E-HoA: A Distributed Layered Architecture for Context-aware Autonomous Vehicles

Jean-Michel Ilie, Ahmed-Chawki Chaouche, François Pêcheux

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2019) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 11th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 6 - 9, 2020, Warsaw, Poland

E-HoA: A Distributed Layered Architecture for Context-aware Autonomous Vehicles

Jean-Michel Ilié[a,*], Ahmed-Chawki Chaouche[b], François Pêcheux[a]

[a] *LIP6, UMR 7606 Sorbonne Université, 4 Place Jussieu, 75005 Paris, France*
[b] *MISC Laboratory, University Abdelhamid Mehri - Constantine 2, Campus Ali Mendjeli, 25000 Constantine, Algeria*

Abstract

The Embedded Higher-order Agent (E-HoA) architecture presented in this paper addresses the growing need for context-aware ambient systems, such as autonomous vehicles. This multi-process architecture, to be embedded into the control loop of these vehicles, includes a Belief-Desire-Intention agent that can consistently assist the symbolic execution of intentions. It also performs the appropriate conversion of these intentions into real physical vehicle maneuvers based on ROS. The proposed architecture offers gradually 4 levels of reactivity, from arch-reflex to the deep modification of the previously built symbolic execution plan. The presented use-case, the daily delivery of a network of pharmacy offices by an autonomous vehicle taking into account contextual (spatio-temporal) traffic features, shows the efficiency and the modularity of the architecture, as well as the scalability of the reaction levels.

Keywords: Ambient systems; Autonomous vehicle; Embedded architecture; Context-awareness; Contextual planning; Reactive behavioral strategies

1. Introduction

The design of smart applications which are able to bring assistance within an ambient intelligence system is a great challenge, that already triggered many works. This domain serves huge industrial projects such as smart cities, e.g. [5] and vehicles. In this paper, we propose a smart guidance mechanism dedicated to autonomous vehicles. The key idea is to compute an execution plan of actions which allows to achieve the various intentions of a mobile vehicle. The main issue comes from the various and numerous asynchronous events that may occur in the ambient environment, that can jeopardize the resilience of the vehicle. In fact, it is of paramount importance to develop context-awareness

* Corresponding author. Tel.: +33-678-139-432 ; fax: +33-144-277-495.
E-mail address: jean.michel.ilie@lip6.fr

mechanisms in order to detect hazards (traffic, car accident possibly involving human beings) and to develop efficient intelligent and reactive activities that prevent the vehicle mission to be aborted despite the events encountered.

Different works in the robotic domain enhance some cognitive architectures dedicated to the representation of the human mind. In particular, the symbolic architecture SOAR is built on a two layered system to capture both the human cognition processes and the operational activities like perception. Related concepts can also be modeled, such as attention and the motivations which can have an impact on the design of an intelligent system [7, 6]. These works mostly suggest a system composed of many connected processes , each of which representing a specific sub-task [13].

Various agent models have already been proposed to handle the ambient complexity, among which the Belief-Desire-Intention (BDI) approach which has the advantage of introducing smart software agents with high level reasoning capacity, mainly in terms of intentions (I), coming from agent Beliefs (B) and Desires (D) [1]. Since these native basic agents lack context awareness capacities, the authors of [14] proposed a reactive model as a supplement to the agent APL programming language in order to control the software components of the vheicle. This work is related to previous software multi-layered architectures, like 3T, ATLANTIS and LAAS [9], which all subsum the robot behavioral information with the price to handle all the event messages at the deliberative/planning layer. In this context also emerges the use of the standard ROS operating system that greatly simplifies the implementation of operational physical robotic architectures.

the HoA agent is an higher-order BDI agent which mainly provides soft assistance on smartphone application to guide the user in confined environments like an airport or a hospital. HoA agents are particularly well-suited to handle the concurrency of intentions and learn from past contextual information to provide appropriate execution plans that will be successfully achieved if applied in a new but yet similar context [4]. The presented work can be viewed as an extension of the HoA approach to help the decision making of a concrete self-driving vehicle.

In this paper, we aim at embedding a high level BDI agent like HoA in a ROS-based platform [12]. This yields a E-HoA architecture or Embedded HoA, having immediately two advantages: (1) From the ROS viewpoint, the agent becomes context-aware, it can learn from field information and can react at real-time; (2) From the HoA viewpoint, symbolic intentions concretized as a plan of actions, can be scheduled according to different multi-criteria decisions among with some behavioral preferences and concurrency aspects.

