JAHRVIS, a Supervision System for Human-Robot Collaboration

Amandine Mayima, Aurélie Clodic, Rachid Alami

JAHRVIS, a Supervision System for Human-Robot Collaboration

Amandine Mayima[1], Aurélie Clodic[1,2] and Rachid Alami[1,2]

***Abstract*— The supervision component is the binder of a robotic architecture. Without it, there is no task, no interaction happening, it conducts the other components of the architecture towards the achievement of a goal, which means, in the context of a collaboration with a human, to bring changes in the physical environment and to update the human partner mental state. However, not so much work focus on this component in charge of the robot decision-making and control, whereas this is the robot puppeteer. Most often, either tasks are simply scripted, or the supervisor is built for a specific task. Thus, we propose JAHRVIS, a Joint Action-based Human-aware supeRVISor. It aims at being task-independent while implementing a set of key joint action and collaboration mechanisms. With this contribution, we intend to move the deployment of autonomous collaborative robots forward, accompanying this paper with our open-source code.**

I. Introduction

The supervision component is the binder of a robotic architecture. Without it, there is no task, no interaction happening, since it is in charge of controlling the other components of the architecture. Indeed, what we define as 'supervision', in the context of HRI, is the higher level of control in the architecture, the process involving real-time decision-making required for shared human-robot activity execution and monitoring. It is the robot decisional kernel. Now, for a robot, collaborating with humans is substantially different from acting alone or in coordination with other robots. Indeed, when humans are involved in a joint action [1], several socio-cognitive mechanisms come into play in order to conduct and facilitate collaborative activity between two or several humans. Some of these mechanisms are also triggered in humans' minds when they interact with robots, and they are essential for a successful collaboration. Therefore, it is important to take these into account when designing and implementing a software for a collaborative robot. Thus, we posit that in a human-robot collaborative task, the supervisor should handle coordination, communication, monitoring, and repair strategies considering joint action and collaboration mechanisms such as joint attention [2], common ground alignment [3], shared representations [4] (including shared plans [5] and joint goals [6]) and Theory of Mind (ToM) [7].

In this paper, we present the Joint Action-based Human-aware supeRVISor (JAHRVIS), a supervisor providing the control of a set of collaboration mechanisms borrowed from joint action. It is presented in full details in [8]. JAHRVIS is fully implemented and has been tested with several tasks and various contexts (see the attached video). One of them will be presented in this paper. This contribution is accompanied by the software open-source code[1].

[1]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France `firstname.surname@laas.fr`
[2]Artificial and Natural Intelligence Toulouse Institute (ANITI)

II. Related work

One can find in the literature numerous contributions proposing components with some collaborative features. We briefly mention here those which have been a source of inspiration to a certain extent. We start with the oldest one, Shary. It is a component dedicated to supervision for human-robot interactions, with a strong emphasis on communication, allowing to execute shared plans and to monitor human and robot actions [9]. Chaski is a task-level executor, focusing on coordination and decision-making. It takes as input shared plans with deadlines and tries to minimize the human idle time while controlling the plans execution [10]. There is also Pike, an online executive, that unifies intention recognition and plan adaptation to deal with temporal uncertainties during Shared Plan execution [11]. Görur et al. [12] developed a robotic system able to handle unexpected human behaviors, for instance, the human doing an action irrelevant to the task or not wanting the robot assistance. For this, they developed a human model which is used to monitor human's actions and generate reactive and proactive robot actions. Similarly, [13] proposed a reactive and proactive robotic system, being able to help when requested by the human or when detected. [14] presented a framework which generates and executes robust plans for service robots. It allows to not explicitly represent all possible situations the robot would face (*e.g.*, low battery means the robot should not navigate) and also to face unpredicted situations where an action failed with no alternative solutions. Finally, [15] implemented a supervisor allowing the robot[2] to estimate the human's beliefs not only about the state of the environment but also about the state of the goals, shared plans and actions.

It has been noticed that it is not so often that complete robotic architectures run autonomously on a real robot and interact with a human. And, when they do, the supervision system is frequently "minimal", the task being scripted, or is designed for a specific task, preventing it to be re-used in another context. Moreover, even when task and context independent contributions are proposed (*e.g.*, [14]), there is no code or documentation to re-implement their system, or they do not necessarily implement joint action or collaboration mechanisms.

[1]`https://github.com/AmandineMa/ld_rjs`

[2]All along the paper, note that we sometimes use the words robot and supervisor interchangeably. For example here, it is the supervisor which estimates and not the robot. Such an intermingling cleary shows the importance of the supervision and that it is the robot decisional-kernel.

Requir.	[9]	[10]	[11]	[12]	[13]	[14]	[15]	Ours
1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	Yes	Yes	No	Yes	No	No	Yes	Yes
3	No	Yes	Yes	Yes	Yes	No	Yes	Yes
4	Yes	No	Yes	Yes	Yes	No	Yes	Yes
5	No	No	Yes	Yes	No	Yes	No	Yes
6	Yes	Yes	No	No	No	No	Yes	Yes
7	No	No	No	No	No	Yes	No	Yes
8	Yes	Yes	Yes	Yes	No	No	No	Yes
9	No	No	No	Yes	No	Yes	Yes	Yes

TABLE I: Summary of our requirements for the related work.

