Towards Goal-driven Behaviour Control of Multi-Robot Systems

Christopher-Eyk Hrabia, Stephan Wypler and Sahin Albayrak
Technische Universität Berlin, DAI-Lab,
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
christopher-eyk.hrabia@dai-labor.de

***Abstract*—The task-level control of mobile robots in uncertain environments demand high adaptivity and cognitive capabilities that allow for flexible decision making and planning. This work introduces the ROS Hybrid Behaviour Planner (RHBP). RHBP combines the advantages of a reactive and adaptive behaviour-based layer with a formal symbolic planner that enables appropriate goal pursuance. Here, we present the first development stage of our system that enables more adaptive multi-robot decision making and planning in the widely used Robot Operating System (ROS) ecosystem.**

***Index Terms*—Behaviour-based Planning, Behaviour Networks, Multi-Robot Systems, Hybrid Planning, Decision-Making, Robot Operating System,**

I. Introduction

Today's robotic systems are able to support or replace humans in a wide variety of tasks, they get more feature-rich and are capable to deal with various situations and environmental conditions. Even more, they are leaving the friendly, well structured world of automation and are facing the challenges of a dynamic world in house-holds, offices, or public space. This becomes especially challenging in case of several cooperating mobile robots, useful in application of logistic tasks, disaster rescue, or precision farming. In consequence robots have to deal with uncertain conditions and unforeseen changes in the environment autonomously. This demand for systems that put emphasis on robustness and adaptivity rather than on behaving in an optimal fashion. As discussed in [1], adaptivity in general, fast and flexible decision making, and planning in particular, are crucial capabilities for autonomous robots. From the engineers perspective it is very difficult to develop autonomous systems that are able to react appropriately to unforeseen changes while still pursuing intended goals. Especially in the widely applied and popular Robot Operating System (ROS) ecosystem [2] developers are mostly using pre-scripted, non-adaptive methods of describing the high-level robot behaviours or tasks. The above methodology requires envisioning all appropriate behaviour execution sequences, states and state transitions in advance. More dynamic approaches providing adaptivity and goal pursuance at the same time are missing.

In order to fill this gap we are working on a new hybrid approach with tight ROS integration that incorporates features from reactive behaviour based systems and classical planning. In this paper the first development stage of our solution is presented that can be used on single and multi-robot systems.

The remainder of the paper is structured as follows. In Section II, we discuss related approaches, Section III presents our general concept including the high-level architecture, with further detail about the hybrid approach in Section IV and V. The ROS integration is explained in Section VI and first experiments and results are presented in Section VII. Finally, we summarise our work and outline the necessary future steps in Section VIII.

II. Related Work

In the ROS community the most commonly used approaches for task-level or behaviour control are rule based and reactive. A popular package is *SMACH* that allows the user to build hierarchical and concurrent state machines (HSM) [3]. All kind of state machine based approaches have the problem that a decision or reaction can only be given if a state transition was already modelled in advance. The more task-oriented behaviour trees, available in the *pi_trees* package [4], are another popular alternative. Even though behaviour trees allow for more reusable behaviour implementations and some higher-level abstractions, by using special operators like sequencer and selector, the path of execution is still preprogrammed and is likely to fail in dynamic environments. The reason is that it is difficult to design such system without missing possible useful dynamic transitions.

More flexible is the BDI-based implementation *CogniTAO* [5] available in the *decision_making* package that also includes modules for the creation of finite and hierarchical state machines. CogniTAO is targeting high-level control of multi-robot systems in dynamic environments. In CogniTAO the user defines behaviours, called plans, as decision-graphs with start and end conditions that are executed and selected by protocols. The selection and decision process is synchronised through a team of robots. The concept is more suitable for uncertain environments because the execution sequence is not fixed and the selection of behaviours (plans) is based on conditions. Nevertheless, it is still difficult to define mission or maintenance goals and there exist only simple protocols for plan selection.

Outside of the ROS community we can find other reactive approaches, for instance the behaviour network architecture [6]. Here, the behaviours are connected with each other based on their preconditions and effects. The executed behaviour is dynamically selected based on a utility function.

The utility function is calculating the positive influence on a goal for each behaviour, called activation that is spread through the network. Different extensions of this approach introduce learning capabilities and multi-robot application [7] or concurrency and non-binary preconditions [8]. The hierarchic version of Allgeuer is available as a ROS package, but simplified in the way that it only relies on pre-computed static inhibitions of conflicting behaviours. A general issue is that network parameters need to be properly tuned towards a specific application to avoid the risk of getting stuck in cycles, since it is difficult to express a required execution order.

Symbolic planners in the tradition of STRIPS [9], like hierarchical task networks (HTN) [10] or further advanced implementations, like graph-based approaches [11] or heuristic planners [12] are very good in directing towards a goal and determining a possible execution order. ROSPlan, a generic symbolic task planning framework in this inspiration was recently presented by Cashmore et al. [13]. In fact, ROSPlan is a PDDL dispatching framework that allows to map PDDL actions to ROS actions. It also includes a OWL based knowledge base, which is used to automatically generate the problem file, and an observer that determines situations requiring replanning. However, the model of the domain has to be provided in PDDL and the mapping has to be verified manually.