For sake of efficiency, we propose to develop E-HoA like a new distributed platform based on multi-processes and a client-server protocol allowing both synchronous and asynchronous communications. This is used to decentralize the agent decision center in several pieces, while facilitating a coherent context-awareness through subscriptions to services managing context information. Also, we aim at showing that we can benefit from this decentralization to graduate the robot reaction at different levels.

The remaining of the paper is scheduled as follows: Section 2 presents the layered E-HoA architecture, which is able to execute the robot intentions on a ROS system, by means of contextual planning and learning mechanisms. The nominal loop of the robot behavior is detailed. Section 3 identifies and details four different levels of reactions, trying to maintain much of the robot intentions. In Section 4 for efficiency purpose, we show how to deploy the E-HoA processes on a concrete distributed platform. Then, a delivery use case is presented based on a city road map to demonstrate the E-HoA interest in practise. Section 5 highlights future work and perspectives.

2. E-HoA Layered Architecture

Like many autonomous driving systems such as SOAR [10, 11], the goal of the proposed E-HoA architecture is to develop the fixed computational building blocks necessary for general intelligent agents agents that can perform a wide range of tasks (path planning, decision making, or problem solving). E-HoA is a computational implementation of a theory that combines BDI reasoning concepts and their physical concretization as a set of maneuvers for an autonomous vehicle, while at the same time considering the dynamic evolution of its surrounding ambient spatio-temporal context.

As stated by Figure 1, E-HoA architecture is composed of four layers that are vertically tightly coupled to achieve a good level of performance and accuracy. In essence, E-HoA consists in a reduced set of cooperating processes that altogether define the robot behavior by exchanging synchronous and asynchronous messages using a publish/subscribe paradigm and taking advantage of a graph-based database for planification. The lower layer instantiates two ROS nodes (Acquire and Drive) to allow E-HoA to be interfaced with all the available ROS building blocks such as sen-

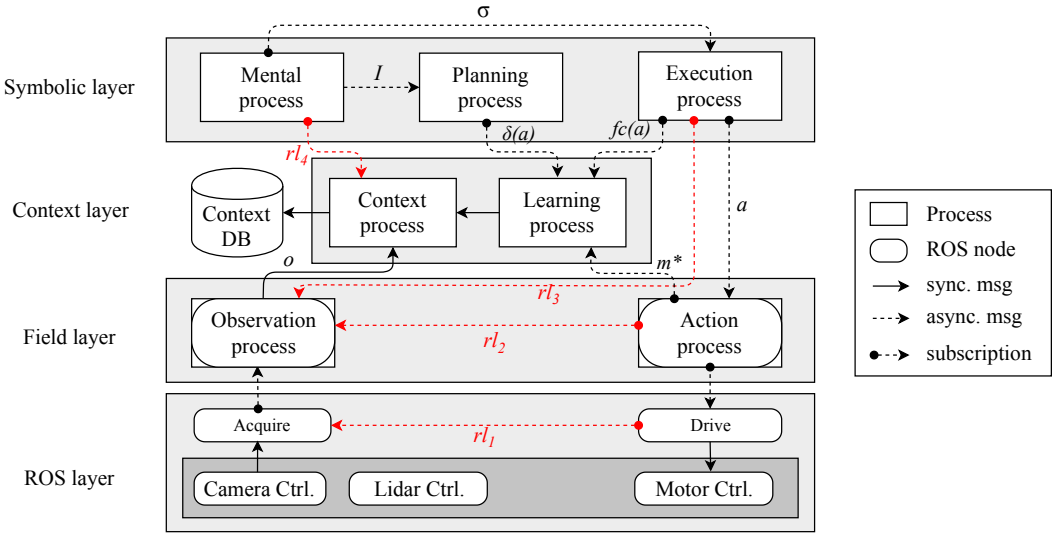

Fig. 1. Embedded Higher-order Agent architecture

sor/actuator libraries or higher level software components such as Simultaneous Localization And Mapping (SLAM) proposed by the vivid ROS community [12]. The context layer is of particular interest as it constitutes the long and short term memory needed at all the levels for learning.

The Symbolic Layer. All the high-level decisions of the E-HoA agent are taken at the symbolic layer according to its contextual information. The major process in this layer is the *Mental process* which reasons in terms of Beliefs (B), Desires (D) and Intentions (I) [1]. In the E-HoA approach, the Mental process, aiming at optimizing the achievement of the agents intentions, asks the *Planning process* on the same layer to compute an optimal execution plan (σ, list of symbolic actions), with respect to the original intentions (I) and the available contextual information data. Then, the Mental process asks to the *Execution process* to execute in order the actions defined by the execution plan.

