Evaluation of a Robot Programming Framework for Non- Experts Using Symbolic Planning Representations

Ying-Siu Liang, Damien Pellier, Humbert Fiorino, Sylvie Pesty

▶ **To cite this version:**

Ying-Siu Liang, Damien Pellier, Humbert Fiorino, Sylvie Pesty. Evaluation of a Robot Programming Framework for Non- Experts Using Symbolic Planning Representations. 26th IEEE International Symposium on Robot and Human Interactive Communication, Aug 2017, Lisbon, Portugal. hal-01583887

HAL Id: hal-01583887
https://hal.science/hal-01583887

Submitted on 8 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Evaluation of a Robot Programming Framework for Non-Experts using Symbolic Planning Representations

Ying Siu Liang, Damien Pellier, Humbert Fiorino, and Sylvie Pesty

***Abstract*— Cobots (collaborative robots) are revolutionising industries by allowing robots to work in close collaboration with humans. But many companies hesitate their adoption, due to the lack of programming experts. In this work, we evaluate a robot programming framework for non-expert users, that requires users to teach action models expressed in a symbolic planning language (PDDL). These action models would allow the robot to leverage modern automated planners to achieve any user-defined goal. We conducted qualitative user experiments with a Baxter robot to evaluate the non-expert user's understanding of the symbolic planning language and the usability of the framework. We showed that users with little to no programming experience can adopt the symbolic planning language, and use the framework.**

I. Introduction

Robots have been working in close collaboration with humans. Cobotic systems [1] have been adopted in several industries from the food-processing industry, to aeronautics, to the health industry. However, many companies remain robot resistant, as they lack programming experts to fully exploit the robots. Programming by Demonstration (PbD) allows non-experts to teach robots new skills by demonstrating a task, without writing any code [2]. It is an intuitive robot programming approach, with the goal to refine the robot's performance, by providing repetitive demonstrations. However, in existing PbD implementations the robot learns an action sequence [3], [4], rather than atomic actions that can be reused independently. Teaching full action sequences is often complicated and time-consuming, as the robot has to be demonstrated a new sequence, whenever the goal changes.

In our previous work [5], we addressed the question "Can non-experts teach a robot atomic actions, which can be reused to automatically generate novel action sequences?" We proposed a framework that combines PbD and Automated Planning [6], where the robot learns action models by demonstration, and the problem of finding an action sequence is delegated to a planner. The robot programming process consists of steps:

1) the non-expert user demonstrates atomic actions to the robot, and teaches *action models*, expressed in a symbolic planning language (STRIPS [7]),
2) the robot uses these action models with an automated planner to generate solutions to user-defined goals,
3) the user can revisit the taught action model to refine them.

Y. S. Liang, D. Pellier, H. Fiorino, and S. Pesty are with the Univ. Grenoble Alpes, CNRS, Grenoble INP*, LIG, F-38000 Grenoble France, *Institute of Engineering Univ. Grenoble Alpes {ying-siu.liang, damien.pellier, humbert.fiorino, sylvie.pesty}@imag.fr

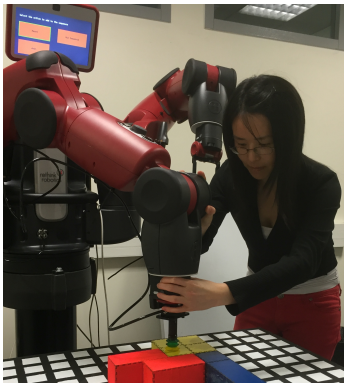

Fig. 1. Programming by Demonstration of a move action by kinesthetic manipulation of the Baxter robot.

As the framework is designed for non-expert users, we first need to assess the framework's usability and the user's adoption of the symbolic planning language. We evaluate the user's ability to construct symbolic action models, in terms of preconditions and effects, used by automated planners. Hence, we conducted two qualitative user experiments, to assess the following questions:

Q1 How do non-expert users adopt the automated planning language with its action model representation?

Q2 Can users teach a robot action models for automated planning using the robot programming framework?

The remaining of this paper is organised as follows. In Section II, we discuss the background and related work. We provide an overview of the existing robot programming framework in Section III. In Section IV, we discuss the basic research questions that are answered in the following sections. Section V presents experiments and results to assess the user's adoption of the symbolic planning language. Similarly, we assess the overall framework's usability in Section VI. In Section VII, we discuss the overall results for both experiments. Finally, we conclude and discuss future work in Section VIII.

