agents [1]. The game runs for multiple rounds, and during each turn players can make individual decisions, participate in bilateral negotiations with other players, or discuss potential collective strategies for future rounds [1].

The focus of this work is to create agents that are capable of realistic decision-making processes according to their beliefs, desires, and intentions [1]. The agents use a reasoning mechanism to decide on their actions by considering their beliefs and desires, choosing the most appealing intentions.

4.1.1. Tools and architecture

The agents in ViP-Jogoman are built upon the BDI model. They have beliefs, or knowledge of the environment, in the form of state of resources, locations of other agents, and active social norms. They are updated using perception mechanisms that detect changes in the environment, such as resource availability [1]. Each agent is given a behavioral profile which dictates its goals and actions. For example, that mayor class wants to improve their city's specific industry, or the water company administrator is invents in sanitizing and water quality in the area [1].

The agents are implemented in AgentSpeak(L) language using Jason, a programming tool that helps agents decide what to do based on rules and events. The environment is a simulation based on Cormas (COmmon pool Ressources and Multi-Agent Simulations), a multi-agent framework specifically designed to run a simplified model of natural resources management [1].

The agents keep track of their beliefs, like information about roles, their own and other players' plots, performed actions, and update them as the situation changes. Their desires are their main goals, such as reaching a destination (improving the water network, sanitization network, preserving the environment) or avoiding danger (environment degradation, resource depletion, failing to meet demands) [1]. Based on these, they form intentions, which are just step-by-step plans to achieve what they want [1]. If some state changing event occurs (e.g. increased pollution rate) they can rethink their plan and choose another strategy that fits the context (Figure 2) [1].

ViP-Jogoman agents are designed to communicate and interact with other agents as well as the virtual environment (see Figure 1). This communication may include exchanging information, performing cooperative tasks, or responding to dynamic changes in the environment [1].

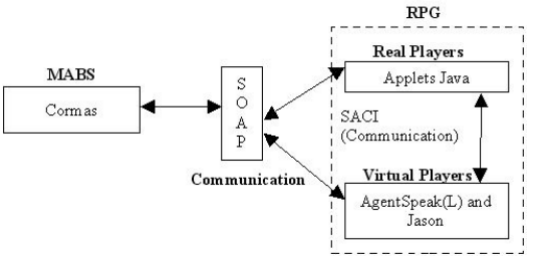

Fig. 1: Architectural design of VIP-Jogoman [1]

```
+plot(L,R): not forest(L)[source(percept)] &
     not settlement(L)[source(percept)] &
     not agriculture(L)[source(percept)] &
     .myName(M) & owner(M,L,P)[source(percept)]
        <- changelanduse(l,agriculture);
           !nextposition(l,r).
```

Fig. 2: Example of strategy in a behavioral profile [1]

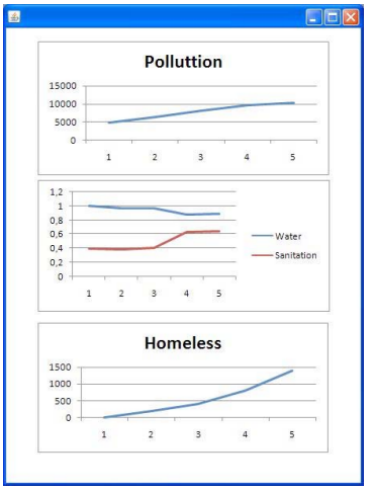

Fig. 3: Example of a game outcome

In order to build up the agents' reasoning, behaviors of real players were carefully analyzed and refined in order to extract behavioral patterns with expert validation, finally compiling them into strategies that are applied depending on whether the environment favors them or not [1].

4.1.2. Communication system

The communication in ViP-Jogoman is needed for agents to talk to each other and to coordinate their tasks and is handled by SACI (Simple Agent Communication Infrastructure) [1].

SACI is a communication layer that allows direct message passing between agents and real players [1]. Agents use this tool in combination with KQML (Knowledge Query and Manipulation Language), a protocol integrated in Jason which lets agents exchange structured messages [1]. This setup makes communication flexible and closer to human behavior, mimicking how people negotiate in real life [1].