The Field Layer. The field layer is the concrete layer of the E-HoA architecture. In practice, the *Action process* of the field layer receives symbolic actions from the Execution process and converts each symbolic action (a) into a finite set of implemented maneuvers ($m*$) controlling the robot operations. To provide context-awareness, a second process called *Observation process* is responsible for capturing the real-world physical values from the robot and its ambient environment that will be abstracted and symbolized to enrich the context layer. The Observation process mainly aims at acquiring raw or abstracted information from the different sensors and actuators.

The ROS Layer. The Action and Observation processes of the E-HoA architecture are in direct contact with their ROS nodes counterparts that manage the sensors (LIDAR, camera, IMU) and actuators (left and right motors). This layer relies on ROS and simplifies the interfacing of E-HoA with real robotic systems. In particular, it allows the seamless shift from a simulated vehicle and environment modeled with Gazebo (ROS modeling and simulation tool) to a physical robot operating in a real world.

The Context Layer. The context layer is inserted between the symbolic (Higher layer) and field (Middle layer) layers. It is composed of two main processes. The first one, namely the *Context process*, aims at storing the observed contextual information for later retrieval. Like the Observation process, it also acts as an information provider other processes can subscribe to. Three kinds of symbolic information managed in practice: The *state context* manages the state of the robot elements and also the environmental information (weather consideration, states of the road map and of the different environmental objects) ; the *execution context* yields the current state of the execution plan ; finally, the *historical context* contains information about the performances of the robot activities, in terms of intentions, plans, actions and maneuvers. The second process of this layer is the *Learning process*. This pivotal process learns about the contextual information in order to help deciding some optimization criteria [3]. For instance, it optimally computes the

best path between several locations, by managing a road map viewed as a graph and estimating the transit durations of the road map sections [8].

2.1. Inter-Processes Communication

Altogether, the three upper layers symbolic-context-field cooperate and exchange information to consolidate the behavior of the robot at all times. From a distributed system viewpoint, the E-HoA agent appears as a coordinated set of concurrent processes that communicate thanks to services according to a client/server (synchronous) approach or a publisher/subscriber (asynchronous) formalism. Messages exchanged fall into one of the three following categories:

- **Synchronous message** which provides a simple transmission scheme: a client process sends a request message and waits for an immediate reply message from the requested server process,
- **Asynchronous message** which provides a bidirectional scheme: A client process sends a request message and is notified by the server process with one immediate or delayed reply message,
- **Subscription message** which provides a publish/subscribe mechanism, thus extending the asynchronous message scheme: A client process sends a subscription request to be notified with several intermediate responses coming from the server process. Such a subscription scheme is useful when a client process needs milestone reporting about the progress of a significantly long operation such as the conversion of a symbolic action into the corresponding set of maneuvers.

2.2. The Nominal E-HoA Execution Loop

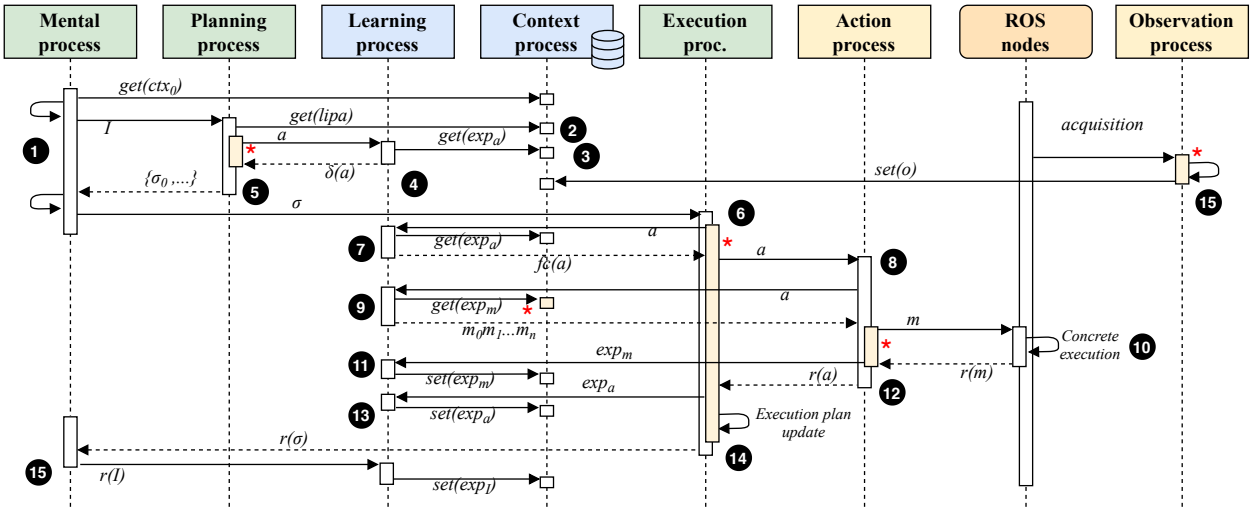

Fig. 2. The Nominal E-HoA Execution Loop

The nominal E-HoA loop addresses first the execution of an intention by means of some successives refinements up to the concrete execution of maneuvers. This principle is extended here due to the fact that a set of intentions are executed concurrently.

