A demonstration of BDI-based Robotic Systems with ROS2

Devis Dal Moro$^{[0000-0003-4075-5937]}$, Marco Robol$^{[0000-0003-4611-0371]}$, Marco Roveri$^{[0000-0001-9483-3940]}$, and Paolo Giorgini$^{[0000-0003-4152-9683]}$

University of Trento
{devis.dalmoro,marco.robol,marco.roveri,paolo.giorgini}@unitn.it

Abstract. The paper demonstrates our BDI-based tool-kit built on top of ROS2. We present its main features by means of a realistic industrially-inspired scenario where a fleet of autonomous and heterogeneous robotic systems are asked to move and sort boxes to target destinations. The aim of the demo is to show the advantages of combining the expressiveness of the BDI architecture with an integrated planning system. We show how agents are able to find suitable solutions to achieve their goals in an evolving environment, and how agents communicate and cooperate to achieve common objectives.

Keywords: Real-Time Multi-Agent Systems · Planning & Execution · ROS2

1 Introduction

Industry requires more and more robotics systems with higher degrees of autonomy and capabilities to cope with problems that cannot be exhaustively predicted at design time. This is particularly relevant for I4.0 where state-of-the-art robotic infrastructures, such as Robotic Operating System - ROS - [11], provide only means to reliably sense the environment and promptly react to stimuli without any possibility for robots to autonomously deliberate the best course of action [7, 2, 6, 1, 4, 3, 9]. Different development paradigms, such as the Belief-Desire-Intention architecture (BDI) [10], have been proposed in the literature to overcome the limitations of hard-coded algorithms with limited or no adaptive capabilities.

In our recent work [5], we proposed a first attempt to combine reasoning and planning capabilities with lower level reactive functionalities of a robotic system. The framework has been implemented on top of ROS2 exploiting the (temporal) planning capabilities of PlanSys2 [8]. In this paper, we demonstrate our work in a realistic industrially-inspired scenario. We show the potentialities offered by the tool-kit and its underlying architecture, its expressiveness and its adaptability to real-world problems. The demo focuses on capabilities related to planning, re-planning, and reactiveness, as well as interaction capabilities that allow robots to collaborate and reach common objectives.

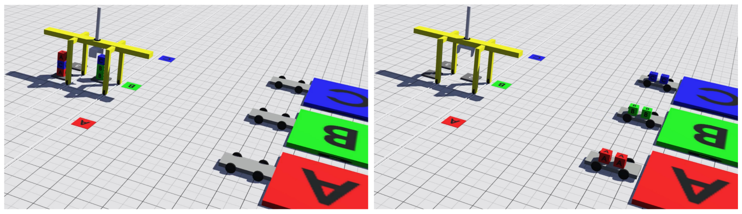

Fig. 1. The initial state with stacked boxes that need to be moved and sorted (left). The final state, where all boxes are delivered to the correct deposit (right).

The demonstration scenario is a logistic problem consisting in sorting and moving boxes to given destinations depending on the specific box type. Robotic agents with different capabilities and partial view of the environment are asked to collaborate one another to solve an overall warehouse problem. The underlying engine allows each agent to reason about the necessary steps for achieving its goals and finding new solutions in the case of a failure or unexpected events (e.g., a collaborator agent is not where it is supposed to be). Additionally, we show how agents can ask other agents to execute tasks (e.g., moving to a target location) for a distributed and adaptive collaboration, without any central control.

2 Main purpose

In this paper, we demonstrate the BDI reasoning and planning system presented in [5]. The demonstration focuses on a very common industrial logistic problem with different collaborative robotic agents having the goal of moving goods. We leverage on a planning-based solution to improve the adaptability of agents so they can cope with unforeseen situations and contingencies. We will demonstrate our framework in two paradigmatic scenarios: the case where everything proceeds as expected, and the case with an unexpected event (anomaly) that forces agents to adapt their behaviour.

3 Demonstration

Figure 1 shows a typical logistic problem, where colored and labelled boxes are initially stacked on different piles. A robotic agent (hereafter `gripper`) can pick and put down boxes, holding a box at a time. It can move boxes between different stacks or load/unload them on "carriers" robots. Carriers can transport boxes back and forth between their assigned loading and deposit areas. Each box is labelled with the target destination: `A`, `B`, and `C`. The `gripper` asks carriers to move in the loading area when they are not there, and carriers when `fully_loaded` move to the deposit area. The final goal for all agents is to transport all boxes to the deposit accordingly to their labels.