The integration between the multi-agent based simulation (MABS) and the RPG game is realized through SOAP (Simple Object Access Protocol) as its "automatic operator" - taking the role of the entity (in case of a physical game - a person) that manages the state of the game based on player's action, the medium between players and the game [1]. SOAP handles XML messages, which makes it ideal in connecting high-level parts of the system that were creating using dissimilar technologies. This ensures smooth interactions between the BDI reasoning layer (Jason using Java) and the Cormas environment which uses SmallTalk language [1].

4.1.3. Evaluation of Results

One of the key results in ViP-Jogoman is that independent real players—who had no prior relationships—displayed remarkably similar behavioral patterns, implying that the BDI agents' reasoning (e.g., rules like "if plot is not forest and belongs to the player, change its use to agriculture") effectively mimicked human decision-making [1]. Three evaluation methods were employed. First, behavioral profile analysis (with roots in HCI) quantifies actions like co-operation or competitivity by correlating variables such as negotiation frequency with specific behaviors) [1]. Then, pre and post questionnaires revealed that real players struggled to identify virtual agents in the game, even sometimes mistaking themselves for bots [1]. Statistics of the rounds (Figure 3) were computed and shown to the players to assist them in solving the questionaries [1]. Post-game Turing tests showed participants found the virtual players' negotiation style realistic [1]. Lastly, message analysis compared interactions in the virtual game to a traditional paper version. Virtual players proactively initiated negotiations (like proposing trades), but real players often ignored these requests unless necessary, leading to a low success rate for virtual-initiated actions [1].

Despite these challenges, virtual agents maintained robust behavioral metrics, adapting seamlessly to real players' unpredictability. Participants rated the game highly realistic, praising its balance of social complexity and environmental dynamics [1].

4.2. Unreal Tournament Agent [5]

An expected critique of traditional game AI it its rigidity and inhuman behavior, which often affects player experience [5]. The work of Davies et al. addresses this matter by proposing a BDI agent framework designed to simulate human-like decision making in dynamic games with open environments. This category is represented in the study by a FPS game named Unreal Tournament [5]. As opposed to rule-based AI, BDI agents use "folk psychology" concepts: beliefs (knowledge about the world), goals (desires), and plans (intentions) in order to emulate the way humans prioritize tasks, adapt to the unexpected, and balance competing objectives. For example, an agent might believe its health is low, desire to survive, and plan to retreat and heal, just like a human player would do [5].

4.2.1. Tools and architecture

Figure 4 describes the design of system components and how they work together. At the core of the agent stays the deliberation layer, which handles reasoning using Jadex, a Java platform specialized for BDI, where agents define beliefs (e.g., enemy locations), goals (e.g., "capture the flag"), and context-sensitive plans (e.g., look for cover if under fire) [5]. Goals are categorized as achieve (target-specific, like reaching a waypoint), maintenance (ongoing, like staying healthy), or perform (state-aligned, like exploring) [5].

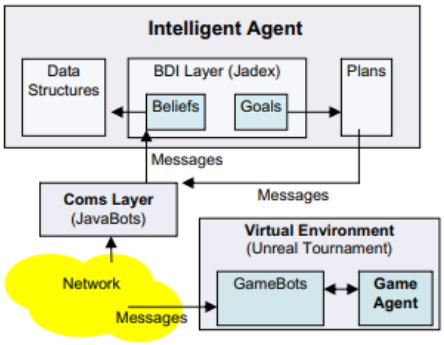

Fig. 4: System design architecture [5]

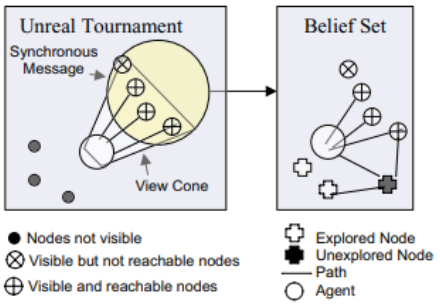

Fig. 5: Message system for navigation purposes [5]

The game itself (Unreal Tournament) provides a dynamic, multiplayer environment (death-match, capture-the-flag) where agents compete with humans or other bots [5]. The authors argue that FPS games are ideal playgrounds because of their fast-paced, unpredictable nature, which requires both reactivity (dodging bullets) and deliberation (planning ambushes) [5].