III. The needs and wants of a supervision system to manage Human-Robot Interactions

A. Requirements

Some of the control features presented here are inspired by [15]. To go further and provide more flexibility, readability and genericity, we developed the Joint Action-based Human-aware supeRVISor (JAHRVIS)[3], a more complete framework of a supervision component enabling human-robot collaboration, controlling the robot to bring changes in the physical environment and to update the human partner mental states. To implement such a supervisor, we based ourselves on joint action and collaboration mechanisms [1], [16], considering among other things that a human and a robot collaborating together need to rely on a shared plan representation of the task in order to reach a joint goal. According to these mechanisms and to our needs, we defined a list of requirements we consider important for a supervisor software. We implemented JAHRVIS based on the following requirements:

1) **Be generic**. The objectives developed in the rest of this list are valid for most collaborative tasks. Thus, it is essential to develop a software not dedicated to a particular task but able to handle plans for various tasks.
2) **Take explicitly into account the human partner**. In HRI, the human and the robot are partners. Partners perform better when taking each other into account [1]. Thus, by considering human abilities, perspective and mental states to make decisions, a supervisor makes the robot a better partner for the human.
3) **Leave decisions to the human**. In some cases, it is not useful, even counterproductive that the robot plans everything beforehand. Indeed, such elements such as the human action parameters, or who should execute a given action when it does not matter, or the execution order of some actions, can be decided at execution time.
4) **Recognize human actions**. To monitor the plan progress, the robot should be able to monitor the human by recognizing their actions.
5) **Handle contingencies**. The robot has a shared plan, that is one thing, but executing it and achieving the goal is another. Indeed, it is not sure that the human has exactly the same, and errors or failures can happen. Thus, it should be able to handle a number of contingencies.
6) **Manage relevant communications**. Communication is one of the keys of collaboration [17]. Therefore, it is important to endow the robot with the ability to manage relevant communication actions, verbal and non-verbal.
7) **Consider the interaction outside collaborative tasks** A robot dedicated to collaborative tasks, in a real-life context, will interact with humans outside or between these tasks. We propose to consider this fact by defining what we called *interaction sessions*, allowing to take into account facts from one task/session to another.
8) **Adapt to the human experience, abilities or preferences**.
9) **Reproducibility, code availability of the software**.

Table I shows where we position ourselves compared to the state of the art. It aims, not to oppose systems but to provides a reading grid of existing solutions.

[3]Also almost the acronym for "Just A Rather Very Intelligent System", see https://en.wikipedia.org/wiki/J.A.R.V.I.S.

B. An architecture for a cognitive and interactive robot

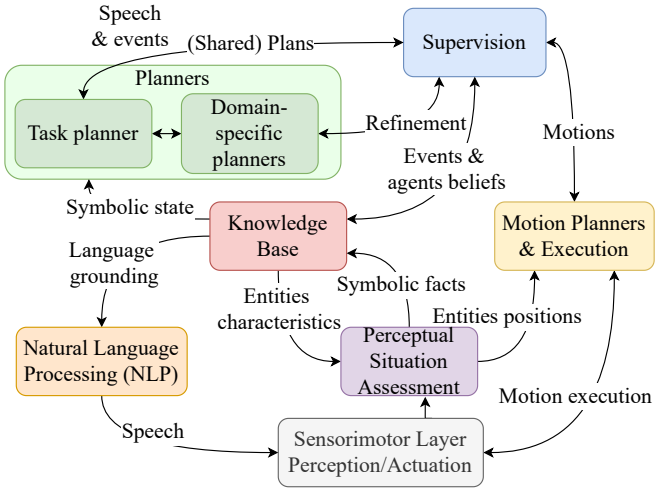


Fig. 1: Overview of the robotic architecture.

Our supervision software, focus of this paper, is part of a robotic architecture and relies on the other software of this latter to work. This architecture in which the supervision is integrated is shown in Fig. 1 and has been described in [18]. In this section, we give a quick overview of this architecture.

The **Supervision** is the puppet master of the system, embedding the robot high-level decisions, controlling its behavior and trying to react to contingencies, always considering the human it is interacting with. It relies essentially on two complementary abilities:

- the human-aware situation assessment, which based itself on data of the **Perceptual Situation Assessment** and of the **Knowledge Base**,
- the synthesis of human-aware behaviors, thanks to the human-aware **Task Planners** and **Motion planner**.

The **Situation Assessment** has two roles: 1) to gather different perceptual information and build a geometric representation of the world; the module runs reasoning processes to interpret this geometric world into a symbolic world [19];

2) estimate the human's perspective and build an estimation of their world representation.

The **Knowledge Base** makes use of the information provided by the Situation Assessment. We use Ontologenius [20], a software allowing to represent the robot knowledge and the estimation of its human partners' knowledge separately. Knowledge is stored in the form of an ontology. It represents the human-robot common ground and the current symbolic world-state.

According to the task needs and the vicinity and role of a human, several **Motion planners** might be pertinent, *e.g.*, MoveIt[4] for object manipulation, or HATEB-2 [21] for human-aware navigation.

The human-aware **Task Planners** are used to generate high-level shared plans in which each agent, human and robot, has actions assigned to them. We worked with two human-aware task planners: Hierarchical Agent-based Task Planner (HATP) [22], and Human Aware Task Planner with Emulation of Human Decisions and Actions (HATP/E-HDA) [23] which maintains conditional plans providing for the potential courses of actions linked to human decisions about their contribution to the task.