Figures 2 and 3 show two timelines, each representing a different run. Each run is initiated by: i) specifying the static information regarding the environment,

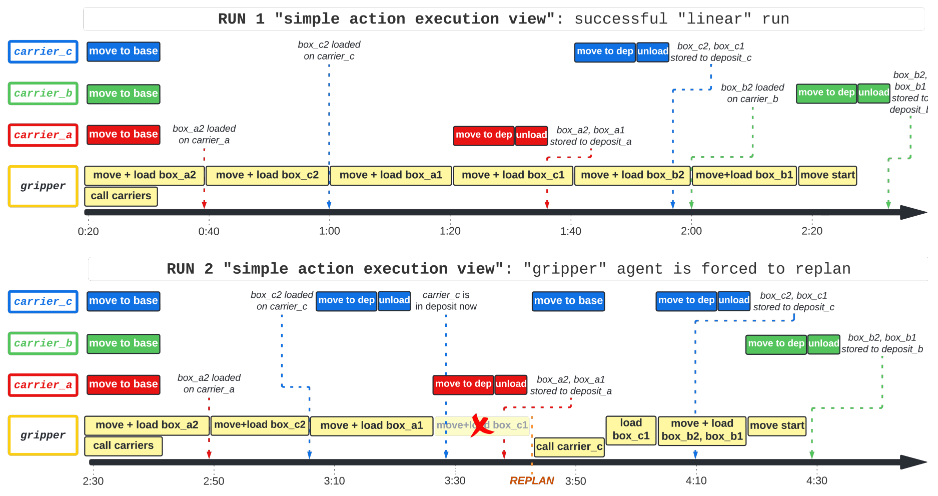

Fig. 2. Timeline of the two runs highlighting the key events and actions.

and the initial configuration to then create the initial knowledge base; and, ii) assigning to the `gripper` agent the goal of loading boxes on "right" carriers. Figure 2 provides a schematic view of the sequence of actions performed over time by the different agents. Figure 3 details what happens in the different phases of the BDI reasoning framework within each agent. In Figure 3, for the sake of readability, we've abbreviated the names of robotic agents, actions, artifacts and places as follows: `gripper` $\mapsto$ `grip`, `carrier_a` $\mapsto$ `c_a`, `deposit` $\mapsto$ `dep`, `base_a` $\mapsto$ `b_a`, `box_a1` $\mapsto$ `bx_a1`, and so on for the others.

In the first run (upper part of Figure 2), starting from the configuration depicted in Figure 1 (left), the `gripper` agent computes and executes a plan consisting in calling carriers to come to their respective base, picking up boxes and loading them on top of the "right" carrier. When a carrier is `fully_loaded`, it goes to unload boxes to the assigned deposit as depicted in Figure 1 (right).

In the second run (lower part of Figure 2), we show what happens in response to an unexpected event. Starting from the same initial state of the first run, the carrier of boxes `C` moves back to the destination area just after the first box has been loaded since it incorrectly believes being `fully_loaded` and that it can go. When the `gripper` attempts to load the second box, it detects that the `carrier` is not there anymore. The execution of its plan fails and the `gripper` has to re-plan from the current state. The new plan will initially require to call back the carrier `C` and then continue as before.

Figure 3 is a more fine grained version of Figure 2 and it aims at providing a dynamic view of what happens in the different phases of the BDI reasoning framework. For instance, it complements the scheduled actions (filled rectangles) performed by each agent with the most relevant information of i) which belief

