Capability-Aware Task Assignment for Human-Robot Teams for Empowering People with Disabilities

Carlo Weidemann,[1] Hyun-Ji Choi,[1] Ritesh Yadav,[1]
Stefan-Octavian Bezrucav,[1] Burkhard Corves[1]

[1] Institute of Mechanism Theory, Machine Dynamics and Robotics, RWTH Aachen University
lastname@igmr.rwth-aachen.de

Abstract

This paper presents a novel approach for task planning in inclusive assembly tasks employing human-robot collaboration for people with disabilities. The motivation for this work is to enable people with disabilities to participate in the first labor market by leveraging collaborative robots to support or enhance physical abilities. The proposed method encodes agents' capabilities in the Planning Domain Definition Language (PDDL) and facilitates task planning within the ROSPlan framework. A real-world scenario of human-robot collaboration in a laboratory environment validates our novel approach. It shows that task planning for individual and collaborative tasks is possible. In the validation, a human and a robot assemble a product consisting of five cylinders and a rectangular plate.

Introduction

In 2008, the United Nations passed the Convention on the Rights of Persons with Disabilities (CRPD). Ever since the right of participation in society for people with disabilities (PwD) on an equal basis is supported. "The purpose of the present Convention is to promote, protect and ensure the full and equal enjoyment of all human rights and fundamental freedoms by all persons with disabilities, and to promote respect for their inherent dignity." (United Nations 2006)

In addition to CRPD on an international level, national legislation promotes active support for the inclusion of PwD in the first labor market. For example, the Code of Social Law IX (Sozialgesetzbuch Neuntes Buch - SGB IX) reinforces the rights of PwD in Germany. Despite legal support, people with disabilities are still disadvantaged in society. The unemployment rate of PwD reflects this situation. At 11.8 percent, the unemployment rate of PwD in Germany is significantly higher than the rate of the general German population at 7.3 percent (as of 2020) (Institut Arbeit und Qualifikation der Universität Duisburg-Essen 2023).

Human-robot collaboration (HRC) workstations enable the inclusion of PwD in the first labor. The assistance of collaborative robots makes activities possible that otherwise cannot be performed independently by humans with severe disabilities. HRC is versatile and flexible in its application and has been integrated into several use cases in the industry. For the organization of HRC appropriate task planning modules are required, especially for PwD.

A task planning module allocates tasks of an overall process to agents to reach an overall goal. There are several approaches to task planning, e.g., cost-oriented (Roncone, Mangin, and Scassellati 2017; Takata and Hirano 2011; Tsarouchi et al. 2017), process-oriented (Müller, Vette, and Mailahn 2016), and capability-based approach (Ranz, Hummel, and Sihn 2017). However, existing HRC planning methods are not applicable for inclusive workstations and assume collaboration between robots and humans without physical impairments. Established task planning approaches do not respect the individuality of humans. In this regard, particularly the individual physical capabilities of PwD are significant. The allocation of a task must respect physical impairments. Additionally, in HRC scenarios, robots do not have all the capabilities to perform complex tasks independently and replace all capabilities of a human. Exemplary, the gripping of different objects is difficult to achieve with a single gripper. Therefore, a new task planning module is required that allocates tasks according to the agents' capabilities.

The main contributions of this work are:

- Development of a capability-based task planning module for PwD in HRC assembly scenarios, and
- validation of the proposed module using a real test scenario.

State of the Art

Task planning for inclusive workstations is a critical research area that aims to provide equal opportunities for individuals with different abilities such that they perform their work effectively and efficiently. In recent years, several task planning methods were developed that consider the needs of diverse user groups, such as people with disabilities or aging-related impairments. This section presents an overview of the latest advancements by focusing on relevant task planning methods for inclusive workstations and on methods for determining the level of challenge of people.

This section also briefly introduces automated planning methods, the Planning Domain Definition Language (PDDL), and the ROSPlan framework (Cashmore et al.

2015) with its extensions. ROSPlan is an automated state-of-the-art task planning and execution framework that is deployed in several industrial applications, but not yet for the scenarios targeted in this work.

Task Planning Methods

For the task allocation between humans and robots, scientists use decisive criteria. In the methods of Roncone, Mangin, and Scassellati (2017) and Takata and Hirano (2011), the focus is set on the economic aspect of time in the form of costs. The collaborative work is measured and evaluated according to the total execution time. Another research work by Tsarouchi et al. (2017) additionally considers the changeover and waiting times. The orientation towards an economic cost analysis aims at minimizing cycle times and thus increasing production.