IV. Which Tool to Implement a Supervision?

In this section, we will present the programming framework we chose for our supervision software.

After a comparison considering potential compatibility with ROS, possible integration with the other software of our architecture, availability of documentation, users' feedbacks, maintenance, and possibility of code modifications, our choice went to Jason [24] which is a Java interpreter of AgentSpeak [25]. It has the advantage to be a Belief-Desire-Intention (BDI) agent-oriented framework which fits with our architecture and implements a sense-decide-act cycle allowing goal-based as well as situation-based incremental task refinement. Hence, it allows to handle contingencies and events and facilitates management of agents' – humans and robot – beliefs. We developed a bridge[5] between Jason and ROS (with different features than the ones proposed in [26]). We defined a customized class of Jason agent that we coined a ROS-Jason Agent (RJA), which has an Agent Node, an action factory and a belief base. An Agent Node has a ROS node (with the needed services, action clients and publishers) and a ROS Parameter Tree. It allows to load YAML parameters from files in which, among other things, are written services, topics and action server data. From these parameters, an Agent Node can automatically instantiate all the needed ROS communication components (*i.e.*, services, etc.) through its ROS node. An RJA can receive perception updates – from other components of the robotic architecture – in its Belief Base through ROS topic listeners. We customized the Jason belief update function as we chose to abandon a state-based perception to adopt an event-based perception (*i.e.*, percepts are updates (additions and deletions) from the external Knowledge Base). The RJA action factory – abstract in the ROS-Jason framework and instantiated in JAHRVIS – contains the list of actions it can perform.

[4]https://moveit.ros.org/

[5]https://github.com/AmandineMa/rjs

V. The Overall Structure of JAHRVIS

JAHRVIS is implemented on top of our ROS-Jason framework. We identified five high-level features for JAHRVIS and implemented their associated processes (in blue on Fig. 2), based on the objectives discusses in Section III-A. The other elements of Fig. 2 are the components of the robotic architecture mentioned in Section III-B.

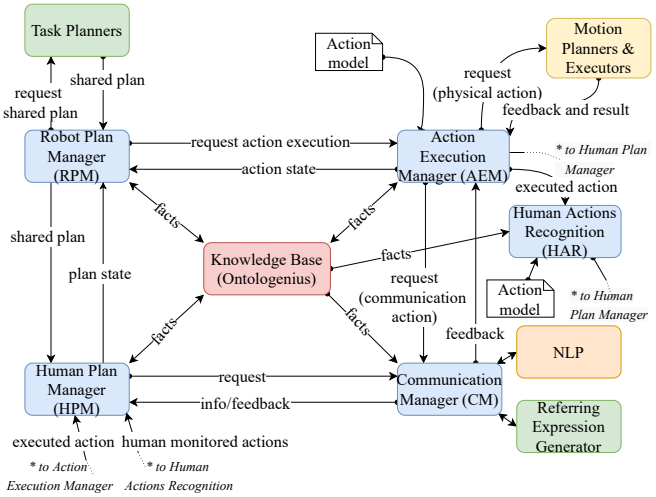


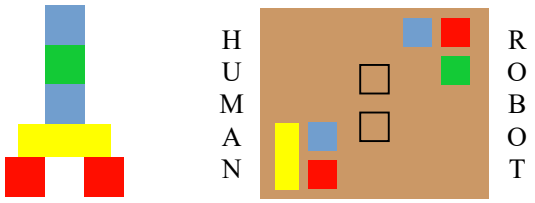
Fig. 2: The JAHRVIS processes (in blue) and the interactions between them and with the other components of the architecture (see Fig. 1).

For each process, we implemented a ROS-Jason Agent (RJA), with the desired behavior coded in AgentSpeak and the needed customizations added in Java. Thus, internal communications between the JAHRVIS processes, and so RJAs, use Jason messages. External communication with the other components of the robotic architecture is based on either ROS messages, services or action clients.

When a shared goal is established (by the Interaction Session Manager which is not described here), the shared plan is handled by the Robot Plan Manager (RPM) (see Section VI-C.1) and the Human Plan Manager (HPM) (see Section VI-C.2), *i.e.*, they allow to follow the plan progression, to make sure that the observed human actions match the ones of the plan and to decide when the robot should act. Robot actions to perform are sent to the Action Execution Manager (AEM) (see Section VI-D) that interfaces with the motion planners and controllers. As for human actions, they are recognized by the Human Actions Recognition (HAR) (see Section VI-B). Finally, the Communication Manager (CM) (see Section VI-E) is in charge of producing the communication for the human when requested by another RJA along with the human communication reception.

VI. JAHRVIS Internal Mechanisms

The examples we will give are based on a collaborative task, the *StackBuildingTask*. It has been adapted from [27]. We assume that the human and the robot have the joint goal to build a stack together as represented in Fig. 3a. At the beginning of the task, the robot and the human can access the colored cubes on their side, as illustrated in Fig. 3b.



(a) Goal (side view) (b) One initial set-up (top view)

Fig. 3: Illustration of the StackBuildingTask.

Two placements are set on the table to indicate where to put the two red cubes which are the base of the stack. Each agent can perform 3 actions: Pick, Place and Wait. The task is simple, but still, it involves both agents which will have to collaborate, and there are potential conflicts and decisions to take into account. It brings situations where the robot has to adapt to the human's needs and preferences. A video is provided with this paper, showing a PR2 robot – controlled by the presented architecture – performing this shared task with a human, coordinating and adapting to various human decisions and contingencies.

