Computing and Informatics, Vol. 42, 2023, 716–740, doi: 10.31577/cai_2023_3_716

INTEGRATION OF A CONTEXTUAL OBSERVATION SYSTEM IN A MULTI-PROCESS ARCHITECTURE FOR AUTONOMOUS VEHICLES

Ahmed-Chawki Chaouche

MISC Laboratory, University of Constantine 2 – Abdelhamid Mehri Ali Mendjeli Campus, 25000 Constantine, Algeria
e-mail: ahmed.chaouche@univ-constantine2.dz

Jean-Michel Ilié

LIP6, UMR 7606 UPMC – CNRS 4 Place Jussieu 75005 Paris, France
e-mail: jean-michel.ilie@lip6.fr

Assem Hebik

MISC Laboratory, University of Constantine 2 – Abdelhamid Mehri Ali Mendjeli Campus, 25000 Constantine, Algeria
e-mail: assem.hebik@univ-constantine2.dz

François Pêcheux

LIP6, UMR 7606 UPMC – CNRS 4 Place Jussieu 75005 Paris, France
e-mail: francois.pecheux@lip6.fr

Abstract. We propose a software layered architecture for autonomous vehicles whose efficiency is driven by pull-based acquisition of sensor data. This multi-process software architecture, to be embedded into the control loop of these vehicles,

includes a Belief-Desire-Intention agent that can consistently assist the achievement of intentions. Since driving on roads implies huge dynamic considerations, we tackle both reactivity and context awareness considerations on the execution loop of the vehicle. While the proposed architecture gradually offers 4 levels of reactivity, from arch-reflex to the deep modification of the previously built execution plan, the observation module concurrently exploits noise filtering and introduces frequency control to allow symbolic feature extraction while both fuzzy and first order logic management are used to enforce consistency and certainty over the context information properties. 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: Autonomous vehicle, multi-process architecture, context-awareness, contextual planning, reactive behavioral strategies, logical context modeling

1 INTRODUCTION

The design of autonomous vehicles is a highly active area of research. Developing a vehicle which is able to observe, plan and react safely with the surrounding environment is a major challenge for both researchers and industrialists [1].

Different works in the autonomous vehicles domain and in particular on robotics enhance some cognitive architectures dedicated to the representation of the human mind. In particular, the symbolic architecture SOAR [2] is built on a two layered system to capture both the human cognition processes and the operational activities. Related concepts can also be modeled, such as attention and the motivations which can have an impact on the design of an intelligent system [3, 4]. These works mostly suggest a system composed of many connected processes, each one representing a specific sub-task [5].

Moreover, various agent models have already been proposed to handle the ambient context. In particular, 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) [6]. Since these native basic agents lack context awareness capacities, authors of [7] proposed a reactive model as a supplement to the agent APL programming language in order to control the software components of a robot. This work is related to software multi-layered architectures, like 3T, ATLANTIS and LAAS [8], which all subsum the agent behavioral information with the price to handle all the event messages at the deliberative/planning layer. In this sense, the standard ROS operating system [9, 10] emerges greatly in order to simplify the implementation of operational physical robotic architectures.

Our main objective is to provide the autonomous vehicle with a guidance mechanism that computes an execution plan of actions while taking into account unex-

pected changes in the context. Thus, 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. Being context aware, this vehicle needs to identify its context correctly (traffic, road signs and traffic light, obstacles, current location, etc.) to make the suitable decision at the right time. The context information is acquired using sensors which may be physical (from hardware source), virtual (from software applications or services) or logical (from composition of several sources) [11] and passed as raw data inside a context aware system in order to be analyzed, filtered and symbolized.

Due to its complexity, context-awareness is the subject of many works and studies. In [12], context aware system architectures are introduced and deployed in different applications. Other studies about context modeling and reasoning techniques were presented in [11, 13, 14]. These works explore existing context aware systems, context modeling approaches and identify the major challenges and requirements for such systems.

Moreover, various techniques for data acquisition and preprocessing are presented in [15]. These studies present the different existing models for acquiring sensor data, in addition to data smoothing and noise filtering techniques. Generally, the design of a context aware system can be divided into sub-problems, consisting first of acquiring the context, then modeling and reasoning about it. In particular, the fact that sensors are relatively uncertain sources of information, requires the model to be able to handle uncertainty while being consistent and expressive. Another important aspect is the ability to represent relations between context information, which helps in the reasoning phase to deduce more information about the context and to check its consistency.

In this paper, we opt for Embedded Higher order Agent (E-HoA) architecture [16], dedicated to context-aware autonomous vehicles. This architecture embeds the high level BDI agent HoA [17] in a ROS-based platform. Actually, 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. The presented work can be viewed as an extension of the HoA approach to help the decision making of a concrete self-driving vehicle. The E-HoA architecture has two main advantages:

1. From the ROS viewpoint, the agent is context-aware, it can learn from field information and can react in real-time;
2. From the E-HoA viewpoint, reasoning on context information helps generate symbolic intentions concretized as a plan of actions that can be scheduled then performed at ROS level.

For the sake of efficiency, we aim to develop E-HoA architecture 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 vehicle reaction at different levels.