Another approach is process-oriented task planning by Müller, Vette, and Mailahn (2016). For this purpose, the execution of individual manufacturing steps by humans and robots are weighted, compared, and evaluated, whereby the evaluation is based on product and manufacturing requirements. Furthermore, Ranz, Hummel, and Sihn (2017) introduce a capability-based task scheduling. In doing so, they consider the universal capabilities of humans and robots and evaluate them according to executability. This approach considers the individual capabilities of humans and robots and divides tasks with a focus on human work quality and satisfaction. The Methods-Time-Measurement-Analysis (MTM-Analysis) breaks down the overall process into fundamental motions and divide it into variable and invariable tasks (Bokranz, Landau, and Becks 2006). Invariable tasks are fixedly assigned to an agent and variable tasks are flexibly assigned to the human or robot.

The latter approaches demonstrate the diversity and technological progress of the task planning of HRC. They have in common that the individuality between humans and robots is recognized and evaluated based on different criteria. However, in the case of HRC in inclusive workstations, it is not adequate to solely address the differences between humans and robots. Regarding PwD, considering individual capabilities is decisive, as these vary greatly depending on different impairments and the individuality of humans. Thus, a generalization of human capabilities is not expedient for the development of inclusive workstations (Hüsing et al. 2021).

Determining Level of Challenge

Weidemann et al. (2022) present an approach for determining the level of challenge of humans. On this basis, they develop inclusive human-robot workstations using static task allocation. To consider the individual capabilities of PwD, they define 87 assessable capabilities $\mathbf{f} = (f_1, f_2, ..., f_n)^T$. These capabilities are evaluated based on defined criteria. It applies:

$$f_i = [-3, 3] \in \mathcal{Z}. \tag{1}$$

The higher the value, the higher the level of capability f_i. The approach is divided into three steps. First, the overall task is divided into a sequence of standard processes $T = \{SP_1, SP_2, ..., SP_m\}$. Each standard process consists of a sequence of basic elements $\mathbf{b}_{k,j}$, called required capability profiles (e.g., $SP_1 = \{\mathbf{b}_{1,1}, \mathbf{b}_{2,1}, \mathbf{b}_{3,1}\}$ for the basic elements *reaching*, *grasping*, and *bringing*). Second, the capabilities are assessed and specified in the capability profile $\mathbf{p}$, which includes all capabilities in $\mathbf{f}$ but assessed for a specific person. Finally, the level of challenge $\Delta\mathbf{b}_{k,j}$ is obtained by:

$$\Delta\mathbf{b}_{k,j} = \mathbf{b}_{k,j} - (\mathbf{p} \cap \mathbf{b}_k), \tag{2}$$

where $(\mathbf{p} \cap \mathbf{b}_k)$ is the sub-set of the person's capability profile $\mathbf{p}$ that includes the relevant capabilities for the required capability profile $\mathbf{b}_k$. An element value of $\Delta\mathbf{b}_{k,j} < 0$ indicates an underchallenge, and an element value > 0 is an overchallenge.

Weidemann et al. (2022) further develop this methodology and implement a capability-based software tool called RAMB (translated from German: Robotic Assistance for Manufacturing including people with disaBilities). The RAMB tool enables the implementation of the previously explained method. It performs the comparison and the analysis automatically. Based on the analysis results, they distribute tasks between humans and robots: The robot takes over all standard processes where the PwD is overchallenged. The methodology is validated using a visual quality inspection task. Based on the analysis, the robot manipulates the parts while the human visually checks for defects and decides whether a part is OK or not OK. The validation shows that the capability-based approach is feasible. PwD can complete tasks with the assistance of a collaborative robot, which they could not perform without the robot's assistance. Weidemann et al. (2022) allocate tasks statically. The overall sequence of standard processes is predefined. Therefore, the focus of this work is the development of an adaptive task planning module that respects the individual capabilities of participating agents.

