Autonomous self-learning agents in 3D virtual worlds

Development and Performance Comparison of a BDI Agent Utilizing Reinforcement Learning

Yusri Dollie
University of Cape Town
yusri@zhc.co.za

ABSTRACT

This paper discusses the development and performance comparison of an extended Belief-Desire-Intention (**BDI**) agent which includes notions of reinforcement learning. Referred to as the **R-BDI**, the agent is designed to better adapt and act in a dynamic and noisy environment. The performance of the R-BDI agent developed is compared to a traditional BDI agent implementation, a Partially Observable Markov Decision Process (**POMDP**) and a Markov Decision Process (**MDP**). The performance of the four agent architectures was evaluated using the Q-Cog experimental platform testbed and are compared and benchmarked in terms of survival time, a simulation performance score and overall task completion. Under this criteria we have found that while the R-BDI agent provided a statistically significant improvement over the POMDP and MDP architectures there was no observable improvement in the inclusion of reinforcement learning when comparing the R-BDI to the BDI agent.

CCS CONCEPTS

• **Computing methodologies** → **Reasoning about belief and knowledge**; *Planning under uncertainty*; • **Theory of computation** → *Reinforcement learning*;

KEYWORDS

BDI, Reinforcement Learning, Dynamic Environments, Java, JADEX, 3D Virtual Environments, BDI-POMDP, Cognitive Agents, Hybrid Cognitive Agents

1 INTRODUCTION

In the field of artificial intelligence (**AI**) and agent learning, one of the truly cutting edge and interesting problems lies in creating agents which are able to learn, plan and make decisions in unpredictable and noisy environments. To this end a cognitive agent which embodies the tenets of the Belief-Desire-Intention (**BDI**) agent architecture was developed and subsequently extended to include reinforcement learning mechanisms. The goal being the creation of a rational agent which not only models the concept of rational human decision making, but also includes generic aspects of human behavior such as learning from past experience. To showcase the efficacy and evaluate the performance of such agents an extended version of the Q-Cog platform testbed was used. The platform which is owned and managed by the Centre for Artificial Intelligence Research (**CAIR**) [9] and extended by William Grant [13] to include additional metrics and testing scenarios provides the foundation of the agent implementation. This testbed combined with an extended collection of metrics provide the basis for comparing the different agent architectures and their relative performance when acting in a dynamic 3D environment.

Ultimately the aim of this project was to design and implement a cognitive agent capable of self-learning and adaptation within a simulated 3D virtual world as part of a joint research effort to further the development of the Q-Cog experimental platform, with the extended BDI agent architecture developed by the author and POMDP agent architecture developed by Jonah Hooper [15] providing extensions to agent module and the extended testing environments and metrics developed by William Grant providing the evaluation mechanism for agent performance[13].

The BDI model was conceived by Bratman as a theory of human practical reasoning [5]. BDI agents excel at goal management and planning and manage to act quickly as they utilize already (or partially) generated plans once a particular goal is focused on. Another benefit of BDI's is their ability to handle multiple objectives [27]. However, while the BDI agent metaphor is a well suited analogy for human reasoning, there are many generic aspects of human behavior which it fails to effectively capture. One major aspect which is not incorporated is the concept of learning from prior experiences, and thus many implementations of the BDI architecture completely omit such behaviors and notions [14, 20].

When considering the implementation of such an agent in an unknown and previously unobserved environment where an agent runs the risk of there being no pre-generated plans/policies to execute, it is extremely difficult to ensure that there is a plan for every possible state or occurrence. Successful agents in such environments generally rely on plan generation. However the plans generally do not include stochastic actions or probabilistic observations, and thus in such environments research and development has shifted towards utilizing POMDP planners to generate plans and policies [26].

2 RELATED WORK IN BDI AGENTS

Current developments within the field of BDI agents, and extensions thereof utilizing reinforcement learning provide both inspiration and a guideline for the development of our agent. A brief overview of current and related work is provided, highlighting agent implementations in dynamic and noisy environments.

2.1 Current BDI Agent Research & Development

There has been much research around BDI theory and architectures, with recent developments focusing on creating hybrid architectures, building upon the strengths of the BDI architecture and shoring

up its weaknesses through the addition of learning mechanisms or incorporation of other agent architectures.

There have been many differing implementation strategies and frameworks for the BDI architecture, namely:

PRS: The *Procedural Reasoning System* (PRS) formally introduced by George and Lansky in 1987 was presented as a system for controlling mobile robots, specifically for reaction control in malfunction handling [12]. Developments into extensions to PRS have led to support for real-time planning systems in dynamic non-deterministic environments utilizing the POMDP framework. A novel continual planning system known as POMDPRS has been proposed and developed which gives PRS a stronger ability to adapt to a dynamic non-deterministic environment. This is achieved through the substitution of the probabilistic distribution belief model and maximum utility principles of a POMDP for the first-order logic belief representation and plan selection mechanism of PRS. Interleaving plan generation and execution to improve decision making efficiency in dynamic environments. This system has been implemented in a multi-agent system for Robocup. [17].

AgentSpeak: Introduced by Rao in 1996 AgentSpeak(L) was designed as an agent programming logic, based in BDI theory and logic programming with a formal connection to BDI logics [24]. AgentSpeak is of particular interest as there exist extensive extensions such as AgentSpeak(L++) [1] and AgentSpeak$^+$ [2] which feature probabilistic extensions.

Major implementations of AgentSpeak have been realized through the use of Jason, a Java based development platform for an extended version of AgentSpeak(L). Jason is in essence an interpreter for an extension of the logic-based agent-oriented programming language AgentSpeak(L) [3, 4]

AgentSpeak$^+$ features probabilistic planning using the POMDP framework. Beliefs are represented as epistemic states allowing agents to reason about uncertain observations and using a POMDP; optimal actions are selected in pursuit of a goal given an uncertain environment. This in itself provides a platform for developing BDI's in stochastic environments, where a policy library may not exist or represent incomplete domain knowledge [2] however there are no currently available implementations of AgentSpeak$^+$.