II. Related Work

Programming by Demonstration (PbD) provides an intuitive medium that allows non-experts to teach a robot new skills. Research in PbD can be split into two categories [2]: trajectory encoding and symbolic encoding. The first approach generalises low-level representations of the skill, by using statistical modeling [8], [9] or dynamical systems [10], but does not allow representation of complicated high-level skills. The second approach allows this high-level

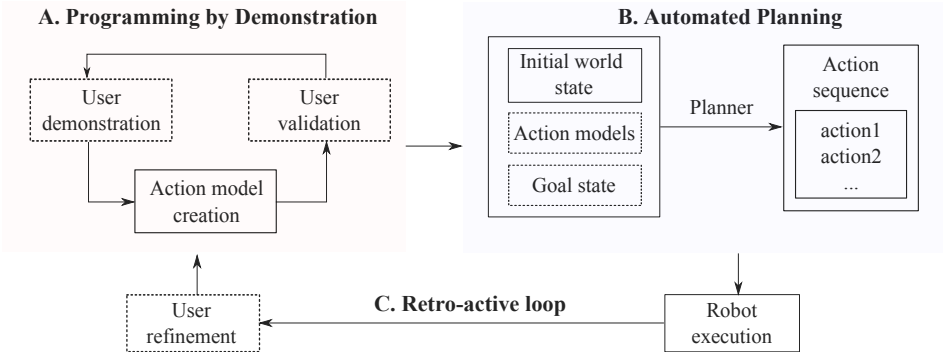

Fig. 2. A robot programming framework for non-expert users: the user teaches action models, which are used by the robot with an automated planner, to generate an action sequence to achieve a goal. After an unsuccessful robot execution, the user can refine the taught action models (dotted lines indicate user actions, solid lines indicate robot actions).

representation, by segmenting a complex skill into a set of *predefined* atomic actions. [11] learn a sequential task plan from demonstrations by symbolically representing action models. However, these action models are already pre-programmed into the robot.

Moreover, the robot often tries to learn the teacher's goal [4], [12] and reproduces it from different initial states. Once the goal changes, the teacher needs to demonstrate a new action sequence. This can be very time-consuming, especially in industrial environments, where the robot's tasks change, depending on the product specification. Symbolic planners can generate action sequences automatically, using atomic actions. Zita Haigh et al. [13] implement a planning algorithm in a robot, but do not teach actions by demonstration. Our approach allows non-expert users to teach action models by demonstration, that can be reused by a symbolic planner. Non-expert users need to interact with the robot to create symbolic action models. As far as we know, there are no user studies in this area, which deal with the user's understanding of the symbolic planning language and action model representation.

III. Overview

We work on the assumption that non-experts can teach robots symbolic planning representations by demonstrating atomic actions. In [5], we designed a 3-step robot programming framework. Figure 2 illustrates the following steps:

A. Programming by Demonstration: user demonstrates atomic actions and teaches symbolic action models, in terms of preconditions and effects.
B. Automated Planning: robot uses action models to generate an action sequence to achieve a goal.
C. Retro-active Loop: user refines the learned action models in case of ambiguities.

In the following sections, we discuss each step in more detail. We refer the reader to a video of the framework: `https://youtu.be/DTm2YjiSNQM`

A. Programming by Demonstration: teaching action models

Programming by demonstration (PbD) techniques generally consist of an iterative process. The user demonstrates an atomic action, such as moving a cube from an initial position A to a final position B (`move cube A B`). An action execution results in a change in the world state, such as the cube's position. The robot observes the changes before and after the action demonstration (Fig. 3 top), and extracts the relevant preconditions and effects to build a generalised action model, expressed in a symbolic planning language (Fig. 3 bottom). The user validates the learned action model, and provides additional demonstrations if necessary.

The robot learns from multiple demonstrations to generalise trajectories and high-level conditions. In the framework, we assume that the learned action trajectory is independent of the trajectory performed by the teacher. Dynamic Movement Primitives (DMP) can cope with the generalisation of a demonstrated trajectory [14]. To learn high-level conditions, the robot uses a perception system (e.g. SIFT [12] or a database of object features [15]) that recognises object properties in the state of the world. In our experiments, we implemented a simple python algorithm with integrated functionalities of the Robot Operating System (ROS), to detect and move objects, based on their colour. Feature-based algorithms, such as k-means clustering [16], can be used to generalise over high-level conditions.