4.2.2. Communication system

The communication layer connects the BDI agent to the game through a Java Bot client, which interacts with the Unreal Tournament server. The bot sends sensory data (e.g., visible enemies, map layout) to the agent and executes actions (e.g., shoot, move) via network commands [5]. This forces agents to "play blind," relying only on human-like inputs (no internal game cheats, and no omniscient awareness of the map), meeting the conditions of an inaccessible environment [5].

An innovative contribution is the separation between the agent's "mind" (Jadex) and "body" (in-game avatar) [5]. The agent runs externally, communicating with the game via sockets, which mimics human limited knowledge of the environment and ensures fair play [5]. For example, the agent's beliefs are updated only through in-game perceptions (like a player's screen), as seen in Figure 5, and its actions are delayed to simulate human reaction times [5].

4.2.3. Evaluation of Results

The paper acknowledges limitations such as the demo agents not relying on advanced strategies, but on simple behaviors like basic pathfinding or combat logic only for proof-of-concept purposes (Figure 6) [5]. However, the

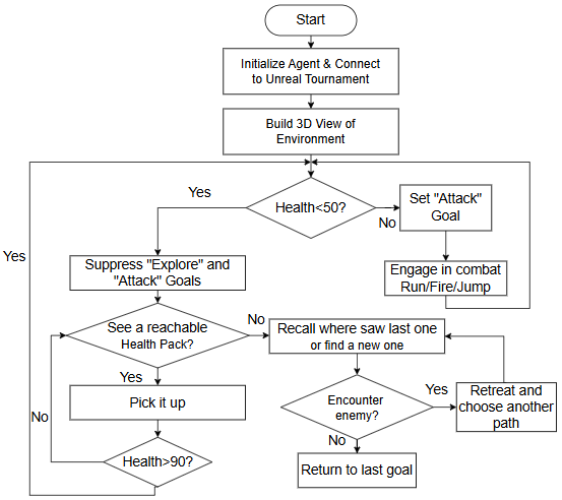

Fig. 6: Unreal Tournament Agent reasoning highlighting prioritizing goals under certain circumstances

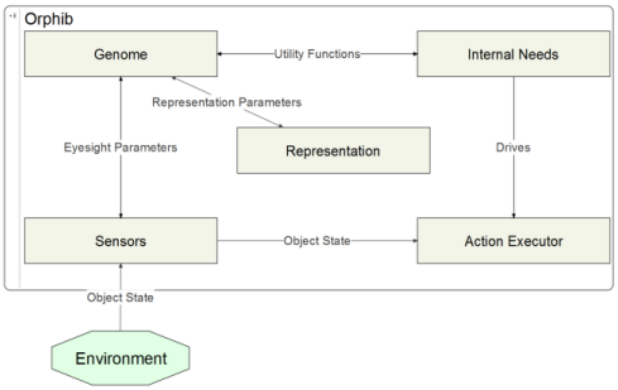

Fig. 7: Architecture of an Orphib's agent [3]

framework's modularity invites future enhancements, such as integrating advanced agent negociation, team cooperation and further architectural improvements [5].

4.3. Orphibs II [3]

Orphibs II is an artificial life (A-Life) game where alien-like creatures (Orphibs) exhibit emergent behaviors through a multiagent system (MAS) [3]. These agents have "personalities" that can be passed genetically and influence their decision-making, blending evolutionary mechanics with goal-driven autonomy [3]. The work exemplifies a bottom-up, agent-centric approach, where individual Orphibs act independently based on internal needs, encouraging complex, unpredictable interactions within a simulated ecosystem [3].