LightJason [1] is a Java based multi-agent BDI framework inspired by Jason. It provides a Java interpreter for the AgentSpeak(L++) grammar, and features a fuzzy-based logical calculus for plan execution and reasoning [1]. While currently still under development LightJason is an ongoing open-source project available to the public.

JADEX: A Java based software framework designed for the creation of goal oriented agents following the BDI architectural model. *JADEX* itself is a reasoning engine developed to simplify the development of adaptive agents for traditional software engineers. Built upon the foundations of software engineering it provides a natural abstraction layer for developing agent oriented systems [6, 22].

Representing a differing school of thought, *JADEX* has moved towards extending the BDI architecture to one which allows for a flexible agenda based on "meta-actions", through which new increasingly complex concrete aspects can be added into the model. These range from simple mechanisms for updating beliefs to more complex goal deliberation strategies which only require minor modifications at well defined extension points. The traditional notion of actions executed as steps of fixed interpreter cycles is abandoned in favor of a interpreter based on the introduction of a new data structure called an "Agenda". The agenda stores a collection of all actions to be processed and the new interpreter continuously selects the next entry from the agenda and executes it thus changing the internal agent state or inclination. This execution may result in the creation of new actions which are inserted into the agenda or may render plans already scheduled obsolete which are subsequently removed. Through this extension of the agent state representation and introduction of a new interpreter this BDI architecture is able to not only provide simple mechanisms for updating beliefs but also proposes a strategy for goal deliberation, a well known area of weakness for classical BDI architectures which assume consistent goals. The shifting of goal arbitration from application level to architecture level allows for the systematic detection of interrelationships between goals and plans at runtime and can be used to preserve a consistent "mental state" for the agent This goal deliberation strategy is known as the "*Easy Deliberation*" strategy and is implemented within *JADEX* BDI reasoner [21].

Unlike other BDI frameworks *JADEX* does not enforce a strict logic based representation of beliefs. It instead allows for ordinary Java objects of any kind to be used and stored as named facts or fact sets (called beliefs), which form the belief base. This belief base can be directly manipulated, with facts being added, removed and updated throughout the agents execution cycle and the reasoning engine monitors these belief states, in turn adjusting the goals and plans in accordance with the current belief states. Goals are seen as concrete momentary desires of an agent. Agents will directly engage in suitable actions until goals are considered as being reached, unreachable or no longer desired, and unlike most other systems adopted goals are not assumed to be consistent to each other. The goal lifecycle [1] distinguishes between goals that are adopted (desired) and those which are actively pursued. In pursuit of goals JADEX utilizes a plan-library approach, with the plans being defined with the goals and or events that the plan is triggered by as well as the set of steps to be executed [22].

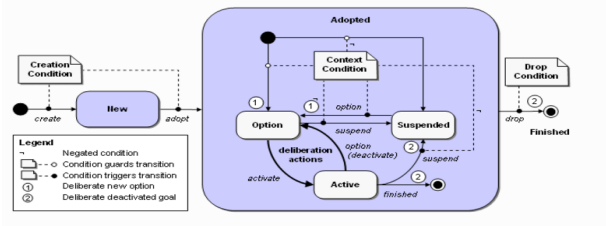

Figure 1: Diagram Showcasing JADEX Goal Lifecycle [7]

[1] see figure 1

2.2 BDI Agents Utilizing Reinforcement Learning

As detailed in section 1 the BDI framework, while providing a fairly comprehensive analogue for human reasoning, fails at capturing aspects such as learning, and as a result many implementations of such agents either plainly ignore such aspects or have to explicitly program them into the agent [20]. One such example can be seen in the video game *Black&White* in which a BDI agent creature can be taught by the user through the use of reinforcement learning, the user provides rewards or punishments in response to the agents behavior, and thus agents future behavior is influenced by the rewards and punishments as certain behaviors are either encouraged or discouraged [19]. This provides a mechanism for shaping agent behavior during execution allowing for it to adapt its behavior to its environment in response to the outcomes of its actions.

Without a mechanism for learning a traditional BDI model would appear to be ill-suited for the development of complex systems where the environment is dynamic, partially observable or wholly unknown wherein learning is key to an agents success. Systems able to successfully operate in such environments are becoming increasingly important not only in the field of robotics but also in business applications, improving productivity and adaptively planning in response to then dynamically changing environment. Lokuge and Alahakoon's paper: *"Reinforcement learning in neuro BDI agents for achieving agent's intentions in vessel berthing applications"* details such an implementation, wherein they discuss how the inclusion of intelligent learning capabilities improves a BDI agents decision making process in a dynamic environment. [18]

Research into these different architectures provided a framework for developing a cognitive agent capable of belief updates and decision making within a dynamic environment

3 GENERAL PROJECT BACKGROUND

As this research project delves into the field of cognitive agents in simulated worlds, looking specifically at the development and performance comparison of a cognitive agent which uses a combination of architecture designs.

It is important to provide the reader with context and clarity. The following sections detail the QCog framework which forms the basis for agent evaluation and comparison, and discusses and defines the BDI architecture, reinforcement learning and the POMDP framework in the context of this paper and current research.

3.1 Q-Cog Experimental Platform

Q-Cog is an experimental platform designed to aid researchers in the development and evaluation of adaptive self-learning agents in 3D simulated environments that are both complex and unpredictable. Developed and owned by the Centre for Artificial Intelligence Research (CAIR)[9], it is built using the Unity3D Game Engine [30] and distributed as both a unity project allowing for the creation of testing environments and as an executable build known as "Q-Cog lite" which allows for cross platform evaluation using built in testing environments. Entities placed in the simulated world are designed to be generic and extensible for ease of use within the platform. Q-Cog features adjustable simulation settings which can be used to: set the number of iterations you wish to perform, environmental variables and initial agent conditions. The results of each simulation are recorded via data recorder outputting a set of performance metrics. It also features a playback engine that allows simulations to be recorded and played back for further study whenever the user desires[13, 31]

3.2 Defining a BDI Architecture

In systems which require high-level management of many different objectives, specifically control tasks in complex dynamic environments where the application of conventional techniques have proven to be difficult and expensive to build, verify and maintain. The BDI architecture represents just one possible solution to this problem in an agent-oriented system. It views the system as a rational agent which has certain mental inclinations of Belief, Desire and Intention (hence the acronym BDI), these inclinations represent respectively: the information, motivational and deliberative states of the agent and determine the agent's behavior [25].