A. Knowledge Representations and Management

When involved in joint actions, humans maintain shared representations [1] of tasks, actions, goals and have a common ground [3]. Thus, it is important to equip the robot with such representations.

During an interaction, JAHRVIS processes use internal and external knowledge bases. Concerning the first type, each RJA has its own knowledge base that is called *belief base* in Jason vocabulary. It is used for knowledge which serves to JAHRVIS internal computations only. The external knowledge base is managed through Ontologenius (see Section III-B). Updates from subscription to Ontologenius facts are received through ROS topics and converted as Jason percepts to be added to the subscribing RJA belief base. Table II shows an example of the data circulating between the JAHRVIS RJAs and Ontologenius.

1) Action Representations: Action representations are used to allow the robot 1) recognizing human actions, 2) executing actions itself, and 3) communicating about its actions and the human ones. We defined three different representations according to these three uses. Action descriptions are loaded during the initialization process. This allows JAHRVIS core to be task-independent.

Information allowing JAHRVIS to communicate about actions are loaded and stored in KB Ontologenius (external storage described in Section VI-A.1.a). This latter representation is described in the next paragraph. Human actions to recognize and robot actions to execute are written in an AgentSpeak file to benefit from Jason reasoning features (internal storage described in Section VI-A.1.b).

	RJA subscriptions (ROS topics) to Ontologenius	**RJA requests (ROS services) to Ontologenius**
HPM	*isPerceiving*(H, R) *isPerceiving*(R, H) *isLookingAt*(H, R) *isLookingAt*(H, Obj)	Class of actions Existence of action effects SPARQL to object list
CM	*isPerceiving*(R, H)	Verb conjugation Class of actions and objects Labels of actions and objects
HAR	$Moves$ $ProgEffects$ $NecessEffects$	Class of objects Exist preconditions Exist *isReachable*(Obj)

TABLE II: Example of information circulating between Ontologenius and the RJAs

a) Action Representation Stored Externally: To enable **communication**, we represented actions, their verbal labels and their effects in Ontologenius. One of the advantages of using action model stored in Ontologenius is the class inheritance: *e.g.*, if it exists multiple classes representing a Place action, let's say `human_place_cube` and `robot_place_cube`, both inherit from the properties of `PlaceAction` such as the label used for verbalization.

Moreover, a class can have labels. They are used by JAHRVIS to verbalize agent actions, based on a simple grammar we defined. For example, in "{Agent} @Place {Pickable}", elements between braces are to be instantiated at execution time by JAHRVIS communication process when needed. The @ symbol indicates that the word is a verb that should be conjugated. Verb conjugations can also be found in Ontologenius. Thus, the communication manager could process it leading to "I placed the blue cube".

b) Action Representation Stored Internally: In order to be able to **recognize a human action**, we defined it as:

$$Act_H = \langle Action, PrecondL, MoveL, ProgEffectL, NecessEffectL \rangle$$

where $Action$ is a predicate in the form of a triplet $ActName(Agent, Params)$ with $ActName$ the action name, $Agent$ the class of the agent performing it (*e.g.*, Human or Worker) and $Params$ a list of action parameter classes; $PrecondL$ the list of action preconditions; $MoveL$ the list of distinctive movements that the human could do when performing the action; $ProgEffectL$ the list of effects that we coined *progression effects* which are action effects, not enough to rule the action end but allowing the plan managers to estimate that an action is under progress; and $NecessEffectL$ the list of effects that we coined *necessary effects* which are action effects existing iff the action is over.

Our action model takes the form of Jason beliefs written in an AgentSpeak file, which is an input of the Human Action Monitoring process. For example, the action Place for a human is represented as:

```
actionModel(place(Human,[Pickable,Support]),
    [isHolding(Human,Pickable)],
    [handMovingToward(Human,SupportList)],
    [not isHolding(Human,Pickable)],
    [isOnTopOf(Pickable,Support)]).
```

The choice to have two kinds of effects has been made in order to allow the Human Action Monitoring to be robust to a potentially unreliable perception. Indeed, for example in the case of a Place action, the perception of an object hold by a human can be hectic, creating multiple addition/deletion of the fact *isHolding*(human_0, blue_cube_1). But, if the robot perceives that the object has been placed on top of a support, it can assume that the action is really over.

The action representation for **robot action execution** allows to match, for a given action, its name and the motion planner and controller needed to execute it. It is possible to specify how the action parameters should be fed to the motion planner and the reaction in case of execution failure. The example given in Listing 1 shows what functions of the motion planner and controller to call and how the system should react in case of failure.

```
@place[max_attempts(2)]
+!place(Params) : planPick("armUsed", Arm) <-
  .nth(1, Params, Obj);
  headManager(Obj,environment_monitoring,urgent);
  planPlace(Obj, Arm);
  execute("place").

-!place(Params)[error_msg(Msg)] :
  not .substring(max_attempts,Msg) <-
      +error_msg(Msg);
      !place(Params).

-!place(Params)[error_msg(Msg)] :
  .substring(max_attempts,Msg) <-
      ?error_msg(Msg);
      .fail_goal(executeAction,[error_msg(Msg)]).
```

Listing 1: Example of an internal action definition for a robot action. The first Jason plan specifies what function to call to execute the Place action. The second one describes what should be done in case of failure. The last one is triggered when the first plan has been tried twice and was requested for a third time. The failure is signaled to the task level.