To reason on context efficiently, we propose a contextual observation system, which offers context modeling and reasoning mechanisms occurring between sensors and *E-HoA* layers. At ROS level, a specific node, called *Acquire*, is responsible for collecting context raw data from different sensors, controlling the frame rate then denoising and smoothing the raw data. The resulting data are delivered to an *Observation process* situated at the E-HoA level. Next, the processed data are passed to a helpful fuzzy logic processor in order to be symbolized, knowing that sensors can provide misleading values sometimes and that the vehicle cannot have a universal knowledge of the real context, therefore inconsistency problems are very likely to happen. To solve such problems, we use an expressive context model that takes into account the certainty of context information, and helps in the reasoning process to detect inconsistencies. Furthermore, the reasoning process allows us to infer new context information based on the specification of defined rules and relations.

The outline of the paper is as follows: Section 2 presents the multi-process E-HoA architecture, which is able to execute the vehicle intentions and actions on a ROS system, by means of contextual planning and learning mechanisms. The nominal loop of the vehicle behavior is detailed. Section 3 identifies and details four different levels of reactions, trying to maintain much of the vehicle intentions. In Section 4, we present how context observation is achieved following the life cycle of sensed data from acquisition to reasoning followed by a brief sample demonstrating the utility of our approach. In Section 5 for efficiency purposes, 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 6 discusses our approach with regard to related works. The last section concludes and outlines our perspectives.

2 E-HOA LAYERED ARCHITECTURE

Like many autonomous driving systems, the goal of the proposed E-HoA architecture is to develop the fixed computational building blocks necessary for general cognitive agents. Those agents can perform a wide range of tasks like 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 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. Concretely, E-HoA consists of a 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 plan-

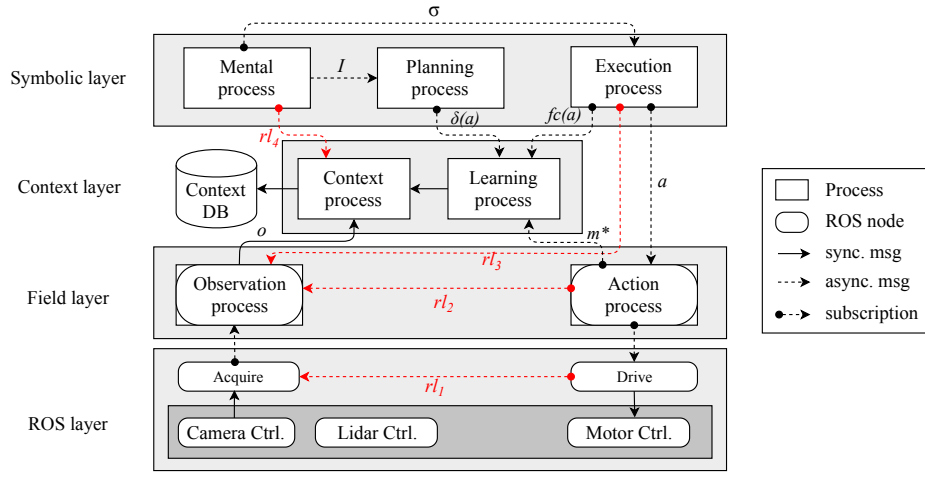


Figure 1. Embedded Higher-order Agent (E-HoA) architecture

ning. 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 sensor/actuator libraries or higher level software components such as Simultaneous Localization And Mapping (SLAM) proposed by the vivid ROS community [10]. The context layer is of particular interest as it constitutes the long and short term memory needed at all the levels of learning and reasoning.

The Symbolic Layer. All the high-level decisions of the E-HoA agent are taken at the symbolic layer according to its context information. The major process in this layer is the *Mental process* which reasons in terms of Beliefs (B), Desires (D) and Intentions (I) [6]. Aiming at optimizing the achievement of the agent's intentions, the Mental process asks the *Planning process* on the same layer to compute an optimal plan of symbolic actions (σ), with respect to the original intentions (I) and the available context information data. Then, the Mental process asks the *Execution process* to perform in order, the actions defined by the 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 (o) 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 context 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 context information in order to help decide some optimization criteria [18]. 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.

2.1 Inter-Processes Communication

Altogether, the three upper layers (symbolic, context and field) cooperate and exchange information to consolidate behavior of the robot at all times. E-HoA architecture as a distributed system is a set of concurrent processes with coordinate and 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: Client process sends a request and waits for an immediate reply message from the requested server.

Asynchronous message which provides a bidirectional scheme: Client process sends a request and is notified by the server with one immediate or delayed reply message.

Subscription message which provides a publish/subscribe mechanism, thus extending the asynchronous message scheme: 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 needs milestone re-

porting about the progress of a significantly long operation such as the conversion of a symbolic action into the corresponding set of maneuvers.