As detailed by Wooldridge [32] the basic components of a BDI are :

- a set or knowledge-base B of beliefs
- an option generation function which generates the objectives the agent would ideally like to pursue (its desires)
- a set D of desires (goals to be achieved)
- a focus function which selects intentions from a set of desires
- a structure of intentions I of the most desirable options/ desires returned by the focus function
- a library of plans and sub-plans
- a reconsideration which decides whether to call the focus function
- an execution procedure, which affects the world according to the plan associated with the intention
- a sensing or perception procedure, which gathers information about the state of the environment
- a belief update function, which updates the agent's beliefs according to its latest observations and actions

The exact implementation of these components is dependent on the BDI architecture. BDI agents are systems which work in dynamic environments and continually receive information (through some form of sensory perception), then based on their inclinations (internal state) carry out particular actions which may also affect the environment [27].

In these environments it is a BDI's flexibility to reason over different goals that allows it to adapt to changing situations by focusing on the most appropriate objective at any particular time [26]

An agent's "Beliefs" are based on everything the agent knows about its environment and internal state, and is stored in its "belief base". Goals describe what an agent seeks to achieve but do not entail any information about the exact set of actions the agent is required to carry out in order to meet said goal. Finally, Plans are composed of a set of instructions which detail the actions an agent will carry

out in an effort to achieve the previously specified goals. The relation between goals and plans is through a reasoning engine, which determines which plans to execute in an effort to satisfy a specific goal [8].

The concept of agency is discussed in detail by Wooldridge and Jennings [33] wherein they identify major areas of concern surrounding agent architecture and agent theory. Specifically they make mention of the connection between the concept of "Beliefs" and its modeling as a set of logical formalisms which can be reduced to a set of mathematical axioms which are then used for reasoning in implementation.

It is here where one can see the link between the BDI architecture and knowledge representation. The agent will have some knowledge base of its beliefs, and decisions are made via a rule engine or logical reasoning engine (sometimes referred to as an inference engine). A reasoning engine provides the BDI with the set of all possible actions based on current beliefs and sensory input, and from the set of possible actions the BDI has to consider, the goals, the agent's inclinations, in order to choose the most appropriate course of action for the agent to carry out. In contrast a rule engine simply has a set of defined rules and productions which when applied produce a possible action for the agent to execute.

3.3 Defining Reinforcement Learning

In the context of cognitive agents, reinforcement learning represents an approach to the problem of agents which interact with an environment, can sense the state of themselves and the environment in which they exist and thus choose actions based on these perceived states and interactions. For an agent to be classified as self-learning or adaptive it must exhibit some form of change in performance and behavior in the execution of future tasks based on prior experience. Reinforcement learning achieves this through the use of rewards and/or punishments in response to an agents performed actions. *Value Iteration*, *Policy Iteration* and *Q-Learning* represent common implementations of the reinforcement learning paradigm [28, 29].

The reinforcement learning paradigm typically separates a problem into four distinct parts: [29]

(1) **A Policy** : A mapping from states to actions representing the agents behavior.
(2) **A Reward Function**: Some function which maps each state-action pair to a real valued number or reward.
(3) **A Value Function**: A function which returns an agent's expected reward given a state and following a policy.
(4) **A Model**: The agent's internal representation of its environment

3.4 Defining MDP's & POMDP's

Basic Markov Decision Processes (MDP) can be described as discrete time stochastic control processes, which form models for decision making when outcomes are uncertain. At each discrete epoch the process observes some state S and may take an any available action A, the resulting transition from $S \rightarrow S'$ provides a corresponding reward $R_a(S, S')$ [23].

In the MDP model the next state and reward depend only on the previous state and action and no other prior state-action pairs, this is what is defined as the Markov property. Agents utilizing the MDP framework attempt to act optimally by calculating an optimal policy, that is a policy which provides the maximum future discounted rewards at the end of a given execution loop. In a partially observable environment however the agent is no longer able to determine its current state with complete reliability [10].

Succinctly a POMDP is simply a MDP where the agent is unable to observe the current state and must instead make an observation based on an action and resulting state according to some probability distribution, while the goal of maximizing future discounted rewards remains the same [10, 16].

4 SYSTEM DESIGN

The system was designed in two parts, with the founding vision being the integration of a new more robust agent architecture within the Q-Cog experimental platform. The initial agent design built upon using an existing Java BDI framework; both for the speed of development it offered, as well as the simplicity of integrating such an agent into an existing Java code base. While the second part of the design focused on utilizing the strengths and functionalities implemented in the initial design and further extending its capabilities through the introduction of Reinforcement Learning mechanisms, leveraging the existing architectural features and utilizing them in unconventional ways.

4.1 BDI Agent Design

The BDI agent was developed using Java and the *JADEX* BDI reasoning framework, specifically the ***JADEX BDI V3*** kernel [22] . The agent implementation is entirely independent of the Q-Cog simulation testbed, with the two interfacing via a TCP connection. Following the BDI architectural design the agent has a set of beliefs referred to as the belief base. Any Java object can be specified as a belief in *JADEX*, and these beliefs represent all knowledge the agent has of its environment. Beliefs are dynamically updated through the use of accessor and mutator methods in following with standard Object-Oriented Programming methodologies. As detailed in section 2.1 *JADEX* does not enforce a strict logical formalism in its representation of knowledge i.e. a description logic such as ***ALC*** or a first-order predicate logic. Instead beliefs are expressed as explicit objects, with the belief base having an elevated role within the *JADEX* agent architecture. The belief base forms an active part of the agent execution cycle, no longer serving only as a passive data store but actively monitoring for any changes in current beliefs and directly triggering the adoption of new goals, the dropping of old goals or leading to direct actions in response. Reasoning as such is no longer limited to just the beliefs stored and entailed by the belief base but extended to allow for reaction deliberation in response to events and changes within the belief base.