2) Shared Plan Representation: As mentioned in Section III-B, we represent shared plans using HTN. This formalism allows to deal with goal-based and situation-based activities at different levels of hierarchy such as task, subtasks – abstract tasks using planning vocabulary – and actions – atomic, primitive tasks – and consequently to consider different levels of granularity. For example, it may be useful to JAHRVIS to be able to request a plan for a given abstract task which failed[6]. Another advantage is that it is easy then for the robot to communicate about subtasks and not only about actions without context.

We define a shared plan as a sequence of primitive tasks having to be performed by an agent and, abstract tasks. An abstract task λ is defined as: $\lambda = \langle id_\lambda, state_\lambda, name_\lambda, \Delta_\lambda \rangle$ where id_λ is an identification number (id) proper to λ, $state_\lambda$ is the task state estimated by the robot, $name_\lambda$ is the name of the task and the decomposition id $\Delta_\lambda = id_{\lambda\prime}$ with $id_{\lambda\prime}$ the id of the abstract task $\lambda\prime$ that has been decomposed into other tasks, including λ.

And, a primitive task Π is defined as:
$\Pi = \langle id_\Pi, state_\Pi, name_\Pi, agent_\Pi, params_\Pi, preds_\Pi, \Delta_\Pi \rangle$

where id_Π is an id proper to Π, $state_\Pi$ is the task state estimation by the robot; $name_\Pi$ is the name of the task; $agent_\Pi$ is the name of the agent that should perform the task; $params_\Pi$ is the list of parameters required for the task execution, $preds_\Pi = id_{\Pi\prime}, ..., id_{\Pi\prime\prime}$ the list of ids of the tasks $\Pi\prime, ..., \Pi\prime\prime$ needing to be achieved before the task Π can start, and the decomposition id $\Delta_\Pi = id_\lambda$ with id_λ the id of the abstract task λ that has been decomposed into other tasks, including Π.

We defined nine possible values for an abstract or primitive task $state$ which are shown in Table III.

State	Description
PLANNED	needs to be done later
TODO	needs to be done now
ONGOING	is in progress
EXECUTED	is achieved
SUSPENDED	needs to be set to UNPLANNED
UNPLANNED	is not part of the plan anymore (cond. plans)
NOT_STARTING	was TODO but took too much time before start
NOT_FINISHED	was started but has not been achieved
NOT_SEEN	was achieved but not observed by the partner

TABLE III: The nine possible state values of an abstract or primitive task.

So, for example, an excerpt of the StackBuildingTask plan in which the human and the robot place the first blue cube of the stack and the green cube, generated by HATP/EHDA is:

$$\lambda_{13} = \langle 13, \text{PLANNED}, \text{h\_place\_blue\_cube}, 1 \rangle$$
$$\Pi_{139} = \langle 139, \text{PLANNED}, \text{human\_place\_cube}, \text{human\_0}, [\text{blue\_cube\_2,stick}], 138, 13 \rangle$$

B. Human Actions Recognition

In order to coordinate properly, humans monitor each other when they are in a joint action [4]. The robot needs the same kind of process to be able to assess what the human is doing and whether it fits with what it expects or not. This allows to follow the plan progress and to estimate the level of human engagement. Existing solutions exist to recognize human actions but none of them matched all our criteria which are: 1) it should be easy to add a new action that the robot can recognize, 2) the process should output the action parameters, 3) the process should give information about the action progress, *i.e.*, modeling the action start and progression when possible and not only the end, 4) it needs to be robust to a potentially unreliable perception, and 5) an available open-source code.

Thus, we implemented our own simple model-based solution with an RJA dedicated to Human Actions Recognition

[6]This is future work.

(HAR). It relies on the action model presented in Section VI-A.1.b which it loads at initiation. We chose to base our action recognition process on human movements and action effects that the robot can observe. As it needs to recognize them, it extracts the predicate types corresponding to those and subscribes to updates about these facts to Ontologenius.

Continuously, the HAR receives facts and human movements that are present in the action model, and sends to the Human Plan Manager (HPM) RJA three types of data about human actions:

- list of actions that may have started that we coined *possibly started actions*
- list of actions that may be progressing that we coined *possibly progressing actions*
- list of actions that are estimated as finished that we coined *possibly achieved actions*

Action states are updated according to the facts defined in the human action model that the HAR receives. When the state of an action changes to *possibly started*, *possibly progressing* or *possibly achieved*, the affected list is updated and sent to the HPM.

When a $Move$ corresponding to an action is added to the RJA belief base, the given action is considered as *possibly started*. Then, if a $ProgEffect$ corresponding to this same action is detected, the action state becomes *possibly progressing* whereas if it is a $NecessEffect$, it becomes *possibly achieved*. It is the same when an action is in *possibly progressing* and that a $NecessEffect$ is detected. When a $NecessEffect$ appears whereas the action is not in *possibly started* or in *possibly progressing*, the HPM checks if a human agent may be at the source of the effect (the current criteria is "was there a human near the object associated to the effect?"). If so, the action is considered as *possibly achieved*, the HPM estimating that it missed the action execution (because the robot was not looking or a perception default). It is similar for a $ProgEffect$, in this case the action state becomes *possibly progressing*.