Automated Planning and the Planning Domain Definition Language (PDDL)

A propositional automated planning problem can be represented as the tuple $\Pi = (F, A, I, G)$ (Kambhampati and Srivastava 1996), where:

- F is a set of Boolean propositions that describe the state of the world;
- A is a set of actions, for each of which a set of preconditions, a set of effects, and a cost are defined;
- I is the initial state of the system, represented by instantiations, as true or false, of all propositions from F; and
- G is the goal state of the system, represented by other instantiations, as true or false, of all propositions from F.

A solution for the planning problem is a plan $\pi = [a_1, ..., a_n]$, an ordered sequence of actions.

The propositional temporal AI planning formulation extends the one presented above through a more complex representation of actions. The set F, the initial state I and the goal state G remain the same, while the new set A contains new type of actions. For each action a_i, conditions

at the start, at the end, or during its entire execution; effects at the start, or at the end of the execution; and a duration are defined. With this formulation time is introduced. Therefore, more realistic problems can be modelled. Action synchronizations and actions concurrences are only some of the new aspects that can be considered. The solution of a propositional temporal AI planning problem is a schedule $\sigma = [\langle a_1, t_1, d_1\rangle, ..., \langle a_n, t_n, d_n\rangle]$, which integrates not only a correct logical sequence of the conditions and effects, but also the time t_i when the action a_i should start and its duration d_i.

Planning Domain Definition Language (PDDL) (Ghallab et al. 1998) has become the standard language for the representation of such planning problems. Beside its characteristic syntax it imposes the splitting of the planning instance in two parts, the *domain* and the *problem*. In the *domain* part, the used propositions, in PDDL named predicates, the functions, and the available actions are defined. The *problem* part contains the definition of the used types, the initial and goals state.

The ROSPlan Framework and its Extensions

The ROSPlan framework (Cashmore et al. 2015) handles planning situations that are formulated as PDDL planning problems and that must be executed within the ROS middleware (Quigley et al. 2009). This framework contains several modules that first read PDDL files, translate their content into machine format, and save this in a database. The database functions as a knowledge base because it gathers all information about a planning situation: the definition of the PDDL types, predicates, and actions; grounded PDDL predicates describing the actual and final state of the planning environment. The information from the knowledge base is passed by another module of the ROSPlan framework to a PDDL planner that generates a plan. In the next step, each action of the plan can be sent for execution. During the execution of each action, the state of the world changes. These changes are tracked by the knowledge base where the values of the predicates are correspondingly updated.

The ROSPlan framework was extended with more modules like the Supervisor (Bezrucav and Corves 2020) or the Action Interface Manager (Bezrucav et al. 2021). The Supervisor enables a continuous sense-plan-act loop. Based on the actual state of the system, a plan is generated whose actions are sent for execution. Upon execution failure (e.g., determined by specific sensing modules) or change of goals, the Supervisor automatically triggers corresponding re-plan procedures. Once a new plan is available, the sense-plan-act loop is continued with the execution of the planned actions.

The Action Interface Manager (AIM) contains a description of planned actions that must be accomplished. For each type of PDDL action, an Executor Finite State Machine (EFSM) is saved as part of the AIM. Each state of the EFSM contains a call to a specific ROS module (e.g., for trajectory planning or navigation). The transitions between the states are triggered by the outputs, *success* or *fail* (e.g., of the navigation algorithm). For each planned action, the AIM first reads the corresponding EFSM from a database. Afterward, the manager instantiates the EFSM with the parameters of

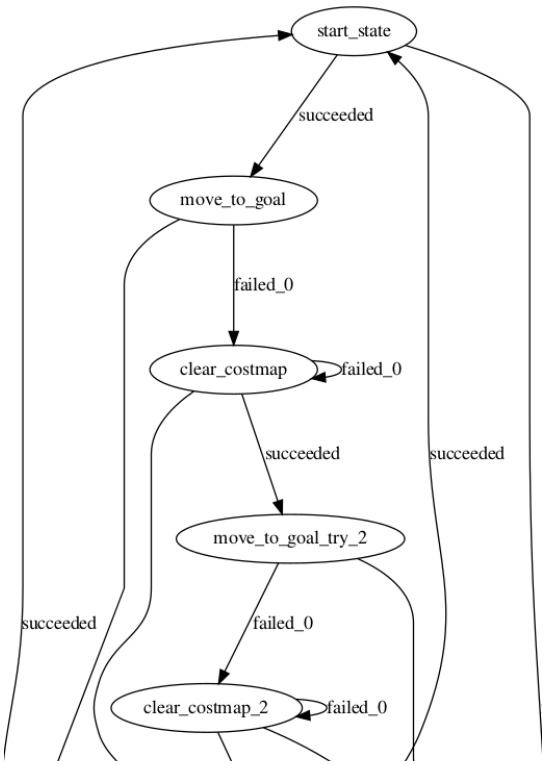


Figure 1: Part of the EFSM Action Interface for a *move* PDDL action represented as a graph.