The agent within the Q-Cog environment was capable of a total of four actions namely:

- **Explore** - The agent moves to a random location within a radius of its current location in an effort to perceive more of the surrounding environment
- **Attack** - The agent moves to attack a visible hostile entity
- **Eat** - The agent moves to eat a visible food source
- **Flee** - The agent moves to a specified flee location in the environment

These action commands are sent via TCP messages from the Java-side agent to the Unity-side simulation platform, which responds with the environment state information updates. [2]

An agent is only aware of its own internal health level and the entities it perceives via its visual sensors, no other environmental information is made available to the agent. A more detailed description of the experimental environment can be found in section 5 and [13].

The plans within the agent's policy library are composed of these commands forming the agents actions. Plan execution is thus in essence the transmission of one or more of these actions to the Q-Cog-platform and the resultant world state is transmitted then back to the agent as shown in figure 2.

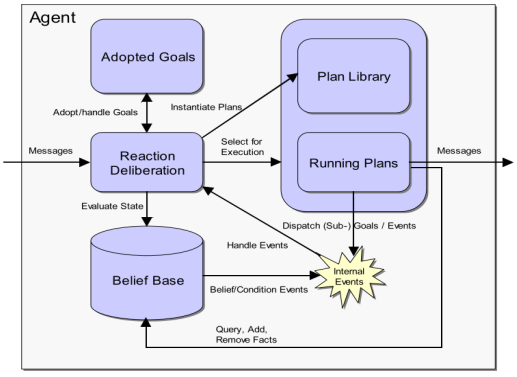

Figure 2: Diagram Showcasing an Abstract JADEX agent Architecture [6]

In order for the agent to have a purpose it must have a goal or set of goals it seeks to achieve, these goals when adopted, or rather when in an active state within the agent can be formally described as the desires of the agent, while the goal currently being pursued by the agent can be described as it's current intention. Goals like beliefs are represented as explicit objects stored within a goal base, these objects are made accessible to the reasoning component but are defined entirely separately from the plans within the policy library [22]. The agent in question was created with four goals:

(1) **Explore**: The agent must seek out to explore it's environment moving around an unknown and initially completely unperceived environment.
(2) **Perceive**: The agent must observe it's surroundings and update its internal belief states based on perceptions which come from its visual sensors.
(3) **Attack**: The agent must seek to engage in combat with hostile entities in an effort to kill all hostile entities in the environment. Attack is triggered on hostile enemies being visible and inhibits the Explore goal
(4) **Survive**: The agent priorities survival over all other goals, it monitors its internal beliefs of health and danger and based on those seeks to flee from danger or seek out food and consume it in an effort to heal. The goal is triggered on health dropping below a certain value and is viewed as achieved once health is above a certain value. It inhibits both the Attack and Explore goals.

JADEX reasons about the goals determining which goal becomes the current intention and this is decided using a combination of logical reasoning based on current beliefs and a native deliberation strategy known as "*Easy Deliberation*". As explained in section 2.1 goals within *JADEX* need not be consistent with each other, and handles goal inconsistency though the use of goal inhibition. For two (or more) goals to be inconsistent it simply means that in the pursuit of progress for one, progress is lost for the other, thus goal inhibition within the context of the easy deliberation strategy allows the agent developer to specify relationships between goals, ensuring that conflicting goals are never pursued at the same time [21]. In this way the pursuit of one goal may inhibit the pursuit of another until the current intention is satisfied. For example in the goals specified above, the Survive goal and Attack goal may at times be inconsistent with one another, the pursuit of combat when the agent has low health is not conducive to its survival. By enforcing the inhibition of Attacking during the execution of the survival intention inconsistent behaviors are prevented and an agent which more closely models human procedural reasoning is created.

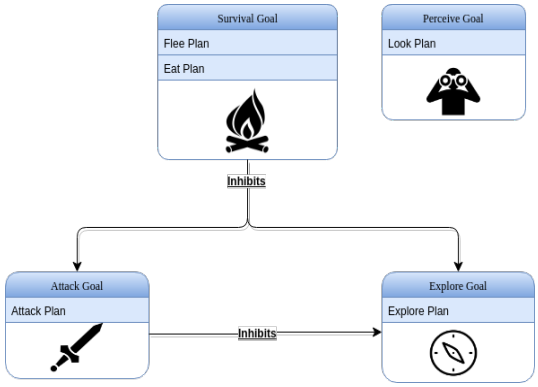

Figure 3: Goal Hierarchy Diagram showcasing Goal inhibitions used for goal deliberation and Plans related to each Goal

Each of these goals is achieved through the enaction of plans which are triggered in response to both a goal being adopted or made active (i.e becoming the current intention) and the current context of the agent's internal belief state. A prime example of this

[2] see figure 2

is the *Survive* goal, the agent may have the desire to survive but the goal only becomes sufficiently desirable and thus considered to be come the current intention in the event that the agent's health is low and/or danger levels are perceived as sufficiently high. Each of the corresponding survival plans themselves will only come into effect in the event that their context conditions are satisfied. The context of the perceived level of danger and health and currently visible food sources determine whether the agent seeks to immediately flee, attempt to eat, or explore until it finds a food source. In addition to this when the Survive goal is triggered it inhibits both the Explore and Attack goals, thus when deliberating about the next desire to become an intention it will not consider exploration or attacking as an option until the survival criteria are met.

The agent was evaluated through the use of the Q-Cog platform testbed, in which it was placed in a dynamic 3D environment and tasked with completing a specific goal or set of goals. In the case of this paper the goal was to kill all hostile entities while ensuring that the agent itself does not die.

Figure 4: Screenshot of Agent running in Q-Cog environment

4.2 BDI Reinforcement Learning Extension

The previous section detailed the implementation of the ordinary *JADEX* BDI agent which was evaluated under the ***Q-Cog*** platform testbed. The full contribution of this research project however is seen in the extension of the BDI agent architecture to include reinforcement learning mechanisms. Referred to as the **R-BDI**, it represents an extension of the BDI agent architecture with the intention of creating an agent capable of self adaptation and learning when acting in a dynamic environment.