2.2 E-HoA Execution Loop

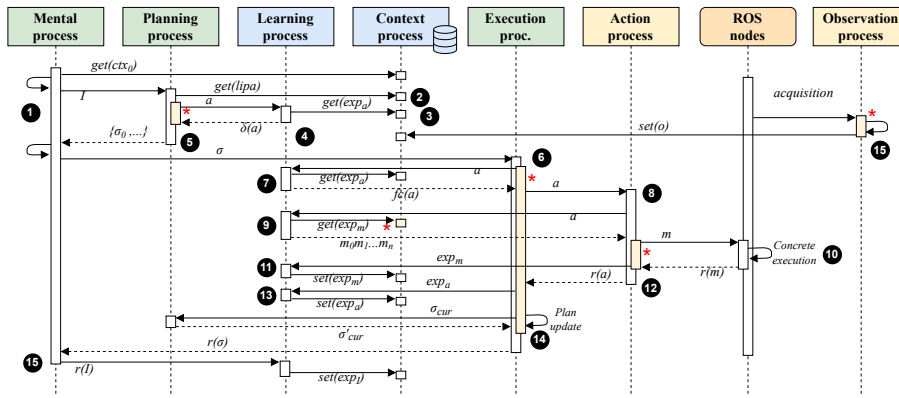


Figure 2. Nominal E-HoA execution loop

The nominal E-HoA loop addresses first the execution of an intention by means of some successive refinements up to the concrete execution of maneuvers. This principle is highlighted 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 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 $\{\sigma_0, \dots\}$, 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 [18]). It also should be noticed that the ROS node responsible for data acquisition and 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 order to control the concrete execution of actions, 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 the Contextual Shortest Path (CSP) to a given point on the map (according to the evolution of the spatial-temporal context).

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 (denoted $r(m)$), 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 [19].

3 MULTI-LAYERED AND CONTEXT 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 able 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.

Figures 3, 4, 5 and 6 are UML sequence diagrams that respectively detail 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).

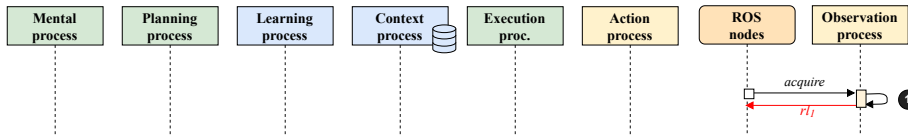


Figure 3. Functioning of arch-reflex (rl_1) strategy

The lowest reactivity level, rl_1 or arch-reflex, operates only at the ROS layer level, and represents the ability of 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 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.

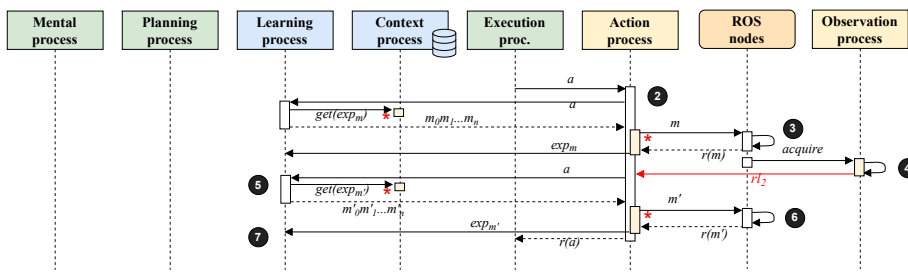


Figure 4. Functioning of field-reflex (rl_2) strategy

The second reactivity level, rl_2 or field-reflex, allows the E-HoA agent to correctly handle situations where a specific maneuver m cannot 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 4 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).

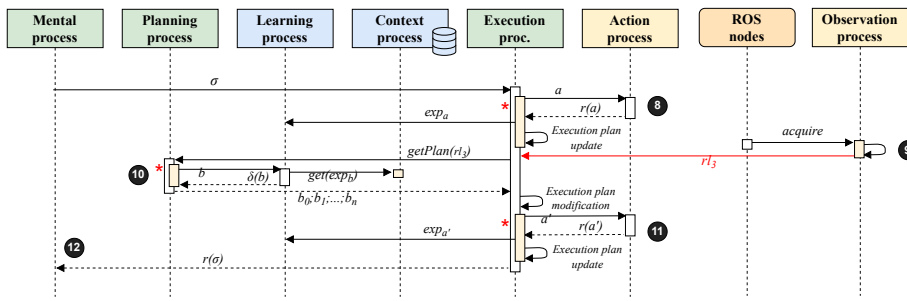


Figure 5. Functioning of action-reflex (rl_3) strategy

The third reactivity level, rl_3 or action-reflex, is activated when a specific action, part of an execution plan cannot 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 5, 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

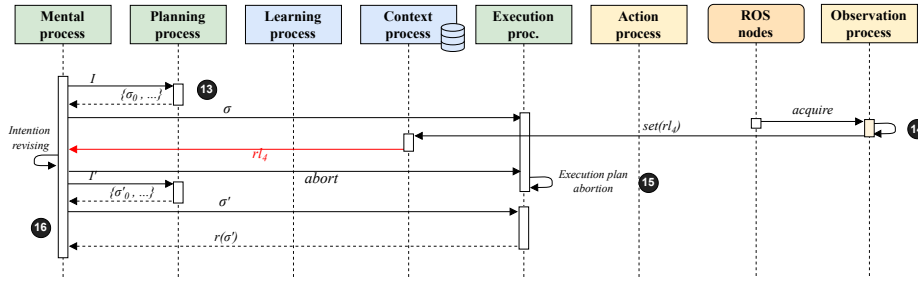


Figure 6. Functioning of intent-reflex (rl_4) strategy

intention (15) and a new planning-execution schema is generated that takes this global contextual change into account (16).

4 CONTEXTUAL OBSERVATION SYSTEM

Context-aware systems are responsible for raw data acquisition from sensors, noise reduction, and data-clearing. The acquired data passed then into features extraction. This low level data is used in the reasoning process for aggregation and composition then for validating the consistency of these acquired data.