Figure 2 is an UML sequence diagram that details how the four layers of the E-HoA architecture intensively cooperate to define its behavior. It all begins with a starting set of intentions I acquired by the Mental process (1). The mental process has to compute an execution plan (a set of ordered symbolic actions) and is assisted in its task by the Planning process which analyses the different plans associated with the intentions according to an available spatio-temporal context. The Planning process can eventually get information from a library of action plans (2) available through the context layer. For each action a of the actions related to an intention I, the Planning process may ask the Learning process some experience data $get(exp_a)$ (3) to evaluate the duration of a (4). The Planning process may then

accumulate all the duration-weighted actions to return a list of feasible sequences of actions $\{\sigma_0, ...\}$, among with an optimal one (σ) in terms of duration $\{\delta(a)\}$ (5). Thus, Planning is a complex process as it may require the service of the Learning process to get a good estimation of the duration of each considered action a concretely (for instance, see [4]). It also should be noticed that the ROS node responsible for ambient data acquisition dynamically notifies the Observation process with environmental data that, once correctly abstracted, are used to feed the context (15).

In general and with respect to the currently available context, the Mental process selects one optimal action sequence σ and then delegates its achievement to the Execution process (6), which is responsible for the correct overall execution of the sequence. It is helped by the Learning process (7) which can determine the potential failure conditions restraining the execution of the actions ($fc(a)$). In this paper, a maximum timeout duration is computed for each action and is considered the single condition which triggers the failure of the action. The Execution process can then delegate the concrete execution of action a to the Action process (8).

For all the consolidated actions, the Action process asks the Learning process (hence also the Context process) the list of learned corresponding maneuvers (9) and then performs them in order. To actually compute the most efficient decomposition, the Action process may ask the Learning process for two kinds of services, a Contextual Shortest Path (CSP) to a given point on the region map (according to the current spatial-temporal context), or the algorithm for a specific maneuver extracted from a library of action/maneuvers. Then, the individual maneuver is executed.

Hence, each symbolic action is decomposed into a series of individual maneuvers that are propagated to the vehicle motors. The Action process is directly connected to the ROS node that actually drives the vehicle and materialize the execution (10).

Once a specific maneuver is completed, it notifies the Action process with the result of m ($r(m)$), success or failure (11). The success or failure of a specific maneuver reinforces the E-HoA experience (exp_m) and the context database is updated accordingly (11).

When the list of maneuvers corresponding to an action a has been entirely processed or in contrast when a problem is detected from some maneuver, the Action process notifies the corresponding outcome of a ($r(a)$) to the Execution process (12). As before, the success or the failure of an action a may be used to increase the E-HoA experience (exp_a). The context database is updated accordingly (13). The Execution process can then proceed to the update of the execution plan with respect to the actual duration time for action a, and thus accumulate experience on its concrete realization (14).

When an execution plan composed of a list of actions has been fully completed or in contrast when one action turns in failure, the Mental process is notified of the intentions that are achieved or failed (15). The Mental process can then deliberate implying possible changes in the considered set of the intentions, before retriggering the so-called nominal loop. It is worth noticing that the intentions that remains in activity can simply be resumed from their reached execution state, as in [2].

3. Multi-Layered and Contextual Reactive Strategies

The E-HoA layered architecture provides four means to handle external or unexpected events, each of which depending on the complexity of the appropriate handling routine to be executed. These handling routines correspond to four reactivity levels (rl_i), depicted as red connection lines in Figure 1. E-HoA is thus natively capable of adapting itself to evolution and changes of the spatio-temporal context. From a system viewpoint, unexpected events correspond to interrupts with respect to the previously described nominal execution loop. Accordingly, the four reactive levels correspond to the four levels of Interrupt Service Routines (ISR) provided by E-HoA.

Figure 3 is an UML sequence diagram that details the four reactive levels noted rl_1 to rl_4, according to the duration and latency of their management (from the simplest and quickest rl_1 that involves only the ROS layer to the most complex rl_4 that may impact the whole architecture).