When considering a traditional BDI agent implementation as described in section 2.2; while the agent is able to act in a reactionary manner in pursuit of its goals based on its internal beliefs it remains unable to learn from prior experiences. Such an agent will execute plans contained within its internal policy library in the context of its current belief state until a goal is satisfied or no longer pursued with no consideration of which plan may be the best to implement or if an adjustment to its internal plans or reasoning conditions under which certain actions are considered may produce a more favorable result. This mechanism of learning from prior experience is realized in our agent thorough the use of reinforcement learning.

The agent described maintains a set of threshold values which are represented as internal belief states. This representation allows one to utilize the underlying features already present withing the framework and exploit the reasoning capabilities of the JADEX reasoner. These threshold values encode the level of health and danger at which the agent determines which goals to prioritize and which plans it is able to execute. Reinforcements are sent to the agent in the form of rewards and punishments in response to specific environmental events and conditions. The agent is rewarded for successfully killing a number of hostile entities, completing a iteration successfully and punished for dying, and under this scheme the agent is able to "learn" an optimal strategy for completing the goal of killing all hostile entities while balancing its levels of aggression and defensiveness. Thorough the leveraging of the dynamic belief state functionality of *JADEX*, one is able to create and manage the threshold values and utilize its native *Easy Deliberation* strategy to inhibit certain actions in relation to these thresholds.

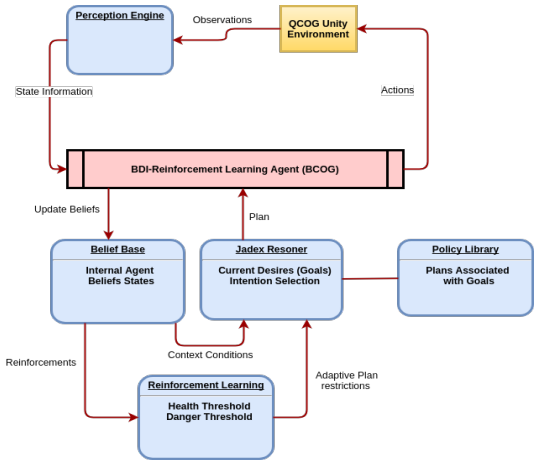

Figure 5: Architecture of Agent Implementation

The introduction of reinforcement learning did not require any extensions to the underlying source code of the framework but its implementation required a different approach to the design of a BDI agent, the reinforcement thresholds were used as the central context conditions for plan execution as opposed to simply the agents current beliefs of the surrounding environment. The threshold values themselves would restrict which goals and plans could become the current intention and thus be pursued. Unlike in the initial BDI agent design described in section 4.1 where goal and plan conditions were dependent only on static values and boolean conditions, the R-BDI is capable of reasoning with ever changing goal and plan conditions, adapting its context conditions in response to the environment in which is is acting.

Algorithm 1 BDI-Reinforcement Learning Agent Algorithm

Require: Initialized set of beliefs **B**, a set of goals to achieve **D**
Input: Updated Belief Base **B'** derived from Perception Engine

1: **Initialization** Initial Belief Base= **B**, Initial Desires =**D** , Initial set of Reinforcement Thresholds = **R** {**B** is the set of initialized belief values at simulation start, **D** is the set of adopted Goals, **R** is the set of threshold values, initialized to arbitrary values and updated via the reinforcement learning mechanism}
2: **while** $Simulation == active$ **do**
3: $intention := deliberate(\mathbf{D}, \mathbf{R})$; {Selecting the current intention based on current context and the set of current desires and current threshold values}
4: $plan := getPlan(intention, \mathbf{R})$;{A plan is selected based on the current intention and the threshold values which have been learned so far governing fight vs. flight}
5: $execute(plan)$;
6: $B = updateBeliefs(\mathbf{B'})$;
7: $R = updateReinforcements(\mathbf{B'})$;
8: **end while**

The above algorithm outlines at a very high level the execution cycle of the agent. The agent manages two threshold values; the health threshold and danger threshold. These values are set to some initial value at simulation start and an optimal value is learned over the ensuing simulation iterations via the reinforcement mechanism. The learned values carry over from iteration to iteration allowing the agent to learn the appropriate strategic approach for the current environment.

Table 1: A Table Depicting Environment Events and Corresponding Reward Value

Environment Event	Reward Value
Agent Dies	-1
Agent kills 2 Predators	+1
Agent kills All Predators	+1

On each of the environment events noted in table 1 above the thresholds for health and danger are adjusted accordingly, the more kills and successful the agent is the more aggressive it becomes as the tolerance for higher danger levels and lower health is increased. It thus attacks more frequently prioritizing attacking over actions such as fleeing and eating. Conversely should the agent die often the lower the tolerance becomes and the resultant behavior exhibited is more cautious and defensive. In this case the agent tends to avoid enemies, fleeing and eating more while approaching combat with greater caution.

For each time the agent dies it is punished and receives a reinforcement of -1 , for every 2 predators it kills it is rewarded and receives a reinforcement of +1 and for each iteration it successfully completes i.e kills all the predators present it is rewarded with a reinforcement of +1.

Adjusting the size of the reward values or the rate at which rewards are given changes the learning rate of the agent, these reward values were developed over many iterations of trial and error to establish an appropriate learning rate, too fast and the agent behavior would wildly swing from one extreme of overly defensive strategies to overly aggressive strategies, while too slow and the agent would not have derived an optimal strategy by the end of the simulation.

5 DEVELOPMENT & EVALUATION PROCEDURES

Development of this project was undertaken in two distinct phases. The initial development phase resulted in the creation of both the BDI and R-BDI agents. The second phase culminated in the evaluation and comparison of the agent architectures developed in phase one to a POMDP and MDP agent architecture.

Development in phase one was undertaken following an iterative methodology, wherein the architecture was refined over time and extensions were added incrementally. While the evaluation in phase two culminated in the statistical analysis of the performance metrics outputted from the extended Q-Cog platform testbed as implemented by William Grant [13] and the discussion of observed agent performance and behaviors exhibited during the execution of the testing simulations .

5.1 Agent Performance & Evaluation

The Q-Cog experimental platform provided an evaluation testbed for measuring agent performance. Each of the four agent architectures were run through three simulations of 20 iterations each. In each simulation they were tasked with surviving in a 3D virtual world which was populated with six hostile predator entities. Iterations ended when the agent either killed all six hostile entities or died. If the agent survived and managed to kill all six of the predators the iteration was deemed successful, if the agent died it was deemed to have failed.