C. Shared Plans Handling

In order to correctly perform collaborative tasks with humans, the robot needs to know how to perform them. One way is to have a planner with a domain, computing a plan at execution time based on its current knowledge about the environment and interaction. Then, the robot must be endowed with a way to manage the execution of this "recipe". As we place ourselves in the context of joint action, plans manipulated by the robot are shared plans [5] (to differentiate from the AgentSpeak plans which code all the RJA). We chose to assume that the human can trivially decide which actions they have to do in order to reach the shared goal (other assumption could be possible). So, we consider it is not necessary to verbalize it from the outset.

As shown by [15], the robot ability to handle and execute shared plan is enhanced when the robot is able to estimate, all along the task, what its human partner knows about the current state of the world and of the task. It allows the robot to be aware of false beliefs or beliefs divergence in the human's point of view. When such things happen, it can react appropriately, either by acting or communicating. We gave the robot such an ability, *via* two processes: the Robot Plan Manager (RPM) and the Human Plan Manager (HPM) (see Fig. 2). Therefore, the first one handles the robot's beliefs about the plan and the action execution while the second handles the estimation of the human's beliefs about the plan and the communication with the human.

As we designed JAHRVIS to be as generic as possible, it can manage different kinds of human-robot plans as input:

- shared plans in which each action is allocated to an agent as well as action parameters are given objects ("usual" shared plans)
- shared plans in which actions might not be allocated to an agent at planning time and parameters might refer to objects with a semantic query ("Agent X" shared plans)
- conditional plans which anticipate different possibilities for the human decision/action.

To generate these plans, we worked with the two planners HATP and HATP/EHDA mentioned in Section III-B[7].

a) "Usual" shared plans with HATP and HATP/EHDA: The first type of shared plans handled are what we could call "usual" shared plan which are fully instantiated. Each action is allocated to an agent as well as action parameters are given objects. Thus, in this kind of plan, no decision is left to JAHRVIS about who should execute the action or with which object.

b) AgentX shared plans with HATP: The second type of shared plans is an extension of the work of [27] about postponing some decisions from planning time to execution time about the actor of some actions and some parameters. We re-implemented her idea of *AgentX* in our plan managers, enabling the *choice* of the agent who should perform the action at execution time when the planner has computed that both agents could do it. This a means to specify a goal in a more abstract way. We enhanced it by having the planner returning a SPARQL query corresponding to the possible objects, allowing to have more expressiveness and genericity in our system and taking advantage of Ontologenius.

c) Conditional shared plans with HATP/EHDA: Finally, the last type of shared plans we manipulate is conditional plan, generated by Human Aware Task Planner with Emulation of Human Decisions and Actions (HATP/EHDA). It is another mean to postpone decision at execution time about an agent actor or parameters, with plans where branch junctions concern human decision. Moreover, it gives a better insight about the human's choices and decisions as they are formalized within the plan. For example, in the StackBuildingTask the human has two choices to make during the task. First, they can choose where to place their red cube or to wait for the robot to choose for the placement. The second choice happens for placing the first blue cube of the stack,

[7]JAHRVIS could be used to execute plans from other HTN planners than HATP and HATP/EHDA by adding a Java class to format abstract and primitive tasks as presented in Section VI-A.2 – HATP and HATP/EHDA have a dedicated Java class each, for action formatting, but their plans are handled with the same code in the plan managers.

either the human can place it on the stick or leave it to the robot. Thus, at planning time, the planner does not know the choice the human will make, but thanks to the conditional plan, all possible solutions are considered and it is up to JAHRVIS to "follow" the proper branch depending on the human action detected during execution.

1) Robot Plan Management: It is in charge of the plan updates, maintaining the robot knowledge about the ongoing goal, and deciding which action should be performed by the robot and when. Moreover, it handles the monitoring of the human's actions through the control of the robot head which also enables a joint attention between human and robot.

2) Human Plan Management: The Human Plan Manager (HPM) keeps track of the estimated human mental state about the ongoing shared plan, endowing the robot with Theory of Mind. The role of this process is central, as it receives the data about the recognized human actions, deduces what the human might or might not know about the plan or action executed by the robot, and requests the communication to perform to the Communication Manager (CM).

D. Action Execution Management

Not only the robot needs decision-making, but also it needs to act. Thus, JAHRVIS has a RJA called the Action Execution Manager (AEM). The AEM is composed of a generic part managing the general flow of an action execution and of a task-specific part, specifying the distinctive characteristic of given actions in a separate AgentSpeak file (see Section VI-A.1.b). Moreover, all actions of the type *physical* are realized based on ROS action clients to communicate with the Motion Planners and Executors (see Fig. III-B) which allows a fine management of the execution through feedbacks and error codes. Finally, at execution time, for a *physical* action, the AEM selects which object or place the robot should look at. *Communicate* actions are sent to the Communication Manager for execution.

E. Communication Management

The last RJA is the Communication Manager (CM). It is not dedicated to complex talks with the human but to enable the robot to perform and interpret communication during collaborative tasks. This process is based on a software for Natural Language Processing (NLP), and closely linked to a domain-specific planner called Referring Expression Generator (REG) and presented in [28]. It aims, regarding the current symbolic state of the world, at finding the minimal set of relations to communicate and allows, when interpreting a communication, to identify a given entity. For example, if the robot wants to talk about a green cube on a table but another green one is on a close shelf and a red cube is on the table, how can it do? It queries the REG which answers with a nominal group, *e.g.*, "the green cube on the table".