However, all pure symbolic planners have problems under uncertain conditions because of high computational costs and calculating alternative decisions in case of unreachable goals.

In order to combine the advantages of both, reactive behaviour approaches, being fast and more flexible, and symbolic planning, being more goal-directed, some researchers have proposed hybrid architectures for single robot systems. An early attempt was given by Hertzberg et al. [14], who proposed a low level behaviour controller using two differential equations, one for creating the actuator control signal and one controlling the activation. The integration of a special term in the activation equation enabled external influences from an operator like a human or a symbolic planner. Here, the reactive behaviour layer is still operational on its own and the model for the planner has to be provided independently. A more recent hybrid approach [15] models each high-level task as an independent behaviour network that is selected by a planner. This architecture is strongly goal-directed, but looses flexibility for dynamic adaptation since behaviours of different networks can not be combined with each other.

Hybrid approaches that combine behaviour networks with symbolic planning are a promising direction to create robots that are adaptive and goal-driven at the same time. The presented literature includes many fruitful ideas [8], [14], [7], [15], [6], even so they are not integrated into one solution, and are not all available for the ROS community. Furthermore, extending the concept towards multi-robot application, enabling learning capabilities and integrating self-organisation mechanisms are open issues. A comparison of the features of the different presented approaches is shown in Table I.

III. Architecture

In order to address the mentioned open challenges, we are working on an enhanced system that supports reactive and adaptive behaviours by utilising behaviour networks on a lower level and combine it with classical symbolic planning on a higher level for a strong alignment towards the mission goal and enabling a mixed bottom-up and top-down control. Here, top-down refers to the definition and pursuing of goals for the entire system, while lower level behaviours can be implemented bottom-up and integrated into the higher-level decision-making and planning.

Figure 1 shows our target architecture. It includes a centralised meta-level symbolic planner that initially or from time to time provides sub-goals for individual robots. The meta-level symbolic planner relies on available behaviours and information it receives from each robot. Each robot has an own symbolic planner that influences its lower level behaviour network and supports the long-term goal-fulfilling. Therefore, the robot remains independent from the centralised planner and is still able to operate in case of interrupted or unavailable communication. Since our architecture is targeting multi-robot applications it requires some coordination amongst the robots, for instance by applying nature-inspired swarm algorithms, the meta-level symbolic planner is also controlling the coordination and adaptation mechanism selector based on its higher level goals. This component is responsible for the determination of an appropriate coordination mechanism for the multi-robot system that is supporting the intended goals. Based on the selected mechanism it is influencing on behaviour parameters.

An important point is that the centralized components for planning, coordination and adaptation selection are only providing the outer boundary for operation with an initial configuration, constraints, and sub-goals. Hence, the centralized part is highly decoupled, visualized with the dotted box in Figure 1. The centralised components are supposed to run on a human controlled computer, like a ground station of autonomous unmanned aerial vehicle.

The behaviour network itself strongly supports the idea of an adaptive robot system by being

- opportunistic and trying to perform the best-suited action at any time even if the symbolic planner cannot handle the situation and does not find a suitable plan;
- light-weight in terms of computational complexity for fast responses;
- performing well in dynamic and partially observable environments under the assumption that actions taken at one point in time do not block decision paths in the future.

The manager module supports and manages the distributed execution of several behaviours on different machines within the robot. In fact, the manager is composing the behaviour network using the information provided by behaviours, like preconditions and effects. Moreover, the manager is monitoring and supervising the behaviour network by interpreting the

Table I
COMPARISON OF SYMBOLIC PLANNING FRAMEWORKS, BEHAVIOUR NETWORKS AND HYBRID PLANNING APPROACHES.
[Y=YES, N=NO, P=PARTLY, W=WORK IN PROGRESS, M=MANUALLY]

	Behaviour Network				Symbolic Planning		General			
	Included	Learning Effects	Hierarchies	Generating plan descriptions	Included	Standard Planner	Non-binary Preconditions	Parallel Behaviour Execution	ROS Integration	Multi-Robot Support
Maes 1989	Y	N	N	N	N	N	N	N	N	N
Jung 1998	Y	Y	N	N	N	N	N	N	N	M
Hertzberg 1998	Y	N	N	N	Y	ADL	Y	N	N	N
Decugis 1998	Y	Y	Y	N	N	N	N	N	N	N
Allgeuer 2013	Y	N	N	N	N	N	Y	Y	Y	N
Lee 2014	Y	N	N	N	Y	N	N	N	N	N
Cashmore 2015	N	N	N	P	Y	PDDL	Y	N	Y	P
RHBP (This work)	Y	W	W	Y	Y	PDDL	Y	Y	Y	Y