During a simulation an agents performance is recorded along 10 different metrics, this paper examines the agent performance along three of these metrics, which will be discussed in further the next section:

5.1.1 ***Experimental Environment.*** A complex and dynamic testing scenario which was designed and integrated into the Q-Cog platform by William Grant [13] was selected for the experimental evaluation. The scenario is intended to evaluate the performance of cognitive agents tasked with adapting, learning and decision making in a complex dynamic 3D environment.

It features a large 3D world with a complex terrain populated by six hostile predator entities of differing levels of health and damage. In addition there are four food source locations housing two units of food each and two flee locations.

The agent takes the form of a humanoid avatar within the virtual world and is tasked with surviving and killing all predators in the scenario. The agent starts with ten health and can visually perceive its environment within a limited range. A full description of the simulation specifications can be found in [13].

Agent performance is evaluated on the following aspects

- G = Goal success/failure (1 or 0)
- T = Survival Time Previous / Survival Time Current
- A = Attack Success / Attack Total
- P = Predators Killed / Predators Total
- F = Flee Success / Flee Total

From which the iteration score is calculated using the following equation:

$$IterationScore = G + (T \cdot Average(A + P + F))$$

In addition to the above iteration score which calculates an averaged performance value per iteration the following metrics are also used individually.

- Total Predators Killed per iteration
- Total Survival Time per iteration

The three metrics above allow for the discussion surrounding the relative performance of an agent within the simulation.

5.1.2 ***Statistical Analysis***. For the analysis of the numerical data in this project the following are taken into consideration: the number of predators killed, the survival time and agent score per iteration for each simulation, all of which are outputted by the evaluation platform. These results are then tabulated and using a Shapiro-Wilk test it is determined whether or not the data is normally distributed, with a threshold value $\alpha = 0.05$. A non-normal distribution in addition to the non-parametric nature of the data allows one to then utilize a Wilcoxon rank sum test to compare the agent performance data and determine whether or not there is any significant statistical difference between agents. However in the case of a normal distribution one can use a parametric Welch Two Sample t-test to determine statistical significance [11].

Looking specifically to determine whether or not there is any significant difference in the performance metrics between the R-BDI agent and the other agent architectures .All statistical analysis and testing was carried out using ***RStudio***

6 RESULTS GATHERED

The results below represent a summary of the the total data sets used and analyzed, With the focus on showing a subset of the performance data produced by the R-BDI agent. The full datasets can be found at:*https://people.cs.uct.ac.za/~DLLYUS002/*

Table 2: A Table Showing Average Agent Performance

	Predators Killed	Survival time	Success	Score
R-BDI	5.4	236.1	90	1.638813
BDI	5.2	323.9	86.67	1.52148
POMDP	5	300.35	83.3	1.190819
MDP	2.3	178	38.3	0.7691907

Table 2 details the average predators killed, survival time, success rate and score for each respective agent architecture averaged across three simulations of 20 iterations each. The success rate is simply the percentage of total predators killed.

Table 3: A Table Showing the R-BDI Predator Kills For Each Simulation

Iteration	Sim #1	Sim #2	Sim #3
1	6	6	6
2	6	6	6
3	3	2	4
4	6	6	1
5	6	6	6
6	6	6	6
7	6	6	6
8	6	6	6
9	6	6	6
10	6	1	6

Results of Shapiro-Wilk normality Test:

W = 0.53907, p-value = 1.921^{-12} thus non-normally distributed as p < 0.05

Table 4: A Table Showing the R-BDI Survival Time For Each Simulation

Iteration	Sim #1	Sim #2	Sim #3
1	289	288	288
2	253	367	367
3	41	32	32
4	369	300	300
5	210	150	150
6	335	362	362
7	498	464	464
8	171	404	404
9	262	220	220
10	250	23	23

Results of Shapiro-Wilk normality Test:

W = 0.96441, p-value = 0.6351 thus normally distributed as p > 0.05

Table 5: A Table Showing the R-BDI Performance Score For Each Simulation

Iteration	Sim #1	Sim #2	Sim #3
1	1.904762	1.893939	1.904762
2	1.495792	1.403053	1.427689
3	0.6972556	0.6885816	0.640123
4	1.493318	1.406028	0.5409189
5	1.732384	1.795734	1.660272
6	1.711534	1.695297	1.647813
7	1.687845	1.710039	1.804958
8	1.823365	1.76021	1.793801
9	1.817751	1.838713	1.805922
10	1.831093	0.6607918	1.868972

Results of Shapiro-Wilk normality Test:

W = 0.71129, p-value = 1.473^{-09} thus non-normally distributed as p < 0.05

6.1 Testing for Significance

For both predator kills and score metrics it has been shown the results to be non-normally distributed and non-parametric, one can thus utilize a Wilcoxon rank sum test to compare the agent performance data and determine whether or not there is any significant statistical difference between agents, while survival time was shown to be normally distributed and thus a parametric Welch Two Sample t-test can be used. Both tests are carried out using a threshold of $\alpha = 0.05$ to test for statistical significance [11].

The performance of the R-BDI is now compared to the other agent architectures along each of the three performance metrics. A summarized result is presented for each agent comparison along each metric measured.

Table 6: Results of Significance Testing R-BDI vs BDI

	Results
Predator Kills	W = 193.5, p-value = 0.8453
Survival time	t = 1.6965, df = 32.105, p-value = 0.09946 95 percent confidence interval: -12.62059 , 138.52059
Score	W = 223.5, p-value = 0.5338

Through the application of the Wilcoxon rank sum test to both the predator kill and score results it can determined that neither report a statistically significant result in the comparison between The R-BDI and BDI agents. Both applications of the test return a p-value>0.05 and thus the null hypothesis cannot be rejected. Likewise when applying the Welch Two Sample t-test to the R-BDI and BDI survival time results, again a p-value >0.05 is obtained and thus no statistical significance can be conclusively determined.