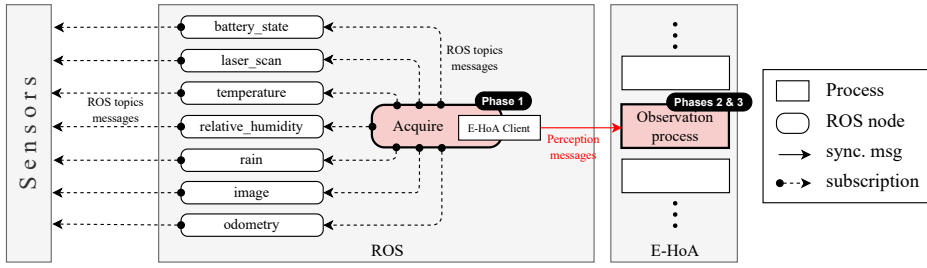


Figure 7. Observation System Architecture

The proposed architecture in Figure 7 describes a distributed context aware system which uses a blackboard model to serve context data acquired by sensors. Components of the system use neither the same protocols nor the same type of messages, however, each two components directly connected must use the same protocol to be able to send and receive messages between each other.

The proposed contextual observation system is organized in three segments: *Sensors*, *ROS* and *E-HoA*. This allows us to divide the complex tasks into smaller and simpler ones while adding more flexibility to the system. Moreover, a whole segment may be distributed or replaced without any breaking changes to the other segments.

The Sensors segment represents sensors of all types, having one basic goal which is sending raw data from sensors to the ROS segment. Sensors may use any protocol and any type of message when sending their data as long as the adapter node in the ROS segment implements the same protocol.

The ROS segment is a software part that acts like a bridge between Sensors and E-HoA layers. In particular, Acquire node is responsible for frequency control and noise-reduction of sensed raw data: It reads data from sensor adapters using ROS topics in subscriber mode; Once the data is acquired, it controls the frequency and reduces noise; Then, it analyzes processed data and outputs perception messages.
Acquire node implements an E-HoA client in order to send synchronously perception messages to the Observation process of E-HoA.

The E-HoA segment includes all the E-HoA processes, in particular Observation one. After receiving filtered context data from Acquire node, the Observation process applies low level and high level processings: First, it symbolizes the filtered context data with a fuzzy logic mechanism, then the resulting data can be processed with a first order logic technique which applies composition or aggregation to them.

In order to obtain relevant context data, we proceed in three successive stages: The acquisition of context data, its symbolization and reasoning about.

4.1 Data Acquisition Phase

In this phase, the Acquire node applies noise reduction and frequency control of sensed raw data. The fact that the sensors are not precise even if the quality of the sensor is high, a noise reduction mechanism is necessary to eliminate the inaccurate values. Noise reduction also called data clearing may be achieved using many different models. Regression models are considered among the most efficient models, these models compute the dependency from sensor value with respect to time, and then consider the regression curves as standards over which the sensor values reside. Otherwise, probabilistic models can also be used for data clearing. The expected natural range for the next sensed value is calculated based on the previous values. As in [15], the value is then excluded if it is outside the calculated field.

In our approach, we use the Savitzky-Golay smoothing filter [20] which performs a local polynomial regression of degree K on a series of sensed values to determine for each one the smoothed value. It thus preserves distribution features of values such as relative maxima, minima and width.

Let $\mathcal{D}$ be the set of all possible sensed values. Applied in a period of time Δt, we use the Savitzky-Golay filtering function $\mathit{filter} : 2^{\mathcal{D}} \times \mathbb{N} \times \mathbb{N} \to 2^{\mathcal{D}}$. The applying of $\mathit{filter}(D, \mathit{win}, \mathit{deg})$ corresponds to filtering the set D of input raw values, such that win is the number of values to consider when smoothing each value, and deg is the order of the polynomial that will be fitted to those raw data ($\mathit{deg} < \mathit{win}$).

For example, to filter the raw data from temperature sensor, we can apply $F_{temp} = filter(D_{temp}, 5, 2)$.

In addition to noise, the frequency of sensing can vary according to the physical sensor types. In the case of very high frequency sensors, the observation system can be quickly overwhelmed with a massive amount of data. Although we can decrease this frequency and save energy by configuring sensors, we can take advantage of high frequencies to increase the accuracy of information. One solution would be to collect the filtered values in a container for a specific period, then calculate only one value that represents all the other ones. We use the frequency control function $freq : 2^{\mathcal{D}} \rightarrow \mathcal{D}$. For instance, we can simply use the average function $freq(F) = \sum_{v \in F} v/|F|$, where F is the set of filtered raw data.

Indeed, this solution should not be overused at the expense of other properties such as excessive energy consumption or over-exploitation of computational resources.

Once filtered and smoothed, the sensed data is wrapped by the Acquire node in a standard message, called the *Perception message*. In addition, the Perception message contains the information of the sensor which acquired the data as well as the moment of acquisition. Thus, the Acquire node swallows different types of messages (via ROS topics) into a one standard perception message, which has the following global shape:

```
PerceptionMessage = {
   "type"   : "temperature",
   "params" : { "value" :  21, "unit" : "C", "seq" : 101,
      "timestamp" : 1594672461 }
}
```

4.2 Symbolization Phase

The goal of context symbolization is to transform raw data into atoms, called *context information*. Upon receiving a perception message from Acquire node, the Observation process performs the extraction of context information. There are many techniques that can be used to give the received raw data specific meanings. Each of these techniques has a particular goal in a particular use case as stated by [12]: Some techniques have been used for detection, classification and identification, like neural networks or Bayesian networks. Other techniques have been proposed to deal with uncertainties of data, such as logical templates, knowledge bases and fuzzy logic.