1) What information to communicate? How to communicate it and when?: As mentioned previously, the HPM and the AEM can query the CM to issue a communication to the human. We focused on the following communicative acts: to give information about the ongoing task (*e.g.*, "I cleaned the table") and to request the human to perform an action (*e.g.*, "Can you clean the table?").

As communication is important, it is essential to minimize the risk that it gets lost. Thus, when the CM receives a communication to perform from another RJA, it ensures that it perceives the human before issuing the communication, so it has more chance to get the human's attention.

Moreover, when the robot needs to inform the human it executed a given action or to ask them to perform one, it needs to verbalize it properly. Still in the spirit of a generic system, the algorithm that we developed is not task/action specific. Action labels (see Section VI-A.1) and verb conjugation are stored in Ontologenius which can be manually fed with new ones when needed. At execution time, each element (subject, verb and parameters) of the action label is replaced by the proper value, thanks to queries to Ontologenius and to the REG to refer to the parameters in an unambiguous way.

2) How to Understand Communications?: Having the robot able to enunciate or ask information to the human is important, but to be able to understand communication from them is as well. The human should be able to communicate about the plan such as to ask precision about a given action or to ask the robot to perform an action. We focused on the latter. When the CM receives a human sentence such as "Take the green cube", it queries the NLP which returns the action name (*e.g.*, "take"), a SPARQL query corresponding to the parameter (*e.g.*, a SPARQL matching "green cube"), and a comprehension score. Then, it requests Ontologenius for the list of objects corresponding to the SPARQL query. If the human has properly given their instruction, the object list size should be 1 and the algorithm is over (*e.g.*, if there was only one green cube). However, for some reason, they may have been imprecise or absent minded (*e.g.*, they forgot that another green cube was on the shelf). In this case, the robot asks more details to the human using the REG to refer to the objects in a non-ambiguous way: "Do you mean the green cube on the table or on the shelf?". Finally, once the CM isolated an action and a parameter, it sends them to the AEM (*e.g.*, `take(pr2_robot,green_cube)`).

VII. Conclusion

We have proposed a generic supervision software dedicated to human-robot collaboration, within a cognitive architecture, which manages the robot high-level decisions. It was used to control the robot in different task contexts: the StackBuildingTask as presented in this paper, the Director Task [18], a tabletop scenario [28], [29], and a direction-giving task [30]. Have we fulfilled the requirements listed in Section III-A?

How does it adapt to human experience, abilities or preferences? It was not the element on which we focused the most, but experience can be taken into account with the REG by referring to past events fed by JAHRVIS to Ontologenius.

How does it consider interaction outside collaborative tasks? We designed a frame, called interaction session, endowing the robot with internal states such a greeting, tasks,

goodbyes. Its management by JAHRVIS is still preliminary and has been used only in some applications [30].

How does it manage relevant communications? Dialog is not the focus of this work, but it appeared important to us to consider communicative strategies that could be pertinent to accompany collaborative task achievement and contribute to make it more fluid. We focused on the ability of the robot to communicate about actions, either actions it executes or those that the human is invited to perform.

How does it handle contingencies? This is only partially taken into account since we focused essentially on contingencies induced by human presence and not on execution failures. This will be more investigated in the future.

How does it recognize the human actions? We showed that JAHRVIS could successfully recognize human actions involving object manipulations. This feature is model-based and although being quite simple, it endows the robot with the ability to recognize a new action and is quite robust to unreliable perception.

How does it leave more latitude to the human? Two methods were integrated to JAHRVIS in order to have the robot flexible and adaptive: the use of an "AgentX" and object parameters under the form of SPARQL queries, and the handling of conditional plans.

How does it take the human partner into account? First, JAHRVIS relies on an architecture incorporating human-aware components. Then, when performing a plan with the human, it monitors them and reacts accordingly to the human's actions. Also, JAHRVIS maintains and manages an estimation of the human's knowledge about the task progress.

How is it generic? We claim that the joint action and collaboration principles we tried to implement are valid in a certain number of collaborative contexts. Indeed, in any task, it is important to estimate the beliefs of the human partner (*e.g.*, what they know of the environment or what they think of the task progress). And, communication is also key. Thus, the core of JAHRVIS does not depend on a task. Every element which is task dependent is loaded at initialization. For the rest, it is task-agnostic, the decision-making processes are task independent since the algorithms used are the same for different collaborative tasks.

The next steps will be to demonstrate the system with more scenarios and to perform a user study to evaluate how humans feel working with a robot performing with JAHRVIS. Moreover, each of JAHRVIS features can and should be refined in future work. The challenge for the current work was to exhibit an effective implementation of a fully comprehensive system.

Acknowledgement: We are very grateful to Guilhem Buisan, Guillaume Sarthou and Yannick Riou for their help. This work has been supported by the Artificial and Natural Intelligence Toulouse Institute - Institut 3iA ANITI ANR-19-PI3A-0004 and ai4hri project ANR-20-IADJ-0006.