Existing PbD approaches try to teach the robot from a small number of demonstrations [3], [17]. We propose an interactive learning approach, where the user can directly modify the learned action models using a user-interface.

B. Automated Planning: using action models

The robot uses the learned symbolic action models in combination with an automated planner to achieve user-defined goals. Automated planners try to model the robot's strategies, when operating in diverse environments [6]. Figure 4 illustrates an example of a planning problem. Given a description of the world state, i.e. objects, types, properties (Fig. 4a,b), a set of possible atomic actions, and a goal state (Fig. 4d), a planner generates a sequence of actions (Fig. 4c), which guarantees the transition from the initial world state to the goal state. Predicates are used to describe actions and object properties, such as *"red cube is at position A"*: they can be *instantiated* (`at redCube A`) or *generalised* (`at`

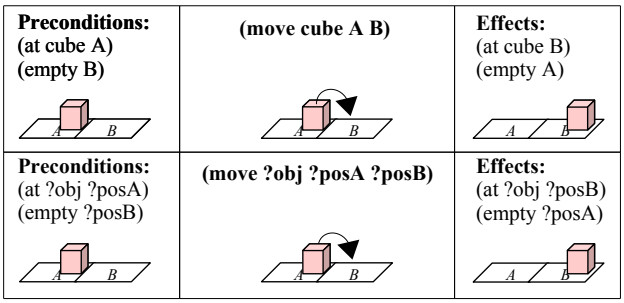

Fig. 3. Action model representation of a move action in terms of preconditions and effects: demonstrated action model for a cube (top), and generalised action model for any object, variables are prefixed with '?' (bottom).

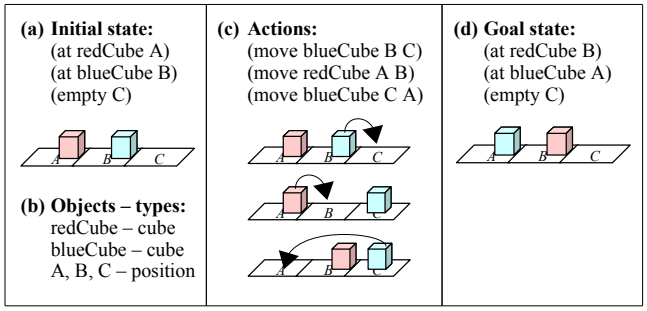

Fig. 4. Definition of a planning problem (a) properties describing the initial world state (b) object names and their types (c) instantiated actions (d) properties describing the goal state.

`?cube ?posA)`, such that variables are prefixed with '?'.

To allow a correct transition between different states of the world, actions are defined in terms of preconditions and effects (Fig. 3). For a move action `(move cube A B)`, preconditions are the states required to perform the action `(at cube A)`, and effects are the states obtained after the action `(at cube B)`. Classical planning algorithms use the Planning Domain Definition Language (PDDL) [6] as their standard encoding language, which extends the STRIPS [7] formalism with greater expressivity, such as type structures (Fig. 4b). An example of a generalised move action in PDDL, for moving an arbitrary cube, from one position to another, is given below (`?cube`, `?posB`, `?posA` are variables for cube, initial position, final position respectively):

```
(:action move
 :parameters (?cube ?posA ?posB)
 :precondition (and (at ?cube ?posA)
                    (empty ?posB))
 :effect     (and (at ?cube ?posB)
                  (empty ?posA)))
```

The initial world state is automatically recognised by the robot using the same perception system as in the PbD phase. The user can change the initial object configuration of the world state, and enter an arbitrary goal, that can be achieved using the taught action models. If the user changes the goal, a new action sequence is generated by the planner. Generating a plan under different initial states allows the user to test the created action models. There exist various planners that can be used for symbolic planning with robots: Metric-FF [18], or a fast downward planner [17].

C. Retro-active Loop: refining action models

The execution to a new context is an important step to test the taught action models. It is likely that the execution of the plan does not produce the desired outcome, especially if the plan is executed in a context different to the demonstration (e.g. different object colour or position). Missing preconditions or effects for an action model can lead to suboptimal or non-existing solutions and will need to be corrected by the user. The retro-active loop allows the user to revisit the created action models. Knowledge engineering tools provide a user-interface to facilitate this process of modifying action models used for automated planners. They often provide useful functionalities for dynamic testing, model checking and visualisation [19], where the quality of the models generated does not depend on the user's expertise.