4.3.1. Tools and architecture

The system is built in Unity3D, featuring a component-based architecture that allows modularity and scalability [3]. Agents get their visual perception via sensors that detect objects and other agents. They choose between internal needs (drives) and environmental interactions (e.g., eating, playing) in order to execute an action [3]. Internal needs represent dynamic drives (hunger, energy, fun, social) with decay rates, prioritizing urgent goals, and actions that are chosen by a satisfaction score-based reasoning that takes into account these needs [3]. The genome encodes heritable traits (e.g., color, behavior preferences) for reproduction [3]. This work proposes a Goal-Based Behavior (GBB), which is a simplified BDI model where needs directly translate to goals (e.g., "reduce hunger" induces the "seek food" desire), bypassing complex hierarchies of intentions (Figure 7) [3]. This architecture improves The Sims' need-driven design by adding genetic inheritance, enabling evolutionary dynamics [3].

4.3.2. Communication systems

Orphibs communicate indirectly via environmental interactions and by exerting social behaviors [3]. Social drives prompt agents to seek companionship, thus indirectly influencing group dynamics [3]. Reproduction involves genetic crossover, passing down personality traits to their offsprings [3].

Object interactions (e.g., playing with items) serve as non-verbal actions which alter the environment for others [3].

While explicit communication (e.g., messaging) is not employed, social structures arise from shared needs and environmental feedback [3].

4.3.3. Evaluation of results

Performance is measured through population sustainability and age statistics. revealing the system's ability to balance survival, reproduction and longevity [3]. In a 14 minute simulation with 15 initial random Orphibs, it was observed that the system tends to reach increasing populations with diverse age distributions and no signing of aging population, indicating over-satisfaction of needs (an individual is likely to die from old age rather than other causes)[3]. Moreover, the presence of emergent behaviors (e.g., resource competition, social clustering) validate the MAS design [3]. Future work indicates improving population control and integrating the Computational Belief-Desire Theory of Emotion (CBDTE) to model emotions via memory mechanisms, adding a complexity layer to the GBB system [3].

Finally, Orphibs II demonstrates how MAS and goal-driven architectures can simulate lifelike ecosystems in A-Life games. By prioritizing genetic diversity and need-based autonomy, the system achieves emergent complexity while staying computationally tractable, which may prove useful for both research and game design.

4.4. 0 a.d. Real-time strategy game [4]

ABot is a BDI-based multiagent system designed for the open-source RTS game 0 A.D., providing a new alternative to centralized AI architectures [4]. The work demonstrates how BDI agents can perform adaptive, human-like decision-making in dynamic real-time strategy (RTS) games while balancing realism and computational efficiency [4].

4.4.1. Tools and architecture

This work features an agent-centric BDI multi-agent architecture (Figure 8) designed for the RTS game 0 A.D, where each in-game entity (units and buildings) is governed by an autonomous agent [4]. These agents (Figure 9) operate through a structured framework: beliefs capture their perceptions of the game state (e.g., resource locations, enemy positions), desires define role-specific goals (e.g., gathering resources, training soldiers), and intentions convert these goals into actionable plans (e.g., moving to a resource node, constructing a building) [4]. Modularity is a core feature of the design. Agents inherit behaviors from role-specific modules (Worker, Soldier, TrainingBuilding), enabling code reuse and hierarchical specialization [4]. A new agent-oriented programming language, called Botalk, is proposed in order to streamline development, which abstracts repetitive JavaScript code into succint scripts for defining agent beliefs, perceptions, and plans [4]. Immaterial agents are responsible for coordination, by centralizing critical tasks without direct ties to in-game entities [4]. For example, the Job Market Agent acts like a sort of brokers between workers and jobs, or the Exploration Registry Agent consolidates map data for efficient exploration. [4].

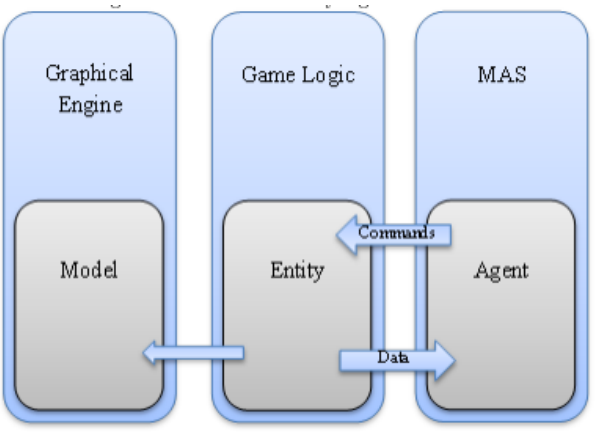

Fig. 8: System architecture [4]