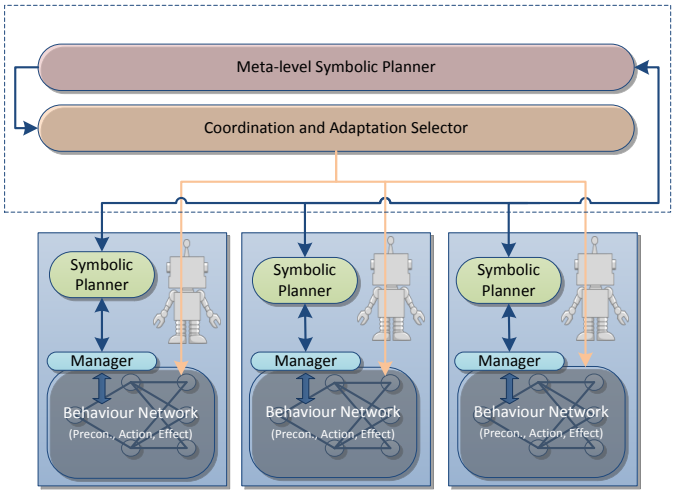

Figure 1. Target architecture for multi-robot hybrid behaviour planning

provided plan and influencing the behaviours accordingly. See next section for more details.

The first expansion stage of our concept, named *ROS Hybrid Behaviour Planner - RHBP*, implements a large subset of the former presented target architecture. At the moment we already implemented all required layers for an individual robot in the target architecture, namely *Symbolic Planner*, *Manager* and *Behaviour network*. However, multi-robot configurations are already supported as well by using a distributed behaviour network with behaviour nodes running on the corresponding robots. Due to the yet missing meta-level planner this requires a much more centralised approach than envisioned for the target architecture. Thus, the individual robots have to be connected to a central control instance running a *Symbolic Planner* and *Manager*. Another already possible configuration would be running all robots completely independently with each having a *Symbolic Planner*, *Manager* and *Behaviour Network*, but this would require to synchronise all goals and configurations manually. The following sections provide more

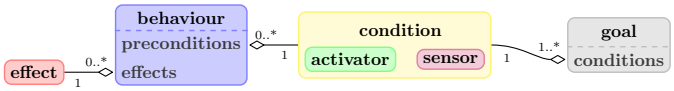

Figure 2. Behaviour network components and their relationship

details about the individual layers of the current development stage.

IV. BEHAVIOUR-NETWORK BASE

A. Model

The behaviour network layer is based on the concepts of Jung et al. [7] and Maes et al. [6], but incorporates other recent ideas from Allgeuer et al. [8], in particular supporting concurrent behaviour execution, non-binary preconditions and effects.

The main components of the network are behaviours representing tasks or actions that are able to interact with the environment by sensing and acting. *Behaviours* and *goals* both use *condition* objects composed of an *activator* and *sensors* to model their environmental runtime requirements, see Figure 2. The network of behaviours is created from the dependencies encoded in wishes based on preconditions and effects.

A *wish* of a behaviour expresses the satisfaction with the world state (current sensor values). It is related to a sensor and uses a real value [-1,1] to indicate both the strength and direction of a desired change, 0 indicates complete satisfaction. Greater values express a stronger desire, by convention negative values correspond to a decrease, positive values to an increase.

Effects model the expected influence to available sensors (the environment) of every behaviour similar to wishes.

Goals describe desired conditions of the system, their implementation is similar to behaviours except that they do not have an execution state or model effects on the network. Therefore, goals incorporate conditions that allow for the determination of their satisfaction and express wishes exactly like behaviours do. Furthermore, goals are either permanent and remain active

in the system as maintenance goals, or are achievement goals that are deactivated after fulfilment.

Sensors model the source of information for the behaviour network and buffer and provide the latest sensor measurements. The type of the sensor value is arbitrary, but to form a valid condition a matching combination of sensors and activator must be used. Raw sensor values of arbitrary type are mapped into the behaviour network by *activators*. Activators compute a utility score (precondition satisfaction) from sensor values using an internal mapping function, see next section for more details. The separation of sensor and activator fosters the reuse of code and allows also the abstract integration of algorithms using more complex activation functions like potential fields. Our implementation already comes with basic activators for expressing boolean, threshold-based and linear mapping of one-dimensional sensors. Multi-dimensional types can either be integrated by custom activators that provide a normalisation function or by splitting dimensions into multiple one-dimensional sensors.

B. Decision Making

The key characteristics and capabilities of a behaviour network are defined by the way activation is computed from sensor readings and the behaviour/goal interaction. Behaviours are selected for execution based on a utility function that determines a real number behaviour score, called activation. There are multiple sources of activation, with negative values corresponding to inhibition. If the total activation of behaviours reach the execution threshold and all preconditions are fulfilled the planner selects behaviours for execution. The behaviour network calculation is repeated in fixed frequency that can be adjusted according to the application requirements.