REFERENCES

[1] N. Sebanz, H. Bekkering, and G. Knoblich, "Joint action: bodies and minds moving together," *Trends in cognitive sciences*, 2006.

[2] B. Siposova and M. Carpenter, "A new look at joint attention and common knowledge," *Cognition*, vol. 189, 2019.

[3] P. R. Cohen and H. J. Levesque, "Teamwork," *Nous*, 1991.

[4] E. Pacherie, "The Phenomenology of Joint Action: Self-Agency vs. Joint-Agency," in *Joint Attention: New Developments*. MIT Press, 2012.

[5] B. J. Grosz and S. Kraus, "Collaborative plans for complex group action," *Artificial Intelligence*, vol. 86, no. 2, 1996.

[6] M. Tomasello, M. Carpenter, J. Call, T. Behne, and H. Moll, "Understanding and sharing intentions: The origins of cultural cognition," *Behavioral and brain sciences*, vol. 28, no. 5, 2005.

[7] D. Premack and G. Woodruff, "Does the chimpanzee have a theory of mind?" *Behavioral and brain sciences*, vol. 1, no. 4, 1978.

[8] A. Mayima, "Endowing the robot with the abilities to control and evaluate its contribution to a human-robot joint action," Ph.D. dissertation, Toulouse, INSA, 2021.

[9] A. Clodic, H. Cao, S. Alili, V. Montreuil, R. Alami, and R. Chatila, "Shary: a supervision system adapted to human-robot interaction," in *Experimental robotics*. Springer, 2009.

[10] J. Shah, J. Wiken, B. Williams, and C. Breazeal, "Improved human-robot team performance using Chaski, a human-inspired plan execution system," in *ACM/IEEE HRI*, 2011.

[11] E. Karpas, S. J. Levine, P. Yu, and B. C. Williams, "Robust execution of plans for human-robot teams," in *ICAPS*. AAAI Press, 2015.

[12] O. C. Görür, B. Rosman, F. Sivrikaya, and S. Albayrak, "Social cobots: Anticipatory decision-making for collaborative robots incorporating unexpected human behaviors," in *ACM/IEEE HRI*, 2018.

[13] J. Baraglia, M. Cakmak, Y. Nagai, R. P. Rao, and M. Asada, "Efficient human-robot collaboration: When should a robot take initiative?" *Int J Rob Res*, vol. 36, no. 5-7, 2017.

[14] L. Iocchi, L. Jeanpierre, M. T. Lazaro, and A.-I. Mouaddib, "A practical framework for robust decision-theoretic planning and execution for service robots," in *ICAPS*. AAAI Press, 2016.

[15] S. Devin and R. Alami, "An implemented theory of mind to improve human-robot shared plans execution," in *ACM/IEEE HRI*, 2016.

[16] K. Belhassein, V. Fernández Castro, and A. Mayima, "A horizontal approach to communication for human-robot joint action: Towards situated and sustainable robotics," in *Culturally Sustainable Social Robotics*. IOS Press, 2020.

[17] H. H. Clark, *Using language*. Cambridge university press, 1996.

[18] G. Sarthou, A. Mayima, G. Buisan, K. Belhassein, and A. Clodic, "The Director Task: a psychology-inspired task to assess cognitive and interactive robot architectures," in *IEEE RO-MAN*, 2021.

[19] S. Lemaignan, Y. Sallami, C. Wallhridge, A. Clodic, T. Belpaeme, and R. Alami, "Underworlds: Cascading situation assessment for robots," in *IEEE/RSJ IROS*, 2018.

[20] G. Sarthou, A. Clodic, and R. Alami, "Ontologenius: A long-term semantic memory for robotic agents," in *IEEE RO-MAN*, 2019.

[21] P.-T. Singamaneni and R. Alami, "Hateb-2: Reactive planning and decision making in human-robot co-navigation," in *IEEE RO-MAN*, 2020.

[22] R. Lallement, L. De Silva, and R. Alami, "HATP: An HTN Planner for Robotics," in *ICAPS Workshop on Planning and Robotics*, 2014.

[23] G. Buisan and R. Alami, "A human-aware task planner explicitly reasoning about human and robot decision, action and reaction," in *Companion of the ACM/IEEE HRI*, 2021.

[24] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak Using Jason*. Wiley, 2007.

[25] A. S. Rao, "AgentSpeak(L): BDI agents speak out in a logical computable language," in *MAAMAW*. Springer, 1996.

[26] G. R. Silva, L. B. Becker, and J. F. Hübner, "Embedded architecture composed of cognitive agents and ROS for programming intelligent robots," in *IFAC*, vol. 53, no. 2, 2020.

[27] S. Devin, A. Clodic, and R. Alami, "About decisions during human-robot shared plan achievement: Who should act and how?" in *ICSR 2017*. Springer, 2017.

[28] G. Buisan, G. Sarthou, A. Bit-Monnot, A. Clodic, and R. Alami, "Efficient, situated and ontology based referring expression generation for human-robot collaboration," in *IEEE RO-MAN*, 2020.

[29] G. Buisan, A. Favier, A. Mayima, and R. Alami, "HATP/EHDA: A robot task planner anticipating and eliciting human decisions and actions," in *IEEE ICRA*, 2022.

[30] A. Mayima, A. Clodic, and R. Alami, "Towards Robots able to Measure in Real-time the Quality of Interaction in HRI Contexts," *Int J Soc Robot*, 2021.