Fig. 9: Agent reasoning loop [4]

4.4.2. Communication systems

Agents in ABot feature an indirect communication system based on a turn-delayed mailbox system, where messages about resources, threats, or job assignments are sent in one game turn and processed in the next turn, thus avoiding synchronization conflicts [4]. This asynchronous approach ensures that agents act autonomously while aligning collective goals, such as prioritizing urgent repairs or directing scouts [4]. By decoupling agent logic from the game engine, the system prevents direct interference with game-state operations, and allows agents to access data about the environment only through perceptions [4]. This design promotes decentralized decision-making while maintaining coherence across the multi-agent societal ecosystem [4].

4.4.3. Evaluation of results

ABot's performance was compared with QBot, the game's centralized AI [4]. The performance was measured for allowing the bots to play for 5 minutes in a preset game scenario and comparing their achieved in-game stats, and eventually the realism test implied comparing the agent's gameplay with one conducted by a real human player [4]. Results proved the potential for adaptability, because ABot exhibits human-like strategies such as early treasure collection and responds to threats proactively (e.g., builds defensive towers after enemy attacks) [4]. Computational efficiency scaled linearly with agent counts, avoiding QBot's performance spikes. Botalk as proved to be 25-35% faster than Javascript, even though perception-handling consumed 30% of runtime due to game-state access bottlenecks [4]. Inside ABot, plan selection takes a considerable 62% of the total time, while perception handling 25%. ABot outperformed QBot in resource management and exploration, but faced late-game inaccuracies from suboptimal parameter tuning (e.g., undervalued food gathering) [4]. While its modularity opened the way for realistic tactics like self-preservation (units on the point of battles flee), rigid military formations revealed limitations in tactical fluidity [4]. However, despite these challenges, this framework demonstrated the viability of MAS for RTS games, and their contribution to balancing autonomy with coordination and the advantage of creating custom tools like Botalk for community-driven extensions [4].

5. Discussion

5.1. Architecture

These four research works showcase diverse approaches to integrating the BDI model in games and simulations. Adamatti et al. (2009) and Davies et al. (2006) both utilize a core BDI architecture, with Adamatti focusing on integrating virtual BDI players into a human-driven RPG simulation for natural resource management [1], and Davies aiming to create human-like AI that controls game characters in a first-person shooter [5]. In contrast, Barreto et al. (2014) in Orphibs II employ a simplified BDI approach (Goal-Based Behavior) augmented with genetically driven personalities, representing a significant yet original deviation from the pure BDI model, which enahnces its flexibility [3]. Dallatana (2012) proposes a highly granular agent-based framework for RTS games where all game entities are modeled as autonomous BDI agents [4].

The level of granularity also differs significantly across these works. Adamatti's virtual players play specific roles within a larger simulation involving human participants, while Davies focuses on the AI for individual non-player characters. Barreto's Orphibs are individual creatures within a life simulation, and Dallatana's approach is the most encompassing, treating every element of the RTS game as an agent. Furthermore, some approaches integrate external concepts that bring benefits to the agents' behavior: Adamatti incorporates behavioral profiles derived from human players, Barreto adds genetic personalities and hints to computational emotions, and Dallatana develops a custom agent platform and domain-specific language for the RTS domain. This wide palette of approaches confirms the adaptability and flexibility of the BDI paradigm and its potential for customization to fit to specific simulation and game requirements.

5.2. Technology

Table 1 displays the technologies employed in each work, reflecting the certains needs and contexts of each application.