the planned action (e.g., move goals) and calls the underlying ROS modules based on the states of the EFSM and their transitions. Exemplary, Figure 1 presents the EFSM action interface of a *move* PDDL action for a planning problem with mobile robots. The states of this EFSM contain calls to the MoveBase Framework in which navigation goals are set or the clearance of the map is triggered.

Methodology

In this paper, we develop a capability-based task planning module, which is operated in ROSPlan. For this, the methodological approach of the RAMB tool is applied and transferred into PDDL. As already presented in section *Determining Level of Challenge*, a capability-based task planning module called RAMB is proposed by Weidemann et al. (2022).

Modifications are required to implement this methodology in PDDL. RAMB employs a process-oriented requirement profile. This step includes the breakdown of an overall task into a sequence of standard processes. Then, each standard process is split into a sequence of basic elements. In contrast to that, the requirement profile in PDDL is part-oriented. In the part-oriented requirement profile, the relevant capabilities are directly assigned to a certain part of a given task. 87 assessable capabilities are used in RAMB, whereas in PDDL the number of capabilities is reduced. We only use relevant capabilities of the underlying task. Therefore, the profiles become more compact. Through this ap-

proach, the requirements for each single part of a task can be set and is respected by the task planning module.

The capability profiles in PDDL are extended by a profile for a collaborative robot. Thus, robotic restrictions and human capabilities can be considered. Thereby, technical restrictions for the robot are defined. For example, the limited gripping width of the applied gripper is a technical restriction that is described in the capability profile of the robot. This modification expands the task planning module, considering the capabilities of all agents participating in the HRC. In contrast, the RAMB tool only analyzes the level of challenge of a person and allocates subtasks that represent an overchallenge for the human to the robot.

The third modification is enforced due to technical limitations in PDDL. Ratings with negative numbers have to be avoided. Therefore, the rating limits of capabilities are changed. In RAMB, each capability in the requirement and capability profile is rated on a discrete scale from -3 to 3. The capabilities in PDDL are rated analog to the guideline in RAMB from 0 to 7. A minimum rating of 0 corresponds to the rating -3 in RAMB, which means that a capability is not required or that a capability is non-existent for an agent. The maximum rating of 7 corresponds to a value of 3 in RAMB and indicates that a capability is maximally manifested or fully given by an agent.

With these modifications, the planning problem is formulated in PDDL 2.1. Additionally, temporal and numerical aspects are regarded in the planning problem. The numerical aspect enables the formulation and solving of the capability-based planning problem by using mathematical expressions.

Capability and requirement profiles are set up with PDDL functions. Figure 2 shows an example of a requirement profile. Two required capabilities *cap1* and *cap2* are defined for an assembly task of two parts *p1* and *p2* by assigning values to the PDDL function *req_value*. According to this example, capability *cap1* is required with a value of 1 to assemble *p1*, while for *p2* capability *cap2* is required with a value of 5. Capabilities *cap2* and *cap1* are not required for the assembly of respectively part *p1* or *p2*.

```
1  (= (req_value p1 cap1) 1)
2  (= (req_value p1 cap2) 0)
3
4  (= (req_value p2 cap1) 0)
5  (= (req_value p2 cap2) 5)
```

Figure 2: Requirement profile represented as functions in a PDDL *problem* file.

The capabilities in the capability profiles are rated and represented for each agent by using the function *cap_value*. Figure 3 shows an example of two profiles. These are interpreted as follows: *agent1* possesses the capability *cap1* with a value of 1 and none in *cap2*; *agent2* has no capability in *cap1*, but has the capability *cap2* with a value of 6.