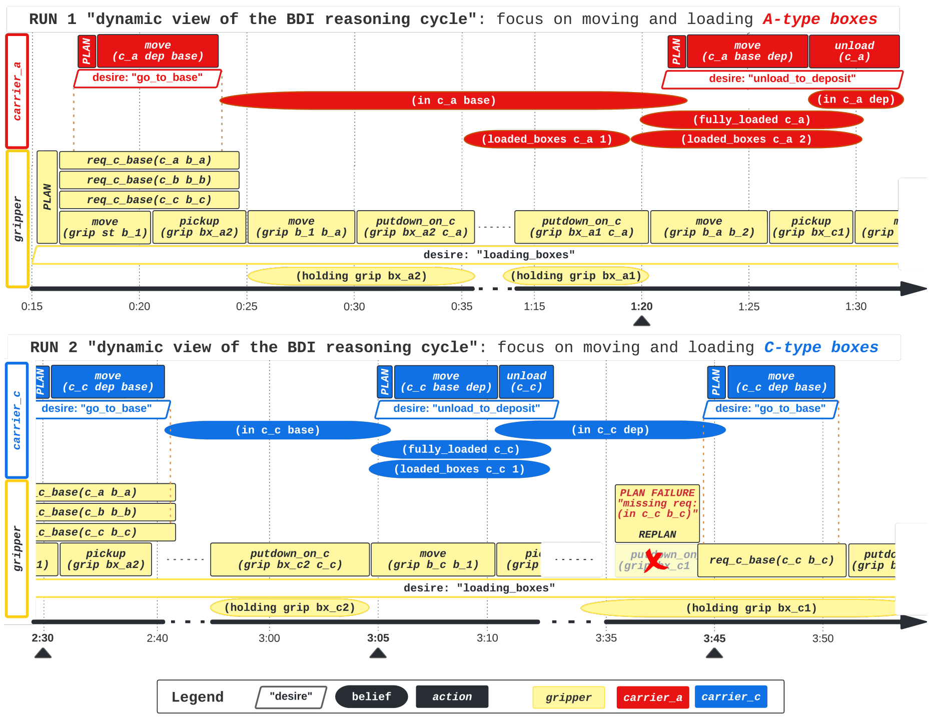

Fig. 3. Timeline of the two runs showing a dynamic view of what happens in the different phases of the BDI reasoning framework with focus on the boxes of type `A` (up), the re-planning triggered by the "misbehaviour" of the `C`-type carrier (down).

and at what time instant it becomes true (filled ovals); ii) which desire and at what time instant it become active for the respective agent (parallelograms).

In autonomous multi-agent systems (MASs), agents act either autonomously or by cooperating/coordinating with other agents. Evidence of this is provided in Figure 3. The phase around instant 2:30 describes a collaborative behaviour in which `carrier_c` and `gripper` communicate with one another. More in details, `gripper` makes a request to `carrier_c`, by posting him the goal `go_to_base`, and expecting him to act accordingly upon acceptance of the request. Then, `carrier_c` decides whether to accept or reject such request, based on static policies defined at agent's design time. In contrast to this, an autonomous behaviour occurs at instant 3:05, when `carrier_c` decides to leave the loading position, without coordinating with the `gripper`, who is still expecting to find `carrier_c` in that position. This behaviour of the `carrier_c` is incompatible with `gripper` current plan, whose action preconditions get invalidated when it is about to load the second type `C` box, therefore the plan execution is aborted. The `gripper`, still trying to achieve its goal, needs to adapt to this new scenario by re-planning and finding another suitable way to pursue it. Again, the newly recomputed plan assumes the collaboration of `carrier_c`, who is asked to move back to its loading base. The request consists in pushing the desire `go_to_base` (at instant 3:45).

The agents in our framework exhibit autonomous behaviours triggered by their reactive capabilities. This corresponds to equip the agents with "option generation" functions in compliance with statically-defined rules. For instance, when carriers are `fully_loaded` (e.g., see Figure 3 `carrier_a` in `RUN 1` at about instant 1:20 and, `carrier_c` in `RUN 2` at instant 3:05), they activate the goal of unloading boxes and then compute and start to execute an adequate plan to fulfill it. Note how this happens also in the `RUN 2` where, in response to an incorrectly defined rule, `carrier_c` erroneously believes to be `fully_loaded` right after the first loaded box (see instant 3:45 in Figure 3).

Figure 3 also provides a view of how the key[1] elements of the knowledge base evolve as result of executing actions, sensing the environment, do reasoning. The successful execution of an action updates the agent knowledge base according to the effect of the action itself. For instance, a successful `pickup` results in updating the knowledge base of the `gripper` agent in a way it knows it is holding the box; a successful `putdown` update the knowledge base to store that it is not holding the box anymore. This update builds on the PlanSys2 Executor [8]. The knowledge base is also updated at the beginning of each iteration of the BDI reasoning cycle by reading the sensors to associate a value to each symbol of the model used for reasoning (e.g., `(in c_c base)`, `(loaded_boxes c_c 1)`). Each sensor publishes its value into a ROS2 topic. The BDI reasoning cycle to acquire the sensor value, subscribes to the needed ROS2 topics, reads the values, processes them and uses them to update the knowledge base. The reading of all the sensors also triggers the "belief revision functions" (a set of static inference rules defined at design time to infer new knowledge). For instance, for the considered scenario a rule states that if carrier agent `c_a` is loaded with at least 2 boxes, it considers itself `fully_loaded`, i.e., the agent updates its knowledge base by adding the belief `fully_loaded` (e.g., if `(loaded_boxes c_a 2)` holds in the knowledge base, then add `(fully_loaded c_a)`).