The lowest reactivity level, rl_1 or arch-reflex, operates only at the ROS layer level, and represents the ability of the E-HoA agent to have vehicle reflex capabilities, i.e. the ability to react with a very small latency to immediate events that would, if not correctly and quickly handled, cause trouble to the vehicle (car crash) or the environment (person injury when the vehicle runs into a human being). The different sensors on the vehicle (LIDAR, distance sensors) and the two ROS nodes (Acquire and Drive) cooperate to constitute altogether a pre-mitigation braking system that can avoid or get around obstacles (1). From an architectural viewpoint, the ROS action node subscribes

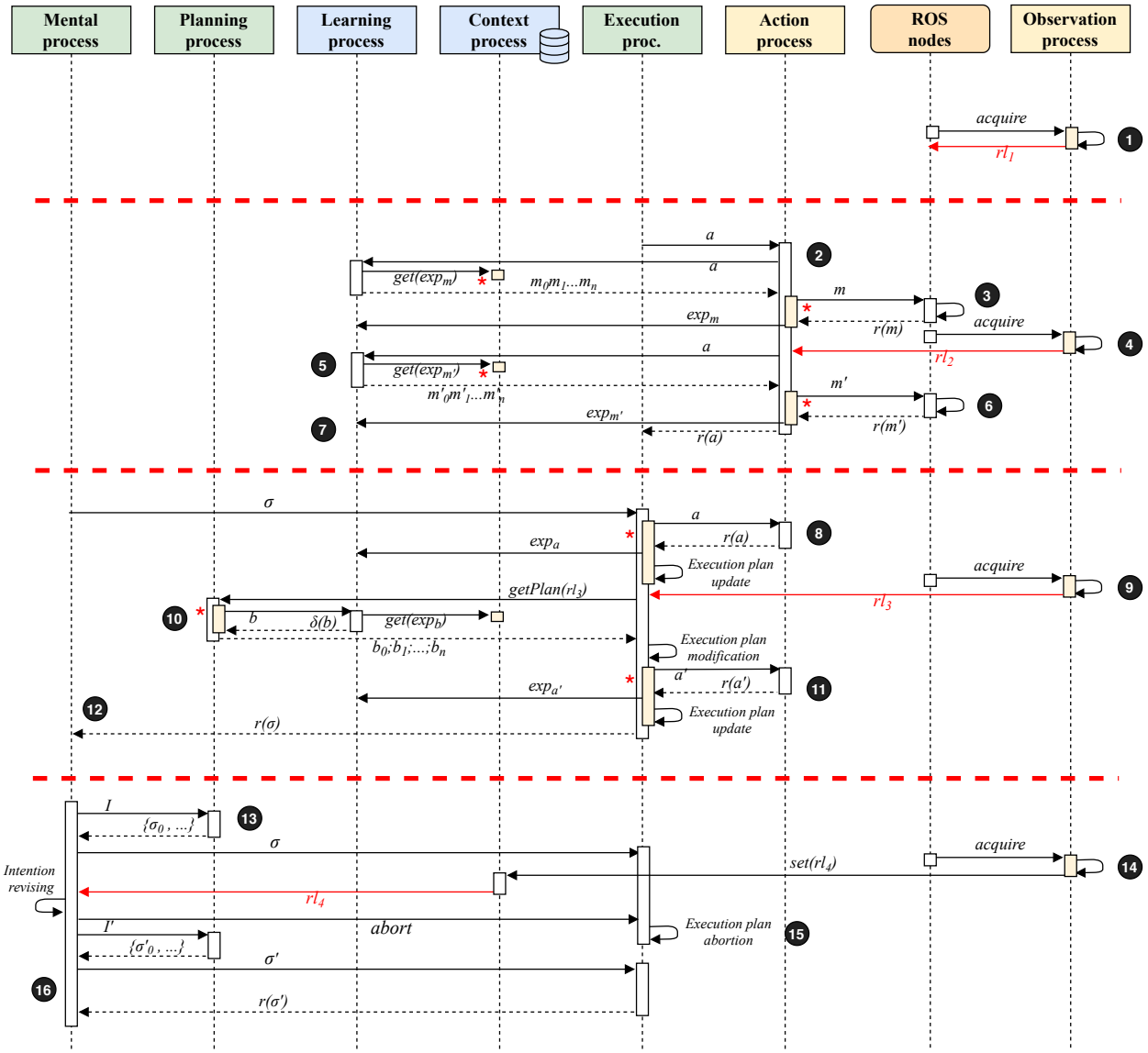

Fig. 3. The Reaction Functioning Strategies

to the observation topics serviced by the ROS Acquire node (hence the direction of the arrow in the rl_1 connection). In practice, the appropriate response is to successively stop the currently executed maneuver, execute the *get around* maneuver service routine, and resume the executed maneuver. It is worth notifying that the upper layers could not be notified with this local modification of maneuvers.