Table 7: Results of Significance Testing R-BDI vs POMDP

	Results
Predator Kills	W = 252, p-value = 0.0855
Survival time	t = -0.54142, df = 34.867, p-value = 0.5917 95 percent confidence interval: -1302.7118 , 754.2118
Score	W = 309, p-value = 0.002643

Examining the results of applying the Wilcoxon rank sum test to both the predator kill and score results it can determined that while there is no statistically significant difference in the number of predators killed, there is a significant result in the comparison of performance score between The R-BDI and POMDP agents. In examining the Survival time the application of Welch Two Sample t-test between the R-BDI and POMDP, returns a p-value >0.05 and thus no statistical significance can be conclusively determined.

Table 8: Results of Significance Testing R-BDI vs POMDP

	Results
Predator Kills	W = 354.5, p-value = 9.029^{-06}
Survival time	t = 1.5144, df = 33.005, p-value = 0.1394 95 percent confidence interval: -203.3791 , 1387.7791
Score	W = 371, p-value = 3.286^{-07}

Finally the application of the Wilcoxon rank sum test to both the predator kill and score results obtained between the R-BDI and MDP agents indicates that a statistically significant result can be found along both metrics, with both tests reporting p-values<0.05. In contrast however when examining the Survival time, the application of Welch Two Sample t-test between the R-BDI and MDP returns a p-value >0.05 and thus no statistical significance can be conclusively determined.

The results conclusively show that there is no statistically significant difference between the R-BDI and BDI agents. While when comparing the R-BDI agent to the POMDP and MDP agents a statistical significant difference is observed along at least one metric.

7 DISCUSSION

The results obtained clearly suggests a a strong statistically significant result between the performance of the R-BDI and POMDP agents when considering the score metric and between the performance of the R-BDI and MDP agents when considering both the predator kills and score metrics.

However there was no statistically significant result observed when comparing the R-BDI agent to the BDI agent along any of the three metrics. The results obtained between the two agent architectures were virtually indistinguishable, with tests for significance reporting p-values approaching 1. This begs the question of whether the addition of reinforcement learning was truly an improvement to the agent architecture. While the average performance showed an improvement the statistical analysis showed that random chance could not be conclusively ruled out as a possible cause for the observed performance increase. Another consideration is the suitability of the testing scenario used as a measure of learning, and whether or not a different testing environment would provide a better environment in which to evaluate the effect of learning and self-adaptation in AI and thus effectively measure the impact the addition of reinforcement learning had.

There was no statistical significance in the comparison of the survival time metric between agents. When considering the survival time metric, a larger survival time may not always be indicative of improved performance, a short iteration survival time but successful iteration completion indicates that the agent was able to successfully complete the task quickly. This can be seen when looking at Table 2, it is clear that the R-BDI has a significantly lower average survival time when compared to the BDI and POMDP agents, yet still maintains a higher average success rate and thus also a higher average number of predators killed. This highlights

that while survival time on its own is not necessarily a sufficiently robust measure of performance when combined with the context of other metrics it provides an important insight into the overall performance of an agent.

In addition to these results during the execution of the experimental simulations the following points of interest were observed:

The behavior of the R-BDI showed that over the course of a simulation it learned, unlearned and relearned certain strategies as a result of sudden successes or failures. This is apparent in observing the way in which the agent becomes extremely aggressive or defensive in response to a series of sudden successes or failures respectively and over-corrects in its change of behavior. The agent would win a number of iterations, become bold in its action strategy prioritizing attacking vs fleeing and eating, as a result the agent would lose a number of subsequent iterations until the reinforcements made it more defensive and it began to succeed once more. In contrast the POMDP and MDP agents maintained rather consistent behaviors by the end of the simulations as a result. This fluctuating behavior can be attributed to the learning rate of the agent being too high and not being discounted over time, and thus it over compensated in response to the reinforcements it received and did not adequately discount future rewards to converge to an optimal strategy, and thus creating the observed cyclical behavior shifts from aggressive to defensive and back.

Finally it is interesting to note that the R-BDI, BDI and POMDP agents all had rather comparable performances, especially given the vastly different underlying architectures of the BDI and POMDP agent agent models .

8 CONCLUSIONS

The lack of any statistically significant difference between the R-BDI and BDI agent performances within the evaluation scenario illustrate how the addition of reinforcement learning provided a negligible change in agent performance and offers no clear improvement or advantage.

The clear statistical difference in performance score between the R-BDI and POMDP agents leads to the conclusion that in the given evaluation scenario the R-BDI agent was better suited and able to produce a better score. This is further reinforced when comparing the relative survival time and success rates, the R-BDI agent is not only successful more often it also completed the iteration tasks in a shorter amount of time. This highlights the strength of the underlying BDI agent architecture to focus on and pursue particular goals, while being less reliant on initially exploring and learning an underlying environment model.

Finally when comparing the R-BDI and MDP agents one can conclude that statistically the R-BDI agent was able to both outperform the MDP agent in terms of performance score and predator kills. The large difference observed in survival time in conjunction with the relative differences in success rate also further support the improved performance the R-BDI agent showcased over the MDP agent.

Ultimately this research project concludes with the design and implementation of the R-BDI cognitive agent architecture. Resulting in the successful extension to the Q-Cog experimental platform with the addition of a rational agent capable of self-learning and adaptation

9 FUTURE WORK

This project has looked at the design and implementation of the R-BDI agent architecture in order to create a rational agent capable of self adaptation and learning within a 3d virtual world.

The R-BDI architecture itself in its current form has the potential to be expanded further to include a more robust reinforcement learning mechanism, which is tied in to more than just a goal and plan restriction mechanism. Rather there is potential for adding reinforcements to particular plans and policies in an effort to create an agent which is defined in a more generic manner and learns the optimal plan to execute given a specific goal. Creating an agent that utilizes a form of policy iteration to select the optimal plan when many are possible.

Given the shift in research towards creating hybridized agent architectures, specifically those utilizing the BDI-POMDP hybrid frameworks, there is potential for the extension of the R-BDI architecture to include a POMDP planner or policy generator, allowing such an agent to operate in an environment where it does not have a complete set of predefined plans. This hybrid architecture would allow the R-BDI to plan and act in an unknown environment without having a completed set of pre-defined policies and be better able to adapt to the environment through the creation of new policies.