In our symbolization approach, we opt for fuzzy logic for its expressiveness and flexibility. The fuzzy logic can handle problems with imprecise and incomplete data. Further, at the symbolization level, it should not have a strict judgment on the context information, which helps the Observation process to reevaluate these judgments in the reasoning phase to be consistent with the other context information.

Definition 1 (Context information)**.** The context information is a tuple $\langle element, value, certainty \rangle$, where *element* is the context element to be described, *value* is either a value/state of this element, and $certainty \in [0, 1]$ is the extent of the credibility of the information.

The Observation process receives a sensor acquired value as an input and produces a context information as an output. Processing a sensor value without considering other sensor values around its context could lead to undesirable results. Therefore, all relevant values from other sensors are merged together to obtain more precise and relevant context information. For instance, the weather context information does not only depend on the temperature value, other factors such as relative humidity, altitude, season, and day period also matter.

In the following example, weather context information is produced from only temperature and relative humidity. The possible labels for these variables are:

- $temperature \in \{low, average, high\}$,
- $humidity \in \{low, average, high\}$,
- $weather \in \{cold, normal, hot\}$.

Many fuzzy membership functions exist, the ones which can be applied for temperature variables are: linear function for *low* and *high*, and triangular function for *average*. The set of rules for this example is:

- if temperature is *average* and humidity is *average* then weather is *normal*,
- if temperature is *low* and humidity is *high* then weather is *cold*,
- if temperature is *high* and humidity is *low* then weather is *hot*.

For example, if the temperature is 26°C and humidity 43 % then the resulting weather context information is $\langle weather, cold, 0.1 \rangle$ and $\langle weather, normal, 0.7 \rangle$ and $\langle weather, hot, 0.2 \rangle$.

As with weather context, all other types of context can be symbolized in the same way but using different variables and rules. However, object detection in image raw data is achieved using the YOLOv3 algorithm [21], as it is one of the best algorithms for applying real time detection accurately. The YOLOv3 classification process produces also context information with the same parameters, like $\langle TrafficLight, red, 0.75 \rangle$ and $\langle TrafficLight, orange, 0.25 \rangle$.

4.3 Reasoning Phase

In the reasoning phase, we first represent context information in a consistent form using first order logic predicates by using Prolog language, then we apply composition and aggregation of context information and a consistency validation. For context modeling, we need to represent information in a generic, consistent and expressive model which makes reasoning more efficient and easy to do.

In this paper, the context information are written in logic expressions by using SWI-Prolog tool [22]. In fact, SWI-Prolog is the most popular implementation of Prolog and supports a large number of features.

In this phase, we consider two types of context information: *knowledge* or facts that have certainty equaling to 1, and *assumptions* that have certainty between 0 and 1. In Prolog, the context information is expressed by the predicate $ctx(element, value, certainty)$. For the sake of clarity, the knowledge predicate $ctx(element, value, 1)$ is simply expressed as $ctx(element, value)$. For example, the assertion *"It is hot in Paris at night"*, can be expressed in SWI-Prolog as:

```
ctx(temperature, hot) * ctx(place, 'Paris') * ctx(period,
   'night').
```

In order to group many context information having the same context, we use the binary operator "*", instead of operator "and", to make the final expression unbreakable. This operator already exists in SWI-Prolog, however we simply define it in any other logical language.

Relations between context information is a critical factor in reasoning, thus a good representation of relations makes reasoning more efficient. Relations are expressed with only one predicate, like for context information.

Definition 2 (Context relation)**.** A context relation is a binary relation represented by a tuple $\langle term_1, relation, term_2, correlation\rangle$, where $term_1$ and $term_2$ are the related operands of the relation, *relation* is the name of relation and $correlation \in [0, 1]$ is the relation value for some relations that needs to be estimated.

In Prolog, the context relation is simply expressed by the predicate $rel(term_1, relation, term_2, correlation)$.

Relations are defined by their properties before they are used. We use the predicate $define(relation, property)$ to define the logical property relation *relation* with the binary relation property *property*. All the possible relation properties are supported. Using SWI-Prolog, we can trivially define the axioms for the most used properties (reflexive, transitive and antisymmetric):

```
rel(A, R, A, V) :- define(R, reflexive).
rel(A, R, B, V) :- define(R, symmetric), break(B, R, A, V).
rel(A, R, B, V) :- define(R, transitive), break(A, R, I, V),
   rel(I, R, B, V).
```

In order to avoid infinite loops in axiom definitions, we use break predicate to break it. Like for context information predicate, we simply express the relation with absolute correlation $rel(term_1, relation, term_2, 1)$ as $rel(term_1, relation, term_2)$.

4.4 An Illustrative Sample

Specifically, to apply the reasoning phase, we need to define each relation by its properties, then we specify rules and facts. Finally we query the knowledge base for results.