IV. Basic Research Questions and Their Evaluations

To evaluate the framework's usability for non-expert users, we conducted two qualitative user experiments to respond to the questions mentioned in section I:

- **Q1** How do non-expert users adopt the automated planning language with its action model representation? (Section V)
- **Q2** Can users teach a robot action models for automated planning using the robot programming framework? (Section VI)

The experimental context was designed around a Baxter robot (Fig. 1). In both experiments, we included elements to assess the user's understanding of action models used by automated planners. Understanding this symbolic representation is a key requirement to use the framework. In the following sections, we briefly outline the experimental setup, measurements and results for each experiment.

V. How do non-expert users adopt the automated planning language with its action model representation?

In this experiment, users were introduced to a symbolic planning language (a simplified version of PDDL), involving the STRIPS formalism with type structures (Sec. III-B). Users were instructed to describe world state configurations to the robot. The goal was to assess the user's adoption of the planning concepts (object types, properties, generalised properties, action models) and to verify that the symbolic planning language is appropriate for non-expert users.

A. Experimental Setup & Participants

We recruited 10 participants (1 male, 9 female), who were Sociology students at Université Grenoble Alpes. 3 participants reported no programming experience, 6 had experience with office productivity software ('beginner'),

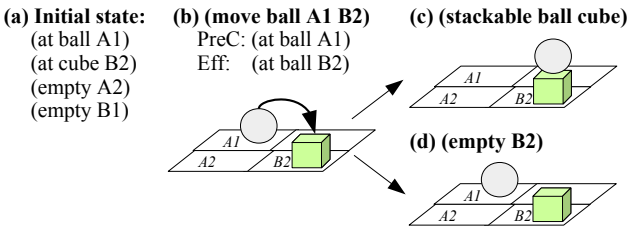

Fig. 5. Users were instructed to provide a description of (a) the initial state of the world and (b) an initial move action model. They derived additional preconditions for moving the ball from position A1 to B2: (c) *(stackable ball cube)*: the ball can be stacked onto the cube, and (d) *(empty B2)*: if the ball cannot be stacked, the target position should be empty.

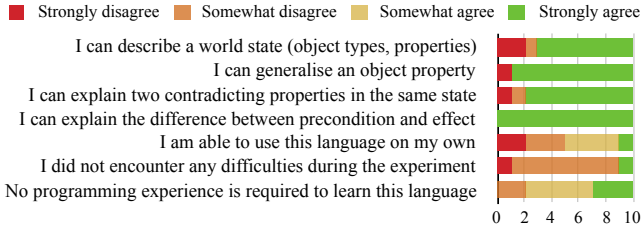

Fig. 6. Summary of questionnaire responses: Extract of the 26 questions on the user's understanding, after the introduction to the automated planning language (Sec. V).

and 1 had previously taken a programming course before ('advanced'). The experimental setup consisted of a 2x2 board, 2 cubes, 1 ball, and 1 ball recipient. The participants were given sheets with empty tables to complete for each task. Each participant was allocated 1 hour, but the average duration of the experiment was 49 minutes. The participants' behaviour was observed by the experimenter and recorded on camera.

B. Experimental Design & Measurements

Users were told that they needed to use a symbolic planning language, to describe the state of the world, and the semantic meaning of actions to the robot. Throughout the experiment, users were faced with three different scenarios. Figure 5 shows an example of the experimental design, where the robot's action was to move the ball to an occupied position (B2). We evaluated their capability, to apply the planning language to different situations. The experiment consisted of the following phases:

- **Training:** Users were presented with an initial world state and the symbolic planning language, to describe the object types and properties. Users were shown how to model a simple move action in terms of preconditions and effects (Fig. 3). Additionally, they were introduced to the concept of *generalised* properties and *generalised* actions (Sec. III-B).
- **Experimental test:** Users were presented a new world state and instructed to provide a description to the robot in the symbolic planning language (Fig. 5a). After defining an initial move action model (Fig. 5b), users were faced with 3 different scenarios to refine the preconditions. Users derived a `(stackable ball cube)` property (Fig. 5c), which allowed a ball to be stacked on top of a cube. When this property did not hold, users proposed the *empty* property (Fig. 5d), which the robot needed to verify before the action execution. At each step, users had to give the generalised representation of the properties and action models.
- **Planning:** Users were presented a description of a new state of the world and a goal state. Then, they had to define an action sequence, that allows the transition to the goal state (similar to Fig. 4c), and explain their reasoning using the symbolic action model representation.
- **Questionnaire:** Users were given a questionnaire containing 26 questions related to their experience, as well as their understanding of the learned planning language (Fig. 6).
- **Debriefing:** Throughout the experiment, users were asked open-ended questions ("What properties do you observe in the current world state?"), so that they were guided as little as possible, and their responses were unbiased. When the participant struggled to find an answer, the experimenter guided the participant into a possible direction ("Why can the ball not be placed on the cube?").