The demo has been implemented using the Webots (`https://cyberbotics.com`) high-fidelity robotic simulation environment. We exploited its APIs and development toolkit for implementing the actions of the different robotic agents, and for retrieving relevant information at run time from the environment. Webots is one of the state-of-the-art 3D robot simulator to demonstrate the functionalities of a ROS2 application with a high level of fidelity, while providing an almost "out of the box" integration with the robotics middle-ware. Notice that all controls developed with Webots can be deployed with no changes directly to real robots, namely that all results of the simulations correspond to those we can obtain in real-world scenario.

All development and deployment details with instructions to reproduce the demo are available at `https://github.com/devis12/ROS2-BDI`, whereas its recording can be watched here `https://youtu.be/zB2HvCR5H9E`.

[1] For the scenario of this demonstration.

4 Conclusions

In this paper, we demonstrated the BDI reasoning and planning system presented in [5]. We focused on a very common industrial logistic problem with different collaborative robotic agents having the goal of moving goods. We showed the benefit of leveraging on a planning-based solution to improve the adaptability of agents to cope with unforeseen situations and contingencies. The demonstration considers two paradigmatic scenarios: the case where everything proceeds as expected, and the case with an unexpected event (anomaly) that force agents to adapt their behaviour. We also showed a deployment of the framework within the Webots high-fidelity robotic simulation environment. The framework allows for an off-the-shelf deploy of the same controls to real robots.

We envisage several possible directions for future work. Firstly, we will validate the framework in a real environment. Secondly, we will enhance the current BDI reasoning cycle to comply with scenarios where a greater degree of real-time compliance is required [12] or where optimizing a certain metric over time is taken into consideration, instead of blindly fulfill the next highest priority desire. Finally, we will enable a more tight integration between planning and execution, thus to avoid computing an entire plan and deciding only the next action.

References

1. Alzetta, F., Giorgini, P.: Towards a real-time BDI model for ROS 2. In: WOA. CEUR Workshop Proceedings, vol. 2404, pp. 1–7. CEUR-WS.org (2019)
2. van Breemen, A., Crucq, K., Krose, B., Nuttin, M., Porta, J., Demeester, E.: A user-interface robot for ambient intelligent environments. In: Proc. of the 1st Int. Workshop on Advances in Service Robotics,(ASER) (2003)
3. Bustos, P., Manso, L.J., Bandera, A., Rubio, J.P.B., García-Varea, I., Martínez-Gómez, J.: The CORTEX cognitive robotics architecture: Use cases. Cogn. Syst. Res. **55**, 107–123 (2019)
4. CogniTAO-Team: CogniTAO (BDI), `http://wiki.ros.org/decision_making/Tutorials/CogniTAO`
5. Dal Moro, D.: A planning based Multi-Agent BDI Architecture for ROS2. Master's thesis, DISI - University of Trento (2021), `https://www.biblioteca.unitn.it/`
6. Duffy, B.R., Collier, R., O'Hare, G.M., Rooney, C., O'Donoghue, R.: Social robotics: Reality and virtuality in agent-based robotics. In: BISFAI-99 (1999)
7. Gottifredi, S., Tucat, M., Corbata, D., García, A.J., Simari, G.R.: A BDI architecture for high level robot deliberation. Inteligencia Artif. **14**(46), 74–83 (2010)
8. Martín, F., Clavero, J.G., Matellán, V., Rodríguez, F.J.: Plansys2: A planning system framework for ROS2. In: IROS. pp. 9742–9749. IEEE (2021)
9. Polydoros, A.S., Großmann, B., Rovida, F., Nalpantidis, L., Krüger, V.: Accurate and Versatile Automation of Industrial Kitting Operations with SkiROS. In: TAROS 2016. LNCS, vol. 9716, pp. 255–268. Springer (2016)
10. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In: KR. pp. 473–484. Morgan Kaufmann (1991)
11. ROS2 - Robot Operating System version 2 (2022), `https://docs.ros.org`
12. Traldi, A., Bruschetti, F., Robol, M., Roveri, M., Giorgini, P.: Real-Time BDI Agents: a model and its implementation. In: Press, A. (ed.) IJCAI 2022 (2022), To appear.