Adamatti et al. chose Jason for its BDI interpretation capabilities and Cormas for its specialization in their domain of interest (natural resources), making use of SOAP for cross-language communication. Davies et al. utilized Jadex, a Java-based platform, to implement BDI agents within the commercial game engine Unreal Tournament, employing

Paper	Agent Platform/ Framework	Programming Language(s)	Game/ Simulation Environment	Communication Technology
Adamatti et al. (2009)	Jason	Java, AgentSpeak(L), Smalltalk	Cormas (MABS)	SACI (KQML), SOAP
Davies et al. (2006)	Jadex	Java	Unreal Tournament	JavaBots/GameBots
Barreto et al. (2014)	Unity3D	*Not mentioned (C# or C++)*	Orphibs II	*Indirectly via environment interactions*
Dallatana (2012)	ABot	Custom agent language (Botalk), Javascript, Prolog	0 a.d.	Mailbox system

Table 1: Technologies employed by the discussed approaches

JavaBots/GameBots for the communication interface. The specific technologies used by Barreto et al. for Orphibs II are not detailed, but assumed to be part of the Unity3D framework. Dallatana developed a custom Javascript agent platform (ABot) and a new agent language for implementing BDI in the open-source RTS game 0 a.d.. This diverse selection of specific platforms and languages often depends on factors such as the availability of BDI interpreters, the requirements for integrating with the game or simulation engine, and the need for specialized functionalities.

5.3. Agents and Environment

The characteristics of the agents and the environments they inhabit vary considerably across the four works.

All four environments include changing conditions (player actions, environmental shifts, time-based needs), making them dynamic. Moreover, actions impact future states and may trigger rethinking strategies, thus they are sequential. Unreal Tournament and 0 A.D. are continuous game environments, since actions can happen in real flowing time with unrestricted movements. Orphibs cements its continuous nature by the choice of modeling time-dependent probability density functions for energy and aging [3]. On the other side, the grid based Cormas simulator is indeed a discrete environment. Agents in VIP-Jogoman can see all information about the game, thus they act in an accessible and known environment [1]. It is also the only one that is not affected by stochasticity, therefore it is deterministic. On the other hand, Orphibs can access the entire environment, but does only know about part of it at a time [3]. The RTS game 0 A.D. features a ”fog of war” element that obscures access to certain areas [4], while Davies chooses to let the agent only perceive the same inputs as a human, restricting access to the environment [5]. Table 2 summarizes the properties of the studied environments. It is worth mentioning that some works also focus on the competitive [5] or collaborative [1] [3] [4] aspect of agents interaction.

Environment	Accessibility	Deterministic	Dinamicity	Discrete/Continuous	Episodic/Sequential	Known/Unknown
Adamatti	Accessible	Deterministic	Dynamic	Discrete	Sequential	Known
Davies	Inaccessible	Non-deterministic	Dynamic	Continuous	Sequential	Unknown
Barreto	Accessible	Non-deterministic	Dynamic	Continuous	Sequential	Unknown
Dallatana	Inaccessible	Non-deterministic	Dynamic	Continuous	Sequential	Unknown

Table 2: Properties of the studied environments

Adamatti’s virtual players are designed to assume specific human roles within a natural resource management game, requiring them to exhibit social abilities and adapt to human interactions. Davies’ agents in Unreal Tournament are meant to achieve human-like gameplay in a competitive FPS environment, putting additional effort in strategic navigation and combat. Barreto’s Orphibs in their life simulation possess genetically driven needs, interacting within a not to complex environment focused on basic survival and social behaviors. Dallatana’s agents, representing all entities in the complex 0 a.d. RTS game, must manage resources, engage in warfare, and explore the environment, revealing an advanced understanding of the game’s strategic and tactical elements. The nature of each environment dictates the complexity and required behaviors of the agents populating it. For instance, the fast-paced, real-time nature of an RTS [4] or an FPS [5] game demands efficient decision-making and responsiveness from the agents, whereas simulations [3] and turn-based interactions [1] are favorable for more deliberative planning.

5.4. Communication systems

Communication between agents and/or environment is approached differently depending on the backbone system and requirements. Adamatti et al. implemented explicit communication using SACI with KQML for interaction be-

tween real and virtual players and SOAP for communication between the agent platform and the simulation engine [1]. Davies et al. utilized JavaBots/GameBots to allow communication between the BDI agent system and the Unreal Tournament game server, enabling the agents to perceive and act within the game world [5]. In Orphibs II, communication is an indirect consequence of interacting with the environment, and may play a role in fulfilling the agents' needs (e.g. talking) [3]. Dallatana's thesis employ a mailbox approach for communication in ABot, that allows asynchronous message passing between agents while handling concurency issues [4]. This is crucial for coordination and strategic play in an RTS game. The complexity of information passed and explicitness in the communication systems reflects the specific interaction requirements of each application.