C. Results

We did not observe any significant differences in the performance of users with or without programming experience. 9 (out of 10) participants found the symbolic representation of properties and actions easy to understand. During the experimental test, the majority (9) of the participants managed to describe the complete world state using the correct syntax. When faced with different scenarios to refine the move action model, half of the participants struggled to formalise the *stackable* condition in the symbolic language. They provided alternative formulations related to the cube's properties ("if the cube can hold the ball"). However, once the condition was defined, the majority (8) of the participants had little to no difficulties generalising this property. Due to time constraints, only half (5) of the participants were presented the planning phase. All 5 encountered no problems when defining the action sequence to achieve the given goal.

In the questionnaire (Fig. 6), the majority (9) of the participants understood the notion of states and object properties. 8 correctly pointed out two properties that could not exist in the same state (e.g. `(empty A)` and `(at cube A)`). All participants gave correct explanations for preconditions and effects of action models, and provided correct examples. 9 participants encountered difficulties during the experiment, 6 stated problems with formalising the language, especially at the beginning of the experiment. Half of the participants believed that they could apply this language on their own. No participants believed that an 'expert' programmer was required to learn the symbolic planning language, but 7 participants believed that the minimum requirement was a 'beginner' programming level, while 3 believed that no programming experience was required.

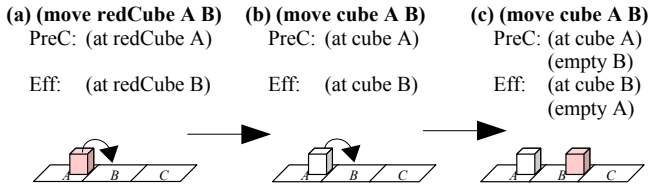

Fig. 7. Continuous refinement of the move action model: (a) initial action model learned by demonstration, (b) action model for all cubes of any colour, (c) action model with an additional condition, if the target position is occupied and cubes can not be stacked.

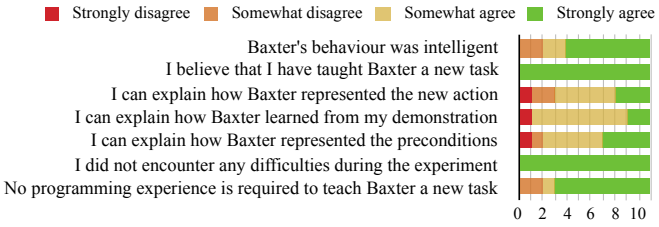

Fig. 8. Summary of questionnaire responses: Extract of 18 questions on the user's understanding, after the experiment to teach action models by demonstration (Sec. VI).

VI. Can Users teach Action Models for Automated Planning using the Robot Programming Framework?

In this experiment, users were presented the robot programming framework (Sec. III), and had to teach action models by kinesthetic manipulation of a Baxter robot (Fig. 1). Users were instructed to demonstrate an atomic action and assign preconditions and effects. The goal was to assess the framework's usability and difficulties encountered, when teaching action models.

A. Experimental Setup & Participants

We recruited 11 participants (7 male, 4 female), who were students and staff members at Université Grenoble Alpes. 6 participants reported programming experience with office productivity software ('beginner'), 2 had previously taken a programming course before ('advanced'), and 3 were pursuing studies in Computer Science ('expert'). The experiments were conducted using a Baxter robot (Fig. 1), mounted with a partial implementation of the framework (e.g. 'learn a new action', 'find an object', 'execute an action sequence'). We used the Wizard-of-Oz technique to simulate the remaining functionalities (e.g. 'generate action sequence'). Participants operated on a table with 2 positions (A and B), 2 cubes (blue and red), that represented parts on an assembly line. Each participant was allocated 1 hour, but the average duration was 29.5 minutes. The experiments were recorded, while the participant interacted with the robot.