The positive execution threshold is coupled to the activity of the network. It is decreased after every iteration without a running behaviour by the *Activation-Threshold-Decay* parameter (by default 20%) and increased by the same amount every time a behaviour is started in order to support the desired opportunistic habits.

At every iteration all 7 activation sources (1-7) are summed to a temporary value called activation step for every behaviour. After the activation step has been computed for every behaviour it is added to the current activation of the behaviour reduced by an activation decay factor of 0.9 by default. The decay reduces the activation that had been accumulated over time if the behaviour does not fit the situation any more and prevents the activation value from becoming indefinitely large.

The fulfilment of *preconditions*, modelled as combination of sensors and an activator, is called satisfaction (real [0,1]), see Equation 1. The overall behaviour satisfaction is the product of all precondition satisfactions.

A *predecessor* (Equation 2) is a behaviour that fulfils a wish of another behaviour. If the product of the effect of behaviour A and wish of behaviour B for a particular sensor is greater than 0 then A is a predecessor of B. A behaviour gets activation from all its executable (preconditions fulfilled) predecessors for every wish. The *successor* (Equation 3) describes the reverse relationship to a predecessor. If the (pre-) conditions of behaviour or goal B are fulfilled by behaviour A then B is the successor of A.

$C =$ precondition; $B =$ behaviour; $P =$ plan; $G =$ goal
$a_{last} =$ last total network activation; $s =$ satisfaction
$a =$ activation; $w =$ wish; $e =$ effect; $\beta =$ bias parameter
$n =$ number of elements sharing a property
$i_B =$ behaviour index in plan sequence

$$a_{prec} = \prod_{C|C \in B} s \cdot \beta_{prec} \tag{1}$$

$$a_{pred} = \sum_{B|B \in pred} \frac{e \cdot w \cdot a_B \cdot \beta_{pred}}{a_{last} \cdot n_w} \tag{2}$$

$$a_{succ} = \sum_{B|B \in succ} \frac{e \cdot w \cdot a_B \cdot \beta_{succ}}{a_{last} \cdot n_e} \tag{3}$$

$$a_{goal} = \sum_{G|G \in succ} \frac{e \cdot w \cdot \beta_{goal}}{a_{last} \cdot n_w} \tag{4}$$

$$a_{conf} = \sum_{B|B \in conf} (-1 \cdot \frac{(1 - |w|) \cdot |e| \cdot a_B \cdot \beta_{conf}}{a_{last} \cdot n_{conf}}) \tag{5}$$

$$a_{confgoal} = \sum_{G|G \in conf} (-1 \cdot \frac{(1 - |w|) \cdot |e| \cdot \beta_{conf}}{a_{last} \cdot n_{conf}}) \tag{6}$$

$$a_{plan} = \begin{cases} \frac{1}{i_B} \cdot \beta_{plan} & \text{if } B \in P \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

If the product of the effect of behaviour A on one sensor and the wish of behaviour B for the same sensor is smaller than 0 then behaviour A is a *conflictor* of B, see Equation 5. Behaviour A is as well a *conflictor* if it has an effect on a wish with an indicator value of 0 of another behaviour B because it would undo a satisfied precondition of B.

The activation by goal a_{goal} is defined similar to the activation from successors except that not the indicator of wishes of successors but those of goals are considered and that another bias term goal bias β_{goal} is used instead of the successor bias term β_{succ}, see Equation 4. Inhibition by a goal is expressed with Equation 6 to model undo or counteracting of maintenance goals. The bias term influences the characteristic of the system to strive after goals.

The behaviour network is influenced by the symbolic planner with the activation term a_{plan}, see Equation 7. It depends on the position of the first concurrence of the behaviour in the symbolic plan using 1-based indices and the plan bias term β_{plan}. This direct influence is similar to operator coupling terms as presented in [14].

After behaviour execution the activation value is reset to 0. Behaviours are not expected to finish instantaneously and multiple behaviours are allowed to run concurrently, if they are not having conflicting effects.

V. Symbolic Planner Extension

As already stated in the previous section one term (Equation 7) in the activation calculation is influenced by the symbolic planner, applying the index position of the particular behaviour in the planned execution sequence. In order to allow for a quick replacement of the planner we based our interface on the widely used Planning Domain Definition Language (PDDL) in version 2.1. Hence, a majority of existing planners can be used. In particular the planner requires deterministic problem solving in a *STRIPS model*, *Negated predicates* for translating Boolean preconditions, *Numeric fluents* and *equality* for modelling non-boolean sensor values. *Conditional effects* are optionally required because they are allowed in the behaviour layer but not essential and most planning scenarios can be designed to avoid them.

For our implementation we developed a ROS Python wrapper for the Metric-FF [16] planner, a version of FF extended by numerical fluents and conditional effects. It meets all requirements and due to its heuristic nature favours fast results over optimality. In fact the wrapper is only responsible for appropriate result interpretation and execution handling.