Rules Definition: In the following sample, we use three relations *locatedIn*, *near* and *surroundedWith*.

```
define(locatedIn, transitive)
define(near, symmetric)
```

Rule Set: We add two rules to demonstrate the efficiency of reasoning phase and the simplicity of rules definition:

Rule 1: If $PLACE_A$ is located in $PLACE_B$ and temperature is $TEMP$ in $PLACE_B$ with certainty of $CERT$, then the temperature is $TEMP$ in $PLACE_A$ but with lower certainty say like $0.9 * CERT$.

```
ctx(temperature, TEMP, NEW_CERT) * ctx(place, PLACE_A,
   1) :-
rel(PLACE_A, locatedIn, PLACE_B,1),
(ctx(temperature, TEMP, CERT) * ctx(place, PLACE_B, 1)),
NEW_CERT is (0.9 * CERT).
```

Rule 2: If $PLACE_A$ is surrounded with mountains and $PLACE_B$ is near $PLACE_A$ and temperature is normal in $PLACE_A$ with certainty of $CERT$, than the temperature is cold in $PLACE_B$ but with lower certainty say like $0.8 * CERT$.

```
ctx(temperature, cold, NEW_CERT) * ctx(place, PLACE_B,
   1) :-
rel(PLACE_A, surroundedWith, mountains, 1),
rel(PLACE_B, near, PLACE_A, _),
(ctx(temperature, normal, CERT) * ctx(place, PLACE_A,
   1)),
NEW_CERT is (0.8*CERT).
```

Facts:

```
rel('Ile-de-France', locatedIn, 'France', 1).
rel('Paris', locatedIn, 'Ile-de-France', 1).
rel('Sorbonne-Univeristy', locatedIn,'Paris',1).
rel('Versailles', near, 'Paris', 0.7).
rel('Paris', surroundedWith, mountains, 1).
ctx(temperature, normal, 0.9) * ctx(place, 'France', 1).
```

Results:

```
ctx(temperature: normal, 0.90), ctx(place:'France')
ctx(temperature: normal, 0.81), ctx(place:'Ile-de-France')
ctx(temperature: normal, 0.72), ctx(place:'Paris')
ctx(temperature: normal, 0.65),
   ctx(place:'Sorbonne-Univeristy')
ctx(temperature: cold,   0.65), ctx(place:'Versailles'}
```

Remarkably, by defining relations with axioms, the temperature in one place allows us to conclude the temperature in the places related to.

We can increase, decrease and make decisions depending on the value of certainty for context information. Moreover, using different relation properties like reflexive and transitive leads to many results about the same context information. In case many concluded assumptions give the same value with different degrees of certainty, an average function can be applied to produce a more accurate result. Otherwise, when some of these results are inconsistent assuming that the defined rules are reliable and do not produce contradictions, it is possible to identify the wrong assumptions.

5 USE CASE

5.1 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 like 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 framework[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 the wireless cameras can be transmitted to an off-vehicle GPU-equipped device, an Nvidia Jetson AGX Xavier, for processing object recognition and tracing.

[1] https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

[2] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[3] https://www.ros.org/

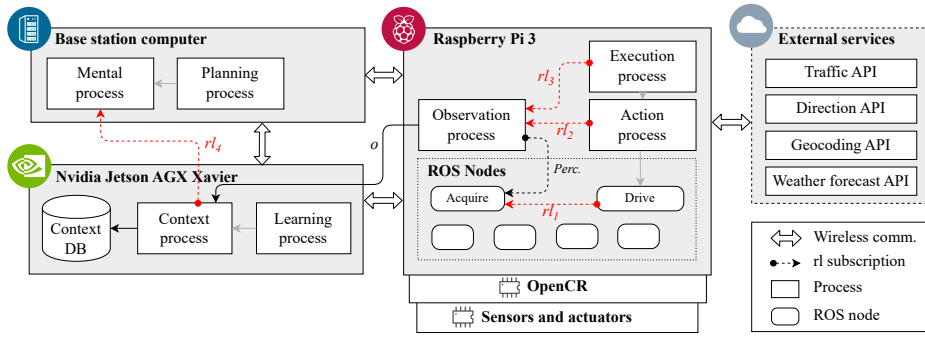


Figure 8. A possible hardware deployment for E-HoA architecture

5.2 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 8 highlights how the four layers of the E-HoA architecture can be efficiently distributed other a hardware:

- The Mental and Planning processes support 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.
- The Learning process supported 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 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 AGX Xavier card with noticeable performance that brings great flexibility in the actual mapping of E-HoA processes onto computing resources.

[4] `https://neo4j.com/`

- 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 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.

5.3 Motivational Example: Pharmacy Drug Delivery with Opportunistic Situations

We consider an imaginary city with pharmacies situated in 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 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

[5] `http://gazebosim.org/`

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 is the event 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.

6 DISCUSSION

The design of context aware systems is an active area of research and a major challenge from raw data acquisition, context modeling and reasoning. Many frameworks, toolkits and middlewares tried to overcome various challenges, like [23, 24].

For the acquisition of context information, diverse models were proposed in [11] from the direct access to sensors to complex proxy middleware. For the sake of robustness and availability, the E-HoA architecture privileges an hybridization of the context information models; The Acquire ROS node implements direct access to field values while the Context process implements both a basic synchronous service and a (asynchronous) blackboard data-centric approaches.

When developing a context-aware system, the choice of a context information model is a corner point since this has impacts on the complexity of context-aware applications, their maintainability and evolvability. Existing approaches vary from the very simple models, which support basic reasoning algorithms that could be deployed in limited use cases, to the powerful ones supporting sophisticated reasoning [13]. In [14], six types of models are mentioned: key-value models, markup scheme models, graphical models, object oriented models, logic-based models and ontological models. The authors conclude that the ontological models are the most promising for the reasoning requirements. But according to [13], an ontological model taken alone is generally unsuited for the recognition of even simple context data. Data cleaning operations and statistical machine learning methods are often required.