There is also potential for the extension of the reasoning capabilities of the R-BDI agent to facilitate reasoning about probabilistic observations and knowledge. The development of a probabilistic reasoner which is centered around the BDI agent model would allow for more complex and comprehensive goal and plan deliberation strategies.

10 ACKNOWLEDGEMENTS

This work would not have been possible without the continued work and support of my co-researchers William Grant & Jonah Hooper. And special thanks to my supervisors Prof.Thomas Meyer and Assoc Prof.Deshen Moodley who have offered guidance and advice throughout this endeavor

REFERENCES

[1] Malte Aschermann, Philipp Kraus, and Jörg P Müller. 2016. LightJason. In *Multi-Agent Systems and Agreement Technologies*. Springer, 58–66.

[2] K. Bauters, K. McAreavey, J. Hong, Y. Chen, W. Liu, L. Godo, and C. Sierra. 2016. *Probabilistic Planning in AgentSpeak using the POMDP framework*. Springer International Publishing. http://eprints.uwe.ac.uk/31001/

[3] Rafael H Bordini, Lars Braubach, Mehdi Dastani, A El F Seghrouchni, Jorge J Gomez-Sanz, Joao Leite, Gregory O'Hare, Alexander Pokahr, and Alessandro Ricci. 2006. A survey of programming languages and platforms for multi-agent systems. *Informatica* 30, 1 (2006).

[4] Rafael H Bordini and Jomi F Hübner. 2005. BDI agent programming in AgentSpeak using Jason. In *Proceedings of the 6th international conference on Computational Logic in Multi-Agent Systems*. Springer-Verlag, 143–164.

[5] Michael Bratman. 1987. Intention, plans, and practical reason. (1987).

[6] Lars Braubach, Alexander Pokahr, and Winfried Lamersdorf. 2004. Jadex: A short overview. In *Main Conference Net. ObjectDays*, Vol. 2004. 195–207.

[7] Lars Braubach, Alexander Pokahr, Daniel Moldt, and Winfried Lamersdorf. 2004. Goal Representation for BDI Agent Systems.. In *ProMAS*, Vol. 3346. Springer, 44–65.

[8] Juan C. Burguillo-Rial, Martin Llamas-Nistal, David Fernandez-Hermida, and Fernando A. Mikic-Fonte. 2010. A BDI-based intelligent tutoring module for the e-learning platform INES. 6. https://doi.org/10.1109/FIE.2010.5673365

[9] CAIR. 2017. CAIR. (2017). http://cair.za.net/

[10] Anthony Rocco Cassandra. 1998. Exact and approximate algorithms for partially observable Markov decision processes. (1998).

[11] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* 1, 1 (2011), 3–18.

[12] MP George and AL Lansky. 1987. Reasoning about Actions and Plans: Proceedings of the 1986 Workshop. (1987).

[13] William Grant. Evaluating and Extending the Q-Cog Platform. Unpublished manuscript. (????).

[14] Alejandro Guerra-Hernández, Amal El Fallah-Seghrouchni, and Henry Soldano. 2004. Learning in BDI multi-agent systems. In *CLIMA*. Springer, 218–233.

[15] Jonah Hooper. POMDP learning in partially observable 3d virtual worlds. Unpublished manuscript. (????).

[16] Sven Koenig and Reid Simmons. 1998. Xavier: A robot navigation architecture based on partially observable markov decision process models. *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems* (1998), 91–122.

[17] Xiang Li and Xiao-Ping Chen. 2005. A real-time planning system in dynamic nondeterministic environments. *CHINESE JOURNAL OF COMPUTERS-CHINESE EDITION-* 28, 7 (2005), 1163.

[18] Prasanna Lokuge and Damminda Alahakoon. 2005. Reinforcement learning in neuro BDI agents for achieving agent's intentions in vessel berthing applications. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, Vol. 1. IEEE, 681–686.

[19] Peter Molyneux. 2001. Postmortem: Lionhead StudiosâĂŹ Black & White. *Game Developer* (2001).

[20] Emma Norling. 2004. Folk psychology for human modelling: Extending the BDI paradigm. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*. IEEE Computer Society, 202–209.

[21] Alexander Pokahr, Braubach Lars , and Winfried Lamersdorf. 2005. *A Goal Deliberation Strategy for BDI Agent Systems*. Multiagent System Technologies, Vol. 3550. Springer Berlin Heidelberg, Berlin, Heidelberg, 82–93. https://doi.org/10.1007/11550648_8

[22] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. 2005. Jadex: A BDI reasoning engine. *Multi-agent programming* (2005), 149–174.

[23] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[24] Anand Rao. 1996. AgentSpeak (L): BDI agents speak out in a logical computable language. *Agents breaking away* (1996), 42–55.

[25] Anand Rao, Michael P Georgeff, and others. 1995. BDI agents: From theory to practice.. In *ICMAS*, Vol. 95. 312–319.

[26] Gavin Rens, Alexander Ferrein, and Etienne van der Poel. 2010. A belief-desire-intention architechture with a logic-based planner for agents in stochastic domains. (Aug 18, 2010). http://hdl.handle.net/10500/3517

[27] Gavin Rens and Deshendran Moodley. 2016. A hybrid POMDP-BDI agent architecture with online stochastic planning and plan caching. *Cognitive Systems Research* 38, 22 (Nov 2016), 185. https://doi.org/10.1016/j.clinmicnews.2016.10.005

[28] Stuart Russell, Peter Norvig, and Artificial Intelligence. 1995. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs* 25 (1995), 27.

[29] Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT Press Cambridge.

[30] Unity Technologies. Unity Game Engine-Official Site. *Online][Cited: October 9, 2008.] http://unity3d. com* (????), 1534–4320.

[31] Michael Waltham. Design and implementation of the Q-Cog Architecture. Unpublished manuscript. (????).

[32] Michael Wooldridge. 2009. An introduction to multiagent systems. (2009).

[33] Michael Wooldridge and Nicholas R Jennings. 1995. Intelligent agents: Theory and practice. *Knowledge engineering review* 10, 2 (1995), 115–152.