The mapping and translation between the domain PDDL and the resulting plan is part of the *manager*. The PDDL generation on entity level is done automatically by the behaviour, activator and goal objects themselves through a defined service interface. This includes the conversion of complex data types of sensors (like locations or other multi-dimensional data) to single dimensional fluents while maintaining as much of the original meaning as possible. Moreover, the manager monitors time constraints defined in behaviours, re-plans in case of timeouts, new available behaviours or if the behaviour network execution order deviates from the proposed plan. This ensures that replanning is only executed if really necessary and keeps as much freedom as possible for the behaviour network layer for fast response and adaptation.

The manager also handles multiple existing goals of a mission by selecting appropriate goals at the right time depending on available information, e.g. if goals can not be reached currently. Since goals can have priorities the manager tries to find a plan that reaches as many high-priority goals as possible. For that, it uses an elimination algorithm that first tries to achieve the highest priority goal together with as many other goals as possible and repeats this process with lower priority goals until it is able to find a valid solution.

VI. ROS-Integration

All components of the RHBP are based on the ROS messaging architecture and are using ROS services and topics for communication. Every component of the behaviour network, like a behaviour or sensor, is automatically registered to the manager node and reports its current status accordingly. The application specific implementation is simplified through provided interfaces and base classes for all behaviour network components that are extended by the application developer and completed by filling hooks, like start and stop of a behaviour. The class constructor automatically uses registration methods and announces available components to the manager. The ROS sensor integration is inspired by Allgeuer et al. [8] and implemented using the concept of virtual sensors. This means sensors are subscribed to ROS topics and updated by the offered publish-subscribe system. Moreover, certain behaviour network components like goals (currents satisfaction) are automatically generating ROS topics that can also be used outside the actual RHBP environment.

For each registered component a proxy object is instantiated in the manager to serve as data source for the actual planning process where the activation is computed based on the relationships arising from the reported wishes and effects. Besides the status service offered by behaviours and goals there are a number of management services available to influence the execution, for instance to start, stop, activate, deactivate and prioritise.

Due to the distributed ROS architecture the whole system works even across the physical boundaries of individual robots on a distributed system. Moreover, parameters and constants can be conveniently set using ROS mechanisms even at run time, for instance by using the provided visual rqt monitoring and configuration frontend. Furthermore, RHBP comes with generic implementations that directly support simple single dimensional topic types for numbers and booleans in order to enable direct integration of existing sensors by just configuring the topic name. For more complex sensor types the user has to extend a normalisation function in a template that reduces one or more sensors to a single dimensional value. Activators for some common ROS types are provided as well and are going to be extended in future.

In order to enable the described target architecture with multiple independent RHBP instances on individual robots in a distributed multi-robot system, several RHBP can also coexist in one ROS environment using name prefixes.

VII. Experiments

The current RHBP development stage has been tested in three different experiments.

First we compared our solution against its ancestor by implementing the same artificial test scenario as Maes [6] has used for evaluating her purely reactive behaviour system. In the scenario, a robot with 2 arms should sand a board and spray paint itself. After the robot has spray painted itself it is no longer operational which is a precondition to sanding the board. There is a vise at the robots workplace to place the board in. Furthermore, a board, a sander, and a sprayer are reachable by the robots arms. Possible plans for solving the task would be to pick up the board and sander, sand the board, put down either of it, pick up the sprayer in the free hand and spray paint itself. Alternatively the robot could use the vise to avoid putting the sander or board down. It is important to note that the robot must not spray paint itself before it has sanded the board although this action would directly lead to the goal.

In order to keep our solution comparable the same behaviours have been implemented. These are: PickUpSprayer, PickUpSander, PickUpBoard, PutDownSprayer, Put-

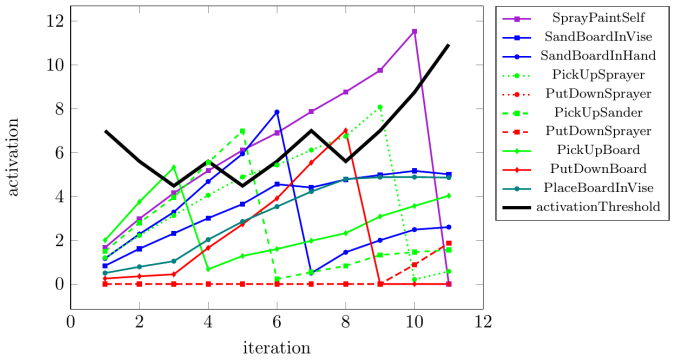

Figure 3. RHBP activation plot of the hybridly planned Maes'[6] scenario.