The analysis determining the agent that fulfills the requirement to manipulate a part is located in PDDL *durative actions*. It is found in the definition of the action conditions. Conditions describe an exact state before executing an action. It is necessary to fulfill this state. Otherwise, the action cannot be executed. The analysis step is given in lines 6-7 of Figure 4. For all declared capabilities, the capability value for each agent is compared with the value of required capabilities for each part. The agent whose capabilities are greater than or equal to the required capabilities is allowed to execute the action. Based on this comparison, a task plan is generated. According to the examples in Figure 2 and 3, the capability-based task plan would show that *p1* is allocated to *agent1* and *agent2* would take over *p2*.

```
1  (= (cap_value agent1 cap1) 1)
2  (= (cap_value agent1 cap2) 0)
3
4  (= (cap_value agent2 cap1) 0)
5  (= (cap_value agent2 cap2) 6)
```

Figure 3: Capability profile represented as functions in a PDDL *problem* file.

```
1  (:durative-action action1
2   :parameters (?agent - agent ?part -
       part)
3   :duration (= ?duration 1)
4   :condition (and
5    (at start (not_acting ?agent)) [...])
6    (at start (forall (?c - cap)
7    (>= (cap_value ?agent ?c) (req_value ?
       part ?c))))
8   :effect (and
9    (at start (not (not_acting ?agent)))
10   [...]))
```

Figure 4: PDDL *durative action* in a PDDL *domain* file.

Validation

This section presents an experiment that validates the proposed methodology. The experimental setup consists of two agents (a robot and a human), in a laboratory environment. The robot is a serial manipulator with a gripper. Besides the agents, a working bench, five cylindrical parts, and a rectangular plate are located in this environment.

In the selected planning situation, the human-robot team should assemble a product from the five cylinders and the rectangular plate. The allocation process of the actions to the agents should consider the specific capabilities of each of the agents and ensure a minimal execution time. The following subsections present the planning problem that encodes the considered planning situation and the extended ROSPlan framework that enables the planning and execution of the actions.

Planning Situation Formulated in PDDL

The planning situation is formulated as a planning problem in PDDL 2.1. The PDDL *domain* file contains the definition of types, functions, predicates, and actions. Figure 5 presents these definitions. For the selected scenario, the types represent the entities that are part of the experiment: the agents,

the physical parts, and the part poses (lines 1-4). Furthermore, predicates define the properties of these entities (lines 5-11), while the capabilities of the agents are modeled as PDDL functions (lines 12-15).

```
1  (:types
2  agent part - object
3  robot human - agent
4  partpose - location)
5  (:predicates
6  (at ?obj - object ?partpose - partpose)
7  (on ?parent - object ?child - object)
8  (not_acting ?agent - agent)
9  (placed ?part - part)
10 (empty ?empty_partpose - partpose)
11 (magnet_part ?magnet_part - part))
12 (:functions
13 (req_joint_action ?part - part)
14 (cap_value ?agent - agent ?cap - cap)
15 (req_value ?part - part ?cap - cap))
```

Figure 5: PDDL types and predicates for the selected planning situation.

In the second part of the PDDL *domain* file, the actions are defined. Figure 6 shows exemplary the definition of the *pick_and_place_two_agents* action. This action can be planned if a set of conditions holds. Most of these conditions are verified at the start of the action and are related to: the positioning of the pieces (lines 5-7), the requirement for collaborative action (line 8), and the verification of the capabilities-requirements match (lines 9-12). The effects of the *pick_and_place_two_agents* action describe how the positioning of the handled pieces has changed (line 13). Besides this action, the PDDL *domain* file of the selected scenario contains the definitions of five more actions.

```
1  (:durative-action
       pick_and_place_two_agents
2   :parameters (?agent1 - agent ?agent2 -
        agent ?part_above - part ?part_below
         - part ...)
3   :duration (= ?duration 1)
4   :condition (and (at start (at ?
        part_below ?start))
5   (at start (on ?part_above ?part_below))
6   (at start (placed ?parent1))
7   (at start (not (= ?parent1 ?parent2)))
8   (at start (= (req_joint_action ?
        part_below) 1))
9   (at start (and (forall (?c - cap)
10   (>= (cap_value ?agent1 ?c) (req_value
         ?part_above ?c)))))
11  (at start (forall (?c - cap)
12   (>= (cap_value ?agent2 ?c) (req_value
         ?part_above ?c))))...)
13  :effect (and    (at end (on ?part_below
         ?parent1)) ...))
```

Figure 6: Snippet of the definition of the pick_and_place_two_agents PDDL action.