Other works promote logic-based modeling. The claim in [25] is that logic approaches appear to be the more expressive although requiring much effort of standardization to improve their re-usability and applicability. Context information is introduced as an abstract mathematical entity with useful additional properties adapted to the artificial intelligence topics. Mainly, an additional relation named $ist(c; p)$ is introduced to assert that the proposition p is true regarding some context c, and a recipe is formalized to perform context lifting from that.

In [26], first order logic predicates are used to describe typed context information. The authors enhance the interest of type checking, considering for instance $Location(Chris, entering, room3231)$ as a typed element, where the first argument must be a person or an object. In [27], a first order logic is used to define context information in a more generic and consistent way, introducing one predicate to model whatever $context\_element(entity, state/value, time)$.

The E-HoA software architecture also privileges an hybrid approach to model and manage the context information, as a third-part approach:

- The Acquire ROS node acts as a preprocessor to clean up input context data.
- With respect to the ontology representing the context information, some typed data and relations are specified in a based-graph database (Neo4j) which is known to be scalable, able to handle huge datasets.
- As the Context process handles the former database, it is able to perform even complex requests on context data and on their relations. In fact, there are three ways yielding values of context data:

  1. The Context process can serve any other E-HoA process subscribing to the truth of some context-based logic formula in order to react to the possible changes on-the-fly;
  2. The Context process can request some machine learning process, e.g. to compute mean traffic information according to some spatio-temporal constraints [18];
  3. And as stated in this paper, context data can also be evaluated with a degree of certainty due to the specification of context data relations.

7 CONCLUSION

Dedicated to context-aware autonomous vehicles, the E-HoA scalable multi-process architecture combines deliberative intentional concepts and reactive capabilities. From the observed events, it is used to guide the vehicle to satisfy some sets of intentions, while adapting its behavior under the dynamic environmental circumstances. The proposed functionalities can be adapted to whatever vehicles which run the known ROS system.

With respect to the existing layered architectures, all the E-HoA 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.

In order to improve the relevance of the contextual information that can be deduced from the observed data, we have investigated the way to formalize the acquisition of contextual information in three successive stages:

1. The acquisition of the raw contextual information (sensors) comes with a low level data processor exploiting noise filtering and frequency control;

2. The symbolization mechanism based on fuzzy logic, it helps certifying the context data properties;
3. The context reasoning introduces first order logic to make the higher level information emerge.

Furthermore, 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.

Although promising, we consider this work as a foundation. Concrete benchmarking approaches are required to evaluate the proposed observation mechanisms and measure its impact on the dynamic of the vehicle behavior. An immediate perspective of improvement would consist in pushing deeper machine learning techniques on data observation, both to help configuring the deduction system and to discover alternative opportunities of actions.

REFERENCES

[1] Van Brummelen, J.—O'Brien, M.—Gruyer, D.—Najjaran, H.: Autonomous Vehicle Perception: The Technology of Today and Tomorrow. Transportation Research Part C: Emerging Technologies, Vol. 89, 2018, pp. 384–406.

[2] Long, L. N.—Hanford, S. D.—Janrathitikarn, O.—Sinsley, G. L.—Miller, J. A.: A Review of Intelligent Systems Software for Autonomous Vehicles. 2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications, IEEE, 2007, pp. 69–76, doi: 10.1109/CISDA.2007.368137.

[3] Fróes, E.—Gudwin, R. R.: Building a Motivational Subsystem for the Cognitive Systems Toolkit. SCASBA, 2017, pp. 1880–1886.

[4] Gudwin, R.—Paraense, A.—de Paula, S. M.—Fróes, E.—Gibaut, W.—Castro, E.—Figueiredo, V.—Raizer, K.: The Multipurpose Enhanced Cognitive Architecture (MECA). Biologically Inspired Cognitive Architectures, Vol. 22, 2017, pp. 20–34.

[5] Paraense, A. L.—Raizer, K.—de Paula, S. M.—Rohmer, E.—Gudwin, R. R.: The Cognitive Systems Toolkit and the CST Reference Cognitive Architecture. Biologically Inspired Cognitive Architectures, Vol. 17, 2016, pp. 32–48.

[6] Bordini, R. H.—Dastani, M.—Dix, J.—El Fallah Seghrouchni, A.: Multi-Agent Programming. Springer, 2009, doi: 10.1007/978-0-387-89299-3.

[7] Ziafati, P.—Dastani, M.—Meyer, J.—van der Torre, L.: Event-Processing in Autonomous Robot Programming. AAMAS'13, 2013, pp. 95–102.

[8] Kortenkamp, D.—Simmons, R.—Brugali, D.: Robotic Systems Architectures and Programming. Springer Handbook of Robotics, Springer, 2016, pp. 283–306, doi: 10.1007/978-3-319-32552-1_12.

[9] Alzetta, F.—Giorgini, P.: Towards a Real-Time BDI Model for ROS 2. Proceedings of the 20th Workshop From Objects to Agents, Parma, Italy, June 26th–28th, 2019, 2019, pp. 1–7.

[10] Martinez, A.—Fernndez, E.: Learning ROS for Robotics Programming. Packt Publishing, 2013.

[11] Baldauf, M.—Dustdar, S.—Rosenberg, F.: A Survey on Context-Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing, Vol. 2, 2007, No. 4, pp. 263–277, doi: 10.1504/IJAHUC.2007.014070.

