Autonomous Software Agents Project Report

Cappelletti Samuele - 247367 samuele.cappelletti@studenti.unitn.it
Lazzerini Thomas - 247368 thomas.lazzerini@studenti.unitn.it

July 28, 2025

1 Introduction

This document contains the final report for the Autonomous Software Agents project, where one or more agents have to play in the Deliveroo.js game. Agents developed in this project must implement the BDI (*Beliefs, Desires, Intentions*) architecture (Fig. 1) where the agent is able to sense the environment defining an internal belief, generate a set of possible intentions, and commit to one or multiple of them via a plan-based system while performing a constant revision of both intentions and beliefs.

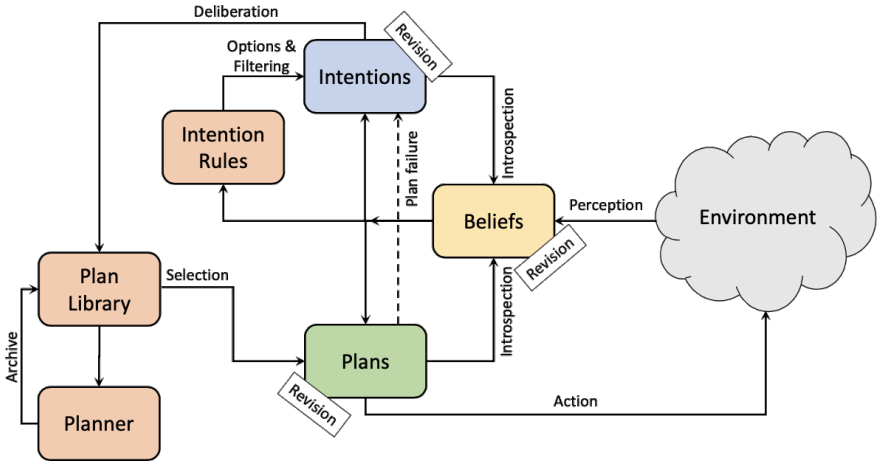

Figure 1: BDI architecture diagram used during the agent development

The project is composed by a single agent part and a multi agent part. The first part should include a single agent implementing the basic functions necessary to him to work correctly. In particular, the agent should represents and mange beliefs from sensing data activate intentions and act on the environment and use predefined plans to achieve its intentions. All of this, while performing a constant revision of beliefs and intentions to allow him to stop, hold or invalid the current running intention. This permits the agent to act accordingly in a rapid evolving environment. Once these functions have been achieved, the agent should interact with an automated planning utility to get the plan of actions to perform.

The second part should introduce a second agent able to cooperate with the first one, and viceversa, to achieve the goal. In particular, the two agents should be able to communicate, exchange beliefs, coordinate and negotiate possible solutions to achieve the goal, which may not be achievable by a single agent.

Furthermore, both should be able to operate in different scenarios, involving other competitive agents and rapidly evolving environments.

2 Single Agent

This part regards the development of a simple single agent, able to perform the basic functions to operate in the *Deliveroo.js* setting. In particular, this should include the ability to represents and manage an internal belief system, built from sensing data received from the server and able to perform revision of outdated or no longer valid beliefs. Based on these beliefs, the agent should be able to define and activate intentions, also performing revision of older intentions, and also act in the environment to achieve such intentions. To achieve the intentions the agent should use a set of predefined plans or an integration with a planner utility to perform the correct actions.

In particular, our single agent operates with the same script as the multi agent configuration, as this multi agent configuration is resilient to the absence of the second agent. Specifically, the single agent configuration will avoid to send messages to the non existing companion and, in all of those situations where it is necessary to know the companion information, like its position or the parcels he is carrying, the agent will consider the companion as null, so like he is nowhere and he is carrying no parcels.

2.1 Initial Connection

The first step is to initialize the agent's *belief set*, this include the initialization of an empty memory. Then, the script will evaluate the command line parameters and will initialize the connection using a specific single agent token or, if required, a token created on the fly. The script will initialize the intention system, will add the predefined plans to its internal library and will initialize the callbacks for the *updates* from the server, like "*onParcelsSensing*" and "*onYou*". Lastly, the script defines a *Promise* on the "*onMap*" and "*onConfig*" callbacks on their resolution before continuing with the normal execution, in particular a temporized loop both on the "*agent's memory revision*" and the "*option generation*" phases.

2.2 Belief

initial connection (mappa e config), memory update (metodi onSensing), memory revision

2.3 Options and Filtering

options generation, e filtering con reward (migliore pickup e delivery)

2.4 Intention

spiegare classi nel file Intentions.js dicendo come gestiamo il push di nuove intention e come facciamo intention revision

2.5 Plan

spiegare i vari piani, come li gestiamo, come facciamo revision dei piani (come la mettiamo insieme a option generation)

2.6 Planning

come abbiamo implementato il planning e per cosa, spiegare che fa schifo, spiegare roba dei negative prepositions (in conflitto con closed world assumption) che fa crashare il FF planner

3 Multi Agent

PICCOLA INTRODUZIONE PICCOLA INTRODUZIONE PER DIRE COSA È STATO INTRODOTTO mettere che il file è unico e lanciamo single o multi agent con parametri (single agent è come il multi agent ma senza pal, agente funziona bene anche senza bisogno di pal (possiamo aggiungere e togliere il secondo agente in qualsiasi momento e il primo continua a funzionare))

3.1 Multi Agent Memory

spiegare cosa abbiamo aggiunto nel belief del single agent

3.2 Communication

spiegare i vari messaggi e il tipo di comunicazione che usiamo

3.3 Belief

spiegare come aggiorniamo belief con info ricevute da pal

3.4 Options and Filtering

spiegare calcolo reward anche per il pal per scegliere solo le options che ci convengono, spiegare come aggiungiamo la option per scambiare parcel (counter per evitare scambi non voluti)

3.5 Plans

spiegare piani aggiunti per scambio carried parcels