To fully described the planning situation, an initial state and a set of goals must be set. This information is encoded in the PDDL *problem* file. Figure 7 shows the PDDL *problem* file for the selected scenario. This file starts with the instantiation of the types to objects. Two agents, different parts, and corresponding poses in the environment are introduced in lines 2-5. In the second part of the PDDL *problem* file, the initial state is defined. This state contains on one side the initial positioning of the parts in the environment at the part-poses (lines 7-8) and on the other side the definition of the capabilities for the agents (lines 11-14), of the requirements for the parts (lines 16-17), and of the requirements for joint actions (lines 19-20). The PDDL *problem* file is finalized with the goal state that describes how the different parts should be positioned at the end relative to each other, such that the final product is obtained. The formulation of the PDDL *problem* file completes the formulation of the selected planning situation in the PDDL format.

```
1  (:objects
2   human1 - human
3   robot1 - robot
4   p1 p2 p3 p4 p5 plate1 - part
5   pp1 pp2 pp3 ... - partpose)
6  (:init
7   (at p1 pp1)
8   (at p2 pp2)
9   ...
10  ;Capabilitites
11  (= (cap_value robot1 cap1) 0)
12  (= (cap_value robot1 cap2) 2)
13  (= (cap_value human1 cap1) 1)
14  (= (cap_value human1 cap2) 1)
15  ;Requirements
16  (= (req_value p1 cap1) 0)
17  (= (req_value p2 cap1) 1)
18  ...
19  (=(req_joint_action p1) 0)
20  (=(req_joint_action plate1) 1)
21  ...
22 (:goal
23  (and
24  (at p4 pp4)
25  (on plate1 p4)...))
```

Figure 7: Snippet of the initial state and the goals formulated in the PDDL *problem* file.

Real-World Planning and Execution

In the following step, the extended ROSPlan framework is set up. The framework receives the formulated PDDL files, loads them in its Knowledge Base, and interfaces them to the automated planner called Partial Order Planning Forwards (POPF) (Fox and Long 2010). The planner computes the plan represented in Figure 8.

The obtained actions are then sent for execution over the Action Interface Manager module. This module triggers, supervises, and interprets the real-world execution of each action of the plan, based on the EFSMs formulated for each PDDL action of the *domain*. Each EFSM describes which underlying execution modules must be called and how the

```
1  0.00000: (
       pick_and_place_part_with_magnet
       human1 p3 pp3 plate1 plate1_m)
       [1.00000]
2  0.00000: (pick_and_place robot1 p4 tpp1
       pp4) [1.00000]
3  1.00100: (pick_and_place robot1 p5 tpp2
       pp5) [1.00000]
4  2.00200: (pick_and_place_two_agents
       robot1 human1 p3 plate1 wbp1 p5 p4)
       [1.00000]
5  3.00300: (pick_and_place_on_part robot1
       p1 pp1 plate1 plate1_l) [1.00000]
6  3.00300: (pick_and_place_on_part human1
       p2 pp2 plate1 plate1_r) [1.00000]
```

Figure 8: The generated plan for the selected planning situation.

results of these modules should be interpreted.

For the considered scenario, four different EFSMs are formulated. Figure 9 depicts the second half of the EFSM for the *pick_and_place_two_agents* action, in which the two agents collaborate. The EFSM of the action *pick_and_place_two_agents* has been subdivided into three distinct sections. The first section represents the retrieval of the plate, which is carried out by the robot. This section is not represented in Figure 9. Once the robot has grasped the plate, it should move to the *home* pose and afterwards position the plate for human manipulation. The positioning basic actions are depicted by the first two states from Figure 9. Each of these states represent a call to the MoveIt framework (Sucan and Chitta 2020) that implements the trajectory planning and execution on the real robot. Once these two basic actions of the second part of the EFSM are successfully executed, the *human_action* state is triggered. This state is implemented by a basic module that prints on a screen what the human should do. If the human returns success, the execution of the planned action is continued with the following steps of the last section that imply further calls to the MoveIt framework and to the robot's gripper. If all states of the EFSM are correctly executed, the Action Interface Manager return success to the ROSPlan framework that dispatches the next planned action.