DownSander, PutDownBoard, PlaceBoardInVise, SandBoardInHand, SandBoardInVise, and SprayPaintSelf. All behaviours execute instantly and fulfil their action before the next planning iteration starts. However, the state representation is slightly different from Maes regarding the point that we model the used hand for picking an item. Although one particular goal of the original scenario was to show the resource management abilities of the planner, Maes' state representation makes this challenge relatively simple for the planner by having two indistinguishable hands. Thus, the system does not have to track which of the two hands can carry out the correct operation. Maes' simplification allows *a* hand to put down an object as long as it is currently located in *any* of the 2 hands. Our extended scenario implementation also applies the new features of numeric sensor values and conditional effects in order to minimize the state space in comparison to pure boolean state representations and less abstracted behaviours. To track the location of an objects its position is in encoded in an integer value: 0 means that the object is not in any hand, 1 means that it is grabbed by the left hand and -1 means that it is located in the right hand. These values determine the conditional effect that putting down an object has on the hand-occupancy sensors.

Figure 3 shows the activation of each behaviour at the end of the planning iteration before action selection is performed. Behaviours above the black activation threshold may be selected for execution if they are active, their preconditions are fulfilled and they do no not conflict with any other already running behaviour. At the end of their execution the activation is reset to 0. In this particular scenario all behaviours are finishing instantaneous so that their activation is reset before the next planning iteration starts. This is clearly visible in the plot as a sudden decrease of activation and marks the point of activation clearly. This results in a behaviour sequence of PickUpBoard, PickUpSander, SandBoardInHand, PutDownBoard, PickUpSprayer, and SprayPaintSelf.

Our hybrid implementation using the same behaviour domain model with the additional used-hand limitation outperforms Maes' with 10 instead of 19 required planning/decision making iterations. Moreover, we encountered that our solution was able to solve the problem without any parameter tuning, in contrast to Maes' implementation, of the behaviour layer due to goal-directing influence of the symbolic planning layer. For comparison our solution using only the behaviour network layer and tuned bias parameters requires 24 iterations due to the slightly more complex scenario we have targeted.

In another experiment we applied our solution on an unmanned aerial vehicle (UAV) that was developed for the national German SpaceBot Camp Competition 2015. [1]. Here, the autonomously operating UAV was required to explore an unknown GPS-denied environment while creating a map and locating three target objects. The implemented high-level behaviours have been take off, land, collision avoidance, return-to-home and exploration together with a number of conditions like keeping a certain altitude, monitoring the battery level, keeping track of the mission time and staying within the competition area. The experiments have been successfully conducted in the simulation environment MORSE [17] as well as on a real UAV running ROS on an embedded x86 computer. Even though the scenario did not include crucial points, rather, it could been implemented with simple hierarchical state machines, we have been able to prove that RHBP performs well on real systems running ROS in dynamic real-time environment. More details can be found in [18].

Furthermore, we tested the RHBP in a multi-robot scenario using the well known turtlesim simulation of the ROS tutorial package. The simulation allows to control the motion of turtle robots with differential-drive velocity commands. These commands are applied for a short amount of time allowing for a simulation in a stepwise execution. In particular we implemented a simple path finding scenario, which included random start and two target positions for each robot and a constraint of avoiding collisions with each other. For that we extended the turtlesim simulation with a simple collision detection of the nearest neighbour robot and additional visual representations (e.g. to highlight collision sensing range of turtles and target positions)[2].

In order to test the current stage of multi-robot support we implemented a centralised approach with one RHBP instance (behaviour network and symbolic planner) managing all robot behaviours. As an example of a specific realisation, the implemented and used behaviour modules for each robot are described below. All behaviours and sensors are instantiated for each robot, configured with the particular name and corresponding topics.

- *Pose-Sensor* uses the provided sensor wrapper *PassThroughTopicSensor* to subscribe to the Pose-Topic of the robot. *PassThroughTopicSensor* simply provides any sensor type to the connected activator without any type conversion, thus requiring a matching activator implementation.
- *Team-Mate-Sensor* is another instance of the *PassThroughTopicSensor* that is subscribed to the Neighbour-Pose topic, which provides the position of

[1] http://www.dlr.de/rd/desktopdefault.aspx/tabid-8101/13875_read-35268/
[2] Source code available: https://github.com/cehberlin/ros_tutorials

the closest team-mate position in a perception radius.

- *Distance-Activator* is a specialisation of RHBP's linear activator that computes activation based on the distance to a position.
- *Move-Behaviour* is a custom behaviour implementation that calculates the necessary velocity command based on the current distance and publishes it on the Twist-Topic of the turtle robot. *Move-Behaviour* is instantiated in two configurations, first as *Move-Goal-Behaviour-1* and second as *Move-Goal-Behaviour-2* leading to the two target position with effects on *Pose-Sensor*
- *Move-Team-Behaviour* is a specialisation of *Move-Behaviour* implementing a simple collision avoidance behaviour (rotating away counter-clockwise from the detected neighbour and moving with a velocity proportional to the neighbour distance) in order to keep the distance to other robots with effect on *Team-Mate-Sensor*.
- Two common acquisition goals are instantiated that use a condition formulated with the *Pose-Sensor* and the *Distance-Activator* to express the target positions.
- A maintenance goal instance applies the *Distance-Activator* and the *Team-Mate-Sensor* to express the collision avoidance.