The second reactivity level, rl_2 or field-reflex, allows the E-HoA agent to correctly handle situations where a specific maneuver m can not be achieved, due to an unexpected spatio-temporal condition (a specific section of the path to be followed by the vehicle as part of its maneuvers corresponding to the current execution plan happens to be an unexpected traffic jam). Figure 3 shows how the management of such a case is dealt with by E-HoA. Consider an action issued by the Execution process and sent to the Action process, which in turn asks the Learning process to give back the correct sequence of maneuvers to be performed (2). Once the sequence is obtained, the corresponding list of maneuvers is executed in order (3). The ROS Acquire node may notify the Observation process with the event of an intractable maneuver (4), corresponding to an rl_2 reflex. In that case, the Action process has to request from

the Learning process an alternative action, with its associated maneuvers m_0 to m_n (5). The calculated sequence of maneuvers is then executed as a whole (6), provided no blocking event is detected during this re-execution (7).

The third reactivity level, rl_3 or action-reflex, is activated when a specific action, part of an execution plan can not be achieved anymore, due to the accumulated delays resulting from the execution of the previous actions in the execution plan or due to specific and urgent conditions such a *battery low* event coming from the Observation process. On the reaction diagram of Figure 3, this action-reflex occurs during the nominal execution of an action (8). When this event occurs, originating from the ROS Acquire node and Observation process, it is directly sent to the Execution process, that has to take the appropriate steps to modify the current execution plan, according to the new context (10). This involves recomputing the new execution plan σ, including the new actions b_0 to b_n (11). Once calculated, the updated execution plan is communicated to the Execution process that obtains a new chance to perform it until its successful completion (12).

The fourth reactivity level, rl_4 or intent-reflex, impacts the whole E-HoA architecture because it has a direct effect on the symbolic layer and the intentions considered by E-HoA agent. Events that can lead to the global re-evaluation of intentions can be related to global environmental conditions such as weather changes. Considering a currently set of intentions I (13), if the sensors detect a major change in a condition (snowfall in several but not all regions) that can sensibly modify the correct achievement of the intention (14), the Mental process must be notified with this global condition *snowfall forecast*, that is simultaneously saved in the context database. Once warned, the Mental process deliberates and possibly discards the intention (15) and a new planning-execution schema is generated that takes this global contextual change into account (16).

4. Use Case, Distribution and Discussion

4.1. Physical Implementation: E-HoA on a Robotic Platform

The E-HoA agent can be implemented on many robotic platforms. For the purpose of validation, we used the widely available Robotis Turtlebot3 Burger[1]. TurtleBot3 is a small, affordable, programmable, ROS-based mobile robot for use in education, research, hobby, and product prototyping. It is composed a layered infrastructure with spacers that can host the different electronic and mechanical parts, such as the continuous servo-motors with encoders for accurate ground movements, a board for controlling these motors and acquiring measures from different sensors (OpenCR for Turtlebot3), and a Raspberry PI 3 Model B[2] on which runs the ROS[3]. Connected to this board, a LIDAR continuously scans the surrounding walls and obstacles. In addition to this standard Turtlebot3 setup, we have added two complementary cameras, one connected to the Raspberry PI 3 for road tracking, and an independent IP camera for observation and detection of objects of interest. The stream captured by this wireless camera can be transmitted to an off-vehicle GPU-equipped device, an Nvidia Jetson TX2, for processing object recognition and tracing.

4.2. Software Considerations: Mapping the E-HoA Processes to Computing Resources

From a system process viewpoint, the E-HoA agent is nothing more than a reduced set of communicating processes that need to be mapped on available computing resources for efficient execution. Due to the demanding amount of computer resources needed by certain tasks (for instance, the observation process requires efficient image processing), some processes or subprocesses may be distributed to computing resources off-vehicle, such as running Convolutional Neural Network software. Also, external services provided in the cloud can be useful to adapt the robot behavior. Figure 4 highlights how the four layers of the E-HoA architecture can be efficiently distributed other a hardware:

- The Mental and Planning processes are of special importance in E-HoA, since supporting the BDI information for mobile applications. Put on the same computer, the Mental process can easily request one or possibly several occurrences of the Planning process to finally develop an execution plan solution.