The formulation of the EFSMs completes the configuration process for the planning and execution framework. With this configuration, the experiment was conducted several times with 100% success rate. The final product was successfully assembled each time. Figure 10 presents six different steps during the execution. For example, Step 3 represents the collaborative action.

Summary and Conclusion

This work presents a methodology that enables PwD to be included in the first labor market by solving the challenge of a correct task allocation procedure. Our approach considers the agent capabilities as part of the planning situation and determines planning problems that encode these capabilities. Further, a self-contained planning and execution framework is developed based on the ROSPlan framework. Finally, we have validated the proposed methodology with a real-world experiment involving a serial manipulator and a human that cooperate in an assembly task.

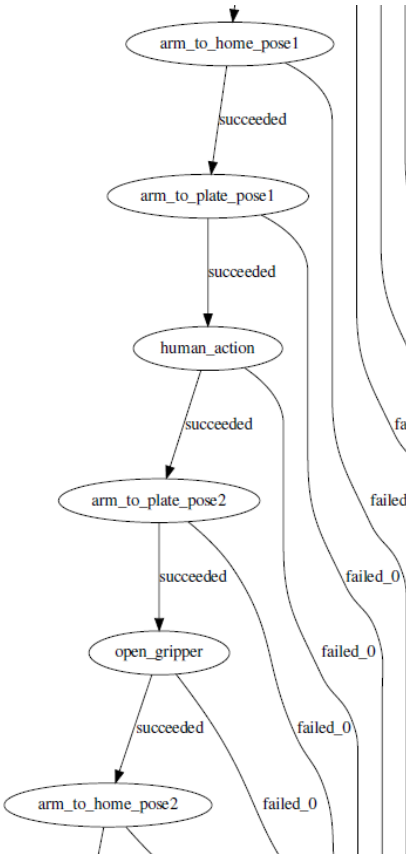


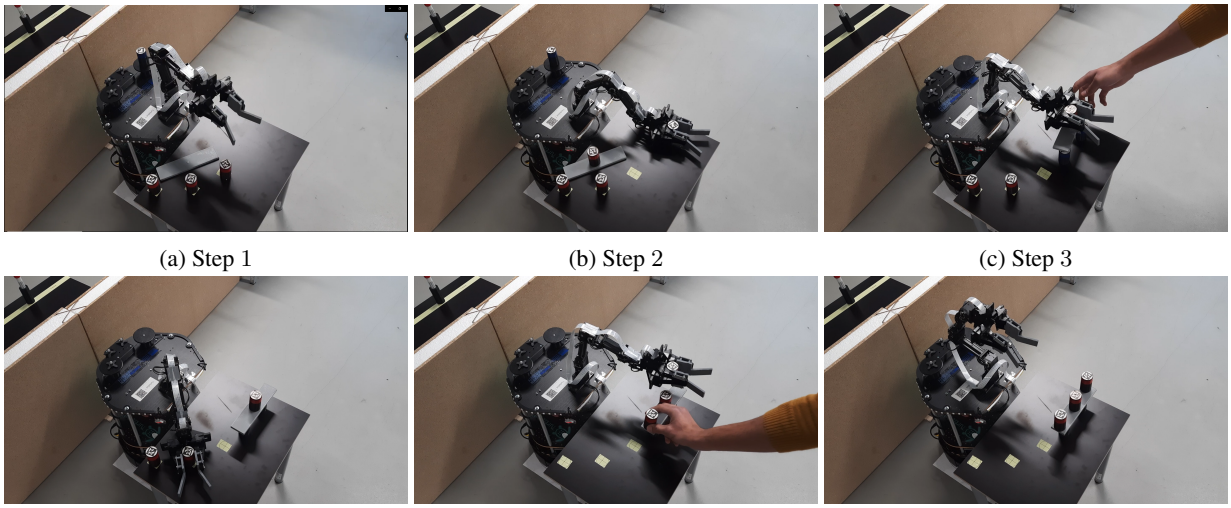
Figure 9: Part of the EFSM Action Interface for a *pick_and_place_two_agents* PDDL action represented as a graph.

In future work, the methodology will be validated with further scenarios. In this sense, new planning situations will be encoded as planning problems in PDDL. The robustness and resilience of the planning and execution framework will also be tested with planning situations in which changing capabilities of the PwD enforce the generation of new plans, or when PwD fail in executing their assigned tasks. For such situations, external sensors will track the execution of the actions and inform the Supervisor about any anomalies.