It is important to consider that the above is one possibility of modelling the problem with RHBP. The collision avoidance could also been realised by using two single dimensional real sensors providing distance and orientation of the closest team-mate and instead of using two goals, the same could have been expressed with a different precondition configuration.

The modelled application scenario was tested on 10 randomly generated start configurations for 5 robots. The execution was monitored and the required number of planning and decision making steps have been recorded until all goals have been achieved. The two target positions stayed the same for all evaluation runs. The selected target position in the lower-left and top-right corner of the quadratic environment represent conflicting goals for the robots as they are exactly mirroring each other on the diagonal axes of the environment. The decision-making cycle of RHBP was configured to 1Hz to simplify monitoring and tracking.

Figure 4 shows scenario 7 as an example including the start configuration and the two resulting final situations using only our Behaviour Network layer and the full hybrid solution of RHBP. The given example illustrates the advantage of the hybrid planning architecture as the planner helps to generate a more efficient coordination amongst the robots and leads to a more optimal solution.

Table II summarizes the results of the comparison between the experiment executions using RHBP in behaviour network only mode and in the full hybrid planner configuration using the behaviour network together with the PDDL planner. The aggregated results show the configuration using the planner require 40 % less planning cycles to resolve the scenario, which is directly related to execution time in our simple simulation. Only start configuration 2 needed more planning

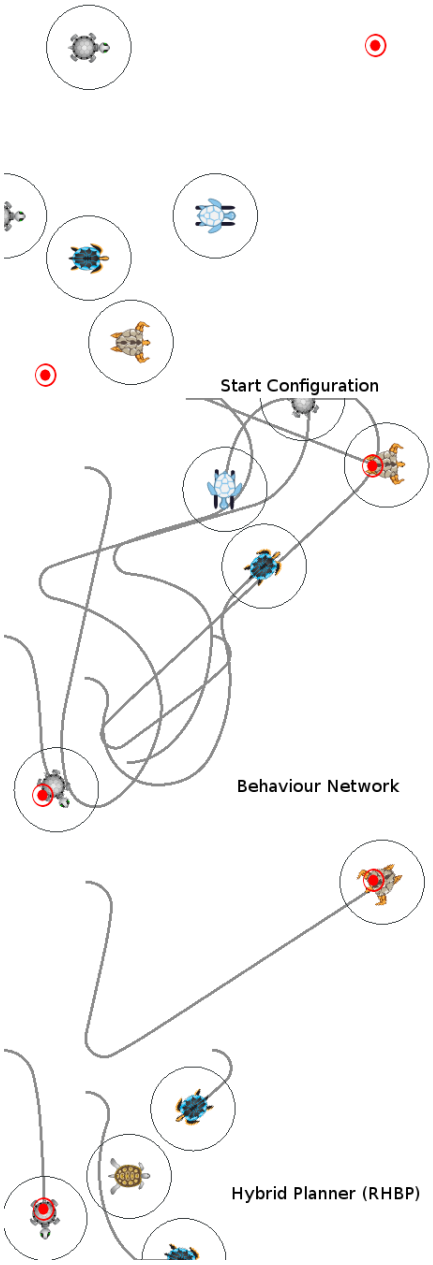

Figure 4. Turtlesim example scenario 7 with start configuration and final results comparing behaviour network only with hybrid planner (RHBP). Red points highlight the target positions. Circles around robots indicate the collision detection range. Grey lines mark the robot trajectories.

steps in the hybrid planning configuration. Here, the plan leaded to a collision avoidance situation that was difficult to resolve and couldn't been considered by the planner in advance due to the simplified model of the world.

The last experiment showed that the combination of a behaviour network with a planner leads to more efficient solutions and the integration of a planner helps to coordinate multiple robots. However, the experiments also show that the current development stage with a centralised planning of multiple robots is limited in the way that the currently used

Table II
COMPARISON OF A RHBP BEHAVIOUR NETWORK LAYER ONLY CONFIGURATION AGAINST THE HYBRID-PLANNING CONFIGURATION IN THE MULTI-ROBOT PATH FINDING SCENARIO.

	Planning steps	
Scenario	**Hybrid Planner**	**Behaviour Network**
1	24	26
2	87	48
3	54	95
4	24	39
5	37	44
6	45	70
7	34	137
8	10	56
9	35	93
10	55	71
Sum	405	679
Mean	40,5	67,9
Median	36	63
STD	20,3	31,4

planner does not support parallel behaviour execution. For that reason our planned extension has to consider parallel behaviour execution not only on the behaviour network level to further improve the results. Furthermore, the multi-robot simulation experiment proved that our solution is also capable of lower level robot control by implementing behaviours that directly steer robot motion, besides validating the current multi-robot planning and decision-making capabilities.