[1] http://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/
[2] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[3] https://www.ros.org/

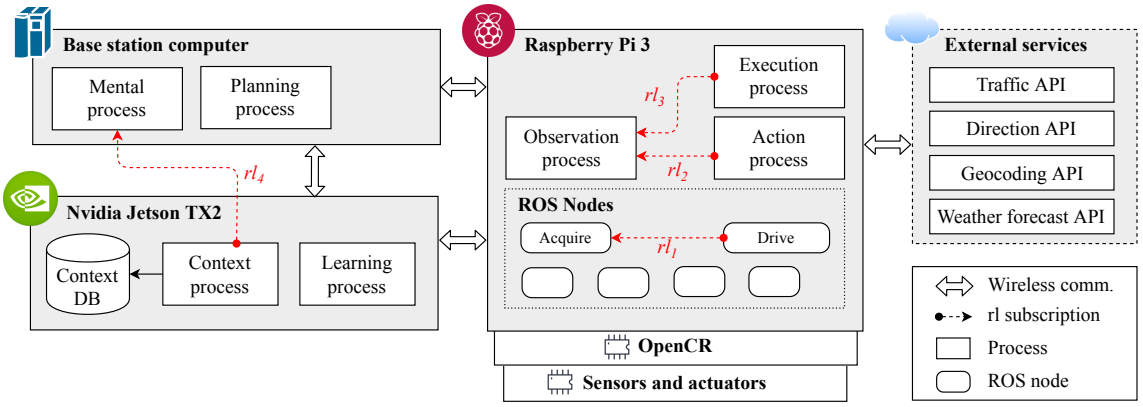

Fig. 4. A possible hardware deployment for E-HoA architecture

- The Learning process helped by the Context process relies on an efficient management of history, experience and road map data. For that purpose, we took advantage of the Neo4j graph-oriented database[4], that is particularly good at handling the consistency of the acquired spatio-temporal data. This NoSQL Database Management System (DBMS) has been selected because of its intrinsic ability to represent relevant history, experience, map and execution plan with a simple paradigm, nodes containing properties and connected by relations also having properties. This way, experiences are not only spatially specified but also temporally, thus defining a global spatio-temporal context for each experience. The Neo4j DBMS can even run on the Nvidia Jetson TX2 card with noticeable performance, that brings great flexibility in the actual mapping of E-HoA processes onto computing resources.
- The execution chain, from the Execution and Action processes to the various ROS nodes, is deployed on a Raspberry Pi 3 directly embedded on the Turtlebot3, so that all the operational activities of the guidance mechanism are concentrated on a single card. The Observation process also runs on the same card, which reveals an efficient way to feedback the environmental information in symbolic terms for playing the nominal and reactive routines.

Choosing ROS as the underlying operating system for E-HoA is natural for the sake of development simplicity and portability. Thanks to ROS-compatible tools like Gazebo[5], the E-HoA vehicle and its operating environment can globally be modeled and simulated in 3D, prior to any physical implementation. Once the Gazebo graphical simulation running the E-HoA agent operates correctly, a standard ROS methodology exists that allows to seamlessly shift from virtual simulation to a real physical vehicle operating in a real full-fledged environment. A meta-tool named E-HoA editor has specifically been designed to encapsulate Gazebo in order to automatically perform the correct-by-construction and parameterized procedural generation of scalable use cases.

4.3. Motivational Example: Pharmacy Drug Delivery with Opportunistic Situations

We consider an imaginary city with pharmacies spread over an urban region. Pharmacies handle client prescriptions and transmit the corresponding drug orders to the drug deposit when they do not have the prescribed drugs in their local stock. The drug deposit receives a list of orders/intentions coming from different pharmacies and enjoins an autonomous vehicle to deliver the prescriptions to the appropriate pharmacies, on a daily basis. The targeted vehicle is an electrical autonomous robot equipped with all the previously described features. Assuming a daily order per each pharmacy, the Mental process of the E-HoA agent for contextual execution is helped by the Planning and Learning processes to select the best road sections to schedule the deliveries in a daily tour. The Execution process is then in

[4] https://neo4j.com/
[5] http://gazebosim.org/

charge of supervising this daily tour while the Action process can control the execution of each underlying symbolic action, mainly some move operations targeting pharmacies, to be converted on maneuvers over the road map. These processes may delegate to the context process the checking of contextual constraints put on the execution of actions, from those directly solved through the neo4j request language to the more complex prolog-based logical formulas.