B. Experimental Design & Measurements

The experiment was set in a simulated assembly line, where objects of the same shape, but different colour arrived consecutively at position A. Users had to teach Baxter the action for moving an object from position A to position B, where objects should not be stacked. Throughout the experiment, users were faced with two different scenarios, where Baxter had to apply the learned action model. We evaluated the user's capability to refine action models, when faced with different situations, and wanted to assess the framework's overall usability. The experiment consisted of the following phases:

- **Training:** Users were shown how to manipulate Baxter's arm, and given time to familiarise themselves with the kinesthetic manipulation.
- **Experimental test:** Users were instructed to teach Baxter a move action of a cube. Then, they were presented the action model, with preconditions and effects, that Baxter learned from the demonstration (Fig. 7a). In the following, users were faced with two different scenarios, to refine the conditions of the action model. Starting with the initial action model for a red cube, users modified the conditions, so that it was applicable to all cubes of any colour (Fig. 7b), and when the target position was occupied (Fig. 7c). At each step, users observed how Baxter applied the learned action in the new scenario. When Baxter failed to execute the action, users had to refine the conditions of the action model.
- **Planning:** Users were presented a new scenario, where Baxter was instructed to achieve a new goal using the learned action model. The new goal was to switch the positions of two cubes on the table. Baxter demonstrated this by executing an action sequence generated by an automated planner.
- **Questionnaire:** Users were given a questionnaire containing 18 questions related to their experience (Fig. 8).
- **Debriefing:** Users were questioned about their expectations on Baxter's behaviour, before applying the learned action model to a new scenario. Users were asked open-ended questions ("What will Baxter do when applying the learned action model?"), so that their responses were unbiased. When they encountered failure scenarios, they were asked to reason about Baxter's behaviour.

C. Results

All (11) users were satisfied with the PbD process and Baxter's abilities to learn and reproduce the demonstrated move action. All users understood the learned action model and managed to adopt the notion of preconditions and effects easily. As expected, no users pointed out the missing conditions (Fig. 7c), when asked for improvements of the initial action model (Fig. 7a). Even users who were 'experts' and who had heard of automated planning before, did not manage to create a complete action model from the start. However, all users detected the missing condition easily, when faced with the relevant failure scenarios.

In the final phase, users with no experience in automated planning (8) did not expect Baxter to solve the permutation problem, and agreed unanimously that it acted in an intelligent manner, when it did. At the end of the experiment, all users believed that they had taught Baxter a new task and the majority understood the representation of the action

models well. No users encountered any difficulties during the experiment. 8 participants believed that no programming experience was required to teach Baxter a task (Fig. 8).

VII. Discussions and Limitations

In both experiments, we did not observe a significant difference in the performance between users with different programming experience. The majority of users had issues formulating the logical representations of object properties used in action models. In the first experiments, users had difficulties formulating a single condition (e.g. `(stackable ball cube)`), but stated equivalent conditions (*'only place the ball, if it is stackable on the cube'*). Similarly, in the second experiment, users formulated the missing precondition (*'position B is empty'*) with other equivalent conditions (*'Do not place the object on position B, if it is occupied'*). This means, that users should be provided with a predefined set of conditions that can be added to the action model.

Some users made wide assumptions about the robot's capabilities. In the second experiments, when both arrival and departure positions were occupied (Fig. 7c), less half of the users (5) expected Baxter to consider the occupied position, even though the condition was not mentioned in its action model. This is a common problem in PbD solutions as there is a difference in the perception of the robot's intelligence perceived by its teacher [20] and can be addressed by reproducing the learned task in a new context and verifying the robot's knowledge base, as we did throughout the experiment.

With these two qualitative experiments, we showed that the automated planning language could easily be adopted by users without any programming background. Moreover, the action model representation, in terms of preconditions and effects, seems to be intuitive for non-expert users. However, these initial experiments only provide us with an idea of how the users might perceive the proposed framework. We intentionally limited the set of concepts that are necessary to use the framework to the bare minimum. Further experiments should test scalability and address more complicated actions involving separate control groups (experts vs non-experts) in less structured scenarios.

VIII. Conclusions

In this work, we evaluated the robot programming framework for non-experts proposed in [5] in terms of qualitative experiments. The framework combines two techniques, Programming by Demonstration and Automated Planning, and uses an action model representation, in terms of preconditions and effects. We showed that non-expert users understood the symbolic planning concepts well, despite learning about them for the first time. Overall, the robot programming process was considered to be very intuitive and easily understood by users.