VIII. CONCLUSION

In this work we presented the ROS Hybrid Behaviour Planner (RHBP)[3] that addresses the needs of the ROS community for an adaptive decision making and planning package for high-level behaviour control. RHBP combines the advantages of a reactive and adaptive behaviour-based decision making component with a symbolic planner that enables appropriate goal pursuance. In particular we extended and combined existing concepts of reactive and hybrid behaviour-based decision making in a tightly ROS integrated solution. In particular our solution automatically generates the required plan descriptions (domain and problem) for the PDDL-based planner and supports multiple robots alongside all features that have only been available in single approaches, but have never been combined.

Furthermore, the current solution is embedded into a bigger concept that is going to provide extended higher-level planning and coordination mechanism management for multi-robot systems. The first development stage of RHBP is incorporating all required components of an individual robot and already has superior performance over his pure reactive behaviour network ancestor. Furthermore, the current stage is capable of controlling multi-robot systems in uncertain environments, but in a different, more centralised configuration, as we intended for the complete target solution.

The next planned step for a more decoupled and decentralised solution is the development of meta-level symbolic planning together with the ability of creating behaviour hierarchies, which will make available sub-goals for individual robots in a goal-oriented top-down fashion. Moreover, we will work on the coordination and adaptation mechanisms selector that provides mechanisms depended on higher level goals. The incorporation of reinforcement learning will further enhance the adaptation capabilities of the behaviour network by automatically learning behaviour effects and goal-supporting behaviours in a model-free fashion.

[3]Source code available: https://github.com/DAInamite/rhbp

ACKNOWLEDGMENT

This work was partially supported by the German Federal Ministry of Education and Research (BMBF grants 13N14093, project EffFeu, http://www.dai-labor.de/cog/laufende_projekte/efffeu/).

REFERENCES

[1] C.-E. Hrabia, N. Masuch, and S. Albayrak, "A metrics framework for quantifying autonomy in complex systems," in *German Conference on Multiagent System Technologies*. Springer, 2015, pp. 22–41.

[2] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," *ICRA workshop on open source software*, vol. 3, no. 3.2, p. 5, 2009.

[3] J. Bohren and S. Cousins, "The SMACH high-level executive [ros news]," *Robotics Automation Magazine, IEEE*, vol. 17, no. 4, pp. 18–20, Dec 2010.

[4] R. P. Goebel, *ROS by Example: Packages and Programs For Advanced Robot Behaviors*, ser. Pi Robot Production. Lulu.com, 2014, vol. 2.

[5] L. CogniTeam, "Cognitao (think as one)." [Online]. Available: http://www.cogniteam.com/cognitao.html

[6] P. Maes, "How to do the right thing," *Connection Science*, vol. 1, no. 3, pp. 291–323, 1989.

[7] D. Jung, "An architecture for cooperation among autonomous agents," Ph.D. dissertation, University of South Australia, 1998.

[8] P. Allgeuer and S. Behnke, "Hierarchical and state-based architectures for robot behavior planning and control," in *Proceedings of 8th Workshop on Humanoid Soccer Robots, IEEE-RAS Int. Conf. on Humanoid Robots, Atlanta, USA*, 2013.

[9] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3, pp. 189–208, 1972.

[10] T. Breuer, G. R. Giorgana Macedo, R. Hartanto, N. Hochgeschwender, D. Holz, F. Hegger, Z. Jin, C. Müller, J. Paulus, M. Reckhaus, *et al.*, "Johnny: An autonomous service robot for domestic environments," *Journal of Intelligent and Robotic Systems*, vol. 66, no. 1-2, pp. 245–272, 2012.

[11] A. L. Blum and M. L. Furst, "Fast planning through planning graph analysis," *Artificial Intelligence*, vol. 90, no. 12, pp. 281 – 300, 1997.

[12] J. Hoffmann, "Extending ff to numerical state variables," in *ECAI*, 2002, pp. 571–575.

[13] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtós, and M. Carreras, "Rosplan: Planning in the robot operating system." in *ICAPS*, 2015, pp. 333–341.

[14] J. Hertzberg, H. Jaeger, U. Zimmer, and P. Morignot, "A framework for plan execution in behavior-based robots," in *Proc. of the IEEE Int. Symp. on Intell. Control*, 1998, pp. 8–13.

[15] Y.-S. Lee and S.-B. Cho, "A hybrid system of hierarchical planning of behaviour selection networks for mobile robot control," *Int J Adv Robot Syst*, vol. 11, p. 57, 2014.

[16] J. Hoffmann, "Extending ff to numerical state variables," in *In Proceedings of the 15th European Conference on Artificial Intelligence*. Wiley, 2002, pp. 571–575.

[17] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular Open Robots Simulation Engine: MORSE," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, 2011.

[18] C.-E. Hrabia, M. Berger, A. Hessler, S. Wypler, J. Brehmer, S. Matern, and S. Albayrak, *Robot Operating System (ROS) - The Complete Reference (Volume 2)*. Springer International Publishing, 2017, ch. An autonomous companion UAV for the SpaceBot Cup competition 2015.