It may appear that the delivery truck somewhere can suffer from an event *battery low*, due to unexpected traffic jam in some sections or cross-sections of the city map. As the Action process is the single one specified to react at level rl_2, it has subscribed to the Observation process to be informed about this kind of event. Once detected, the current move is stopped by the Action process, so that to be replaced by a move to the closest garage, as precised by the Learning process. Once refueled, the Action and Execution processes can offer different ways to resume the daily tour. Thus, the Action process helped by the Learning process could positively evaluate a new series of maneuvers to reach in time the target of the move currently stopped. On the contrary as a service result, the Action process must inform of the failure the Execution process which must try in its turn to find a way to finish the remaining actions in the tour, from the garage location.

It is worth noting that the more low in the layers the event is taken into account the more efficient is the reaction. In particular, when the computation of a totally new execution plan is finally required due to the fact the execution fails to maintain the remaining current one, this should require a deliberation by the Mental process and a heavy activity of the Planning process over a set of intentions.

5. Conclusion

Dedicated to context-aware autonomous vehicles, the E-HoA scalable multi-process architecture combines BDI deliberative and reactive capabilities. With respect to existing layered architectures, all the E-HoAs embedded processes are context-centric and are supervised by a context layer which handles both concrete/symbolic information and offer estimation services based on previously learnt experiments. Thus thanks to the correct handling of four reactivity levels (from arch-reflex to intent-reflex), intentions (globally converted into an optimized sequence of atomic vehicle maneuvers) and events can be tightly and consistently intertwined as the spatio-temporal context evolves, and the computed execution plan can accordingly be updated in real-time. The layered structure of E-HoA architecture and the small number of parallel processes allows for a simple physical implementation with limited resources: A robot consisting of an SBC (Single Board Computer) running Linux, ROS and Neo4j, and several cameras for road tracking, real-time object detection and distance measurements.

References

[1] Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (Eds.), 2009. Multi-Agent Programming. Springer.
[2] Boukharrou, R., Chaouche, A.C., El Fallah Seghrouchni, A., Ilié, J.M., Saïdouni, D.E., 2015. Dealing with Temporal Failure in Ambient Systems: A Dynamic Revision of Plans. Journal of Ambient Intelligence and Humanized Computing 6, 325–336.
[3] Chaouche, A.C., El Fallah Seghrouchni, A., Ilié, J.M., Saïdouni, D., 2015. Improving the Contextual Selection of BDI Plans by Incorporating Situated Experiments, in: AIAI. Springer. volume 458 of *IFIP Advances in Information and Communication Technology*, pp. 266–281.
[4] Chaouche, A.C., El Fallah Seghrouchni, A., Ilié, J.M., Saïdouni, E.D., 2016. Learning from Situated Experiences for a Contextual Planning Guidance. Journal of Ambient Intelligence and Humanized Computing 7, 555–566.
[5] El Fallah Seghrouchni, A., Ishikawa, F., Hrault, L., Tokuda, H. (Eds.), 2016. Enablers for Smart Cities. John Wiley & Sons, Inc.
[6] Fróes, E., Gudwin, R.R., 2017. Building a Motivational Subsystem for the Cognitive Systems Toolkit, in: SCASBA, pp. 1880–1886.
[7] Gudwin, R., Paraense, A., de Paula, S.M., Fres, E., Gibaut, W., Castro, E., Figueiredo, V., Raizer, K., 2017. The Multipurpose Enhanced Cognitive Architecture (MECA). Biologically Inspired Cognitive Architectures 22, 20 – 34.
[8] Ilié, J.M., Lahiani, K., Chaouche, A.C., Pêcheux, F., 2020. An Efficient Learning Assistant for a Contextual Road Navigation. 11th Int. Conf. on Ambient Systems, Networks and Technologies (ANT 2020), Procedia Computer Science .
[9] Kortenkamp, D., Simmons, R., Brugali, D., 2016. Robotic Systems Architectures and Programming. Springer. pp. 283–306.
[10] Laird, J.E., 2012. The Soar Cognitive Architecture. The MIT Press.
[11] Lucentini, D.F., Gudwin, R.R., 2015. A Comparison among Cognitive Architectures: A Theoretical Analysis. Procedia Computer Science 71, 56–61. 6th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2015, 6-8 November Lyon, France.
[12] Martinez, A., Fernndez, E., 2013. Learning ROS for Robotics Programming. Packt Publishing.
[13] Paraense, A.L., Raizer, K., de Paula, S.M., Rohmer, E., Gudwin, R.R., 2016. The Cognitive Systems Toolkit and the CST Reference Cognitive Architecture. Biologically Inspired Cognitive Architectures 17, 32 – 48.
[14] Ziafati, P., Dastani, M., Meyer, J., van der Torre, L., 2013. Event-Processing in Autonomous Robot Programming, in: AAMAS'13, pp. 95–102.