References

Bezrucav, S.-O.; Canal, G.; Cashmore, M.; and Corves, B. 2021. An Action Interface Manager for ROSPlan. In Awaad, I.; Finzi, A.; and Orlandini, A., eds., *9th ICAPS Workshop on Planning and Robotics (PlanRob)*. Online.

Bezrucav, S.-O.; and Corves, B. 2020. Improved AI Planning for Cooperating Teams of Humans and Robots. In Cashmore, M.; Orlandini, A.; and Finzi, A., eds., *Proceedings of the 8th ICAPS Workshop on Planning and Robotics (PlanRob)*. Nancy, France.



(a) Step 1 (b) Step 2 (c) Step 3

(d) Step 4 (e) Step 5 (f) Step 6

Figure 10: Six steps of the real-world human-robot collaboration scenario used to validate the proposed methodology.

Bokranz, R.; Landau, K.; and Becks, C., eds. 2006. *Produktivitätsmanagement von Arbeitssystemen: MTM-Handbuch*. Stuttgart: Schäffer-Poeschel.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; and Carreras, M. 2015. ROSPlan: Planning in the Robot Operating System. In Brafman, R., ed., *Proceedings of the 25th International Conference on Automated Planning and Scheduling*, 333–341. Jerusalem, Israel: AAAI Press. ISBN 9781577357315.

Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In Brafman, R., ed., *Proceedings of the 20th International Conference on Automated Planning and Scheduling*, 42–49. Menlo Park, Calif.: AAAI Press. ISBN 978-1-57735-449-9.

Ghallab, M.; Knoblock, C.; Wilkins, D.; Barrett, A.; Christianson, D.; Friedman, M.; Kwok, C.; Golden, K.; Penberthy, S.; Smith, D.; Sun, Y.; and Weld, D. 1998. PDDL - The Planning Domain Definition Language.

Hüsing, E.; Weidemann, C.; Lorenz, M.; Corves, B.; and Hüsing, M. 2021. Determining Robotic Assistance for Inclusive Workplaces for People with Disabilities. *Robotics*, 10(1).

Institut Arbeit und Qualifikation der Universität Duisburg-Essen. 2023. Arbeitslosenquote von Menschen mit Schwerbehinderungen 2012 - 2020. Accessed on March 3, 2023.

Kambhampati, S.; and Srivastava, B. 1996. Universal Classical Planner: An algorithm for unifying State-Space and Plan-Space planning.

Müller, R.; Vette, M.; and Mailahn, O. 2016. Process-oriented Task Assignment for Assembly Processes with Human-robot Interaction. *Procedia CIRP*, 44: 210–215.

Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; and Ng, A. 2009. ROS: an open-source Robot Operating System. In Matsuoka, Y.; Laschi, C.; and Kaneko, M., eds., *International Conference on Robotics and Automation Workshop on Open Source Software*, volume 3. Kobe,Japan.

Ranz, F.; Hummel, V.; and Sihn, W. 2017. Capability-based Task Allocation in Human-robot Collaboration. *Procedia Manuf.*, 9: 182–189.

Roncone, A.; Mangin, O.; and Scassellati, B. 2017. Transparent role assignment and task allocation in human robot collaboration. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1014–1021. Marina Bay Sands, Singapore: IEEE.

Sucan, I. A.; and Chitta, S. 2020. MoveIt Motion Planning Framework.

Takata, S.; and Hirano, T. 2011. Human and robot allocation method for hybrid assembly systems. *CIRP Ann.*, 60(1): 9–12.

Tsarouchi, P.; Matthaiakis, A.-S.; Makris, S.; and Chryssolouris, G. 2017. On a human-robot collaboration in an assembly cell. *Int. J. Computer Integr. Manuf.*, 30(6): 580–589.

United Nations. 2006. Convention on the Rights of Persons with Disabilities. Available at: https://www.un.org/development/desa/disabilities/convention-on-the-rights-of-persons-with-disabilities.html (Accessed on: 1st March 2023).

Weidemann, C.; Hüsing, E.; Freischlad, Y.; Mandischer, N.; Corves, B.; and Hüsing, M. 2022. RAMB: Validation of a Software Tool for Determining Robotic Assistance for People with Disabilities in First Labor Market Manufacturing Applications. *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 1: 2269–2274.