Now that we have verified the usability of the framework, future work will focus on its full implementation. This involves using state-of-the-art techniques to implement functionalities, that were simulated with the Wizard-of-Oz technique during the experiment. This can be divided into three aspects: Creating an user-interface to facilitate the symbolic language construction, learning low-level motion trajectories [14], and, more importantly, learning high-level representations of actions in terms of preconditions and effects [16], [17], that can be used by symbolic planners.

References

[1] J. E. Colgate, W. Wannasuphoprasit, M.A. Peshkin, "Cobots: Robots for Collaboration with Human Operators", ASME Dynamic Systems and Control Division, 433-440, 1996.

[2] A. Billard, S. Calinon, R. Dillmann, S. Schaal, "Survey: Robot Programming by Demonstration", Handbook of Robotics, chapter 59, 2008.

[3] E. M. Orendt, M. Fichtner, D. Henrich, "Robot Programming by Non-Experts: Intuitiveness and Robustness of One-Shot Robot Programming," IEEE International Symposium on Robot and Human Interactive Communication, 192-199. 2016.

[4] L. Peppoloni, A. Di Fava, E. Ruffaldi, C. Avizzano, "A ROS-integrated architecture to learn manipulation tasks from a single demonstration." International Symposium on Robot and Human Interactive Communication, 537-542, 2014.

[5] Y.S. Liang, D. Pellier, H. Fiorino, S. Pesty, "A Framework for Robot Programming in Cobotic Environments: First User Experiments", International Conference on Mechatronics and Robotics Engineering, 2017.

[6] M. Ghallab, D. Nau, P. Traverso, "Automated planning: theory and practice", Morgan Kaufmann Publishers, Elsevier, 2004.

[7] R.E. Fikes, N.J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", Joint Conference on Artificial Intelligence, 608-620, 1971

[8] S. Calinon, F. Guenter, A. Billard, "On Learning, Representing and Generalizing a Task in a Humanoid Robot", IEEE Transactions on Systems, Man and Cybernetics, 286-298, 2007.

[9] J. Aleotti, S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration", Robotics and Autonomous Systems, 409-413, 2006.

[10] A. Ijspeert, J. Nakanishi, S. Schaal, "Learning control policies for movement imitation and movement recognition", Neural Information Processing System, 2003.

[11] H. Veeraraghavan, M. Veloso, "Teaching sequential tasks with repetition through demonstration", International Joint Conference on Autonomous Agents and Multiagent Systems, 1357-1360, 2008.

[12] S.R. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, D.G. Caldwell, "Learning symbolic representations of actions from human demonstrations", International Conference on Robotics and Automation, 3801-3808, 2015.

[13] K. Zita Haigh, M. Veloso, "Interleaving planning and robot execution for asynchronous user requests", International Conference on Intelligent Robots and Systems, 148-155, 1996.

[14] P. Pastor, H. Hoffmann, T. Asfour, S. Schaal, "Learning and generalization of motor skills by learning from demonstration", International Conference on Robotics and Automation, 763–768, 2009.

[15] M. Mason, M. Lopes, "Robot self-initiative and personalization by learning through repeated interactions", International Conference on Human-Robot Interaction, 433-440, 2011.

[16] Y. Mollard, T. Munzer, A. Baisero, M. Toussaint, M. Lopes, "Robot programming from demonstration, feedback and transfer", International Conference on Intelligent Robots and Systems, 1825-1831, 2015.

[17] N. Abdo, H. Kretzschmar, L. Spinello, C. Stachniss, "Learning Manipulation Actions from a Few Demonstrations", International Conference on Robotics and Automation, 1268-1275, 2013.

[18] R. Cubek, W. Ertel, G. Palm, "High-level learning from demonstration with conceptual spaces and subspace clustering", International Conference on Robotics and Automation, 2592-2597, 2015.

[19] R.M. Simpson, D.E. Kitchin, T.L. McCluskey, "Planning domain definition using GIPO", The Knowledge Engineering Review, 117-134, 2007.

[20] R. Toris, H.B. Suay, S. Chernova, "A practical comparison of three robot learning from demonstration algorithms", International Conference on Human-Robot Interaction, 261-262, 2012.