5.5. Results

The reported outcomes of the research vary depending on the focus of each study. Adamatti et al. presents an analysis of the effects of introducing virtual players on the game results in their natural resource management simulation [1]. Davies et al. aims to create more challenging and believable AI opponents and teammates in FPS, with the findings supporting the idea of improved believability by using of BDI agents [5]. Barreto et al. successfully incorporated genetic personalities and needs-oriented reasoning into their Orphibs agents, leading to more varied and engaging artificial life simulation [3]. Dallatana extensively focused on the design, implementation, and evaluation of BDI agents for the RTS game 0 a.d., assessing their strategic capabilities, realism, and performance. While some studies provide more specific findings than others, the overall results suggest the potential of BDI and related architectures for creating sophisticated and adaptive agents in diverse game and simulation contexts, confirming that they can definitely improve player experience by inducing the feel that the user is facing a real opponent, not a poorly acting AI.

6. Conclusions and future work

This report reveals a range of approaches to the implementation of BDI and related agent architectures in the context of games and simulations. While Adamatti et al. and Davies et al. primarily focus on the direct application of the BDI model, Barreto et al. explore a goal-based behavior, and Dallatana proposes a comprehensive agent-centric framework for an RTS game. The choice of technology, the characteristics of the agents and their environments, and the methods of communication are all linked to the specific goals and challenges of each project. The reported results, though not following a standard evaluation methodology, generally enhance the versatility and potential of BDI for creating intelligent agents capable of exhibiting complex and adaptive or even human-like behaviors in interactive environments. Further research direction could focus the integration of emotions with BDI agents in various game genres, investigate more efficient BDI frameworks for real-time applications, and continue to improve agent-based AI in complex strategic environments.

References

[1] Adamatti, D.F., Sichman, J.S., Coelho, H., 2009. An analysis of the insertion of virtual players in gmabs methodology using the vip-jogoman prototype. Journal of Artificial Societies and Social Simulation 12, 7.

[2] Barambones, J., Cano-Benito, J., Sánchez-Rivero, I., Imbert, R., Richoux, F., 2022. Multiagent systems on virtual games: A systematic mapping study. IEEE Transactions on Games 15, 134–147.

[3] Barreto, N., Macedo, L., Roque, L., 2014. Multiagent system architecture in orphibs ii, in: Artificial Life Conference Proceedings, Citeseer. pp. 588–595.

[4] Dallatana, A., 2012. BDI agents for Real Time Strategy games. Ph.D. thesis. University of Bologna.

[5] Davies, N., Mehdi, Q., Gough, N., 2006. A framework for implementing deliberative agents in computer games, in: Proceedings 20th European Conference on Modeling and Simulation Wolfgang Borutzky, Alessandra Orsoni, Richard Zobel l'ECMS.

[6] De Silva, L., Meneguzzi, F., Logan, B., 2021. Bdi agent architectures: a survey, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, pp. 4914–4921.

[7] Goudoulakis, E., El Rhalibi, A., Merabti, M., Taleb-Bendiab, A., 2012. Framework for multi-agent planning and coordination in dis, in: Proceedings of the Workshop at SIGGRAPH Asia, pp. 65–71.

[8] Mahdi, G., Francillette, Y., Abdelkader, G., Michel, F., Hocine, N., 2013. Level of detail based ai adaptation for agents in video games, in: International Conference on Agents and Artificial Intelligence, SCITEPRESS. pp. 182–194.

[9] Orkin, J., Roy, D., 2009. Automatic learning and generation of social behavior from collective human gameplay, in: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, pp. 385–392.

[10] Rao, A.S., Georgeff, M.P., et al., 1995. Bdi agents: from theory to practice., in: Icmas, pp. 312–319.