[12] Loke, S.: Context-Aware Pervasive Systems: Architectures for a New Breed of Applications. Auerbach Publications, 2007.

[13] Bettini, C.—Brdiczka, O.—Henricksen, K.—Indulska, J.—Nicklas, D.—Ranganathan, A.—Riboni, D.: A Survey of Context Modelling and Reasoning Techniques. Pervasive and Mobile Computing, Vol. 6, 2010, No. 2, pp. 161–180, doi: 10.1016/j.pmcj.2009.06.002.

[14] Strang, T.—Linnhoff-Popien, C.: A Context Modeling Survey. Workshop on Advanced Context Modeling, Reasoning and Management as Part of Ubicomp, 2004.

[15] Sathe, S.—Papaioannou, T. G.—Jeung, H.—Aberer, K.: A Survey of Model-Based Sensor Data Acquisition and Management. In: Aggarwal, C. C. (Ed.): Managing and Mining Sensor Data. Springer US, Boston, MA, 2013, pp. 9–50, doi: 10.1007/978-1-4614-6309-2_2.

[16] Ilié, J. M.—Chaouche, A. C.—Pêcheux, F.: E-HoA: A Distributed Layered Architecture for Context-Aware Autonomous Vehicles. Procedia Computer Science, Vol. 170, 2020, pp. 530–538, doi: 10.1016/j.procs.2020.03.121.

[17] Chaouche, A. C.—El Fallah Seghrouchni, A.—Ilié, J. M.—Saïdouni, D. E.: A Higher-Order Agent Model with Contextual Management for Ambient Systems. TCCI XVI, Springer Berlin Heidelberg, LNCS, Vol. 8780, 2014, pp. 146–169, doi: 10.1007/978-3-662-44871-7_6.

[18] Ilié, J. M.—Chaouche, A. C.—Pêcheux, F.: A Reinforcement Learning Integrating Distributed Caches for Contextual Road Navigation. International Journal of Ambient Computing and Intelligence (IJACI), Vol. 13, 2022, No. 1, pp. 1–19, doi: 10.4018/IJACI.300792.

[19] Chaouche, A. C.—Ilié, J. M.—Pêcheux, F.: Dealing with Failures for Execution Consistency in Context-Aware Systems. Vol. 177, 2020, pp. 212–219, doi: 10.1016/j.procs.2020.10.030.

[20] Schafer, R. W.: What Is a Savitzky-Golay Filter? [Lecture Notes]. IEEE Signal Processing Magazine, Vol. 28, 2011, No. 4, pp. 111–117.

[21] Redmon, J.—Farhadi, A.: YOLOv3: An Incremental Improvement. 2018, arXiv: 1804.02767.

[22] Wielemaker, J.—Schrijvers, T.—Triska, M.—Lager, T.: SWI-Prolog. 2010, arXiv: 1011.5332.

[23] Park, J.—Moon, M.—Hwang, S.—Yeom, K.: Cass: A Context-Aware Simulation System for Smart Home. 5th ACIS International Conference on Software Engineering Research, Management Applications (SERA 2007), 2007, pp. 461–467.

[24] Zeynalvand, L.—Luo, T.—Zhang, J.: COBRA: Context-Aware Bernoulli Neural Networks for Reputation Assessment. 2019, arXiv: 1912.08446.

[25] McCarthy, J.—Buvac, S.: Formalizing Context (Expanded Notes). Technical Report. Stanford University, Stanford, CA, USA, 1994.

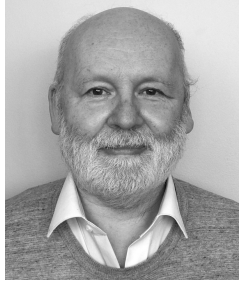
[26] Ranganathan, A.—Campbell, R. H.: An Infrastructure for Context-Awareness Based on First Order Logic. Personal and Ubiquitous Computing, Vol. 7, 2003, No. 6, pp. 353–364.

[27] Miraoui, M.—El-Etriby, S.—Abed, A. Z.—Tadj, C.: A Logic Based Context Modeling and Context-Aware Services Adaptation for a Smart Office. International Journal of Advanced Studies in Computers, Science and Engineering; Gothenburg, Vol. 5, 2016, pp. 1–6.

Ahmed-Chawki Chaouche has received his Ph.D. in computer science from both Sorbonne University (former UPMC) in France and University of Abdelhamid Mehri, Constantine 2 in Algeria (2015). Currently, he is an Associate Professor at the Constantine 2 University. He is also a Permanent Researcher in computer science and Accredited Research Director at the MISC Laboratory. His research interests include ambient intelligence systems, autonomous vehicles, implementation of IoT and connected objects, planning mechanisms and learning approaches.

Jean-Michel Ilié obtained several degrees in electronics and informatics among with the Ph.D. thesis from the University Pierre and Marie Curie in France (1990). Currently, a member of the Paris City University in its conference master higher grade (2009), he is also a Permanent Researcher of the LIP6 laboratory at the Sorbonne University. The fields of his research concern the formal validation of complex embedded distributed systems and the emergence of adapted behaviours when coping with dynamic contexts. In the last 15 years, he has tackled the way to define intelligent software agents in complex ambient systems for autonomous activity. His research keywords include spatio-temporal planning, autonomous guidance, intelligent transportation.

Assem Hebik received his Master's degree in science and technologies of information and communication (2020) from the University of Constantine 2 with excellence. Previously he had obtained a License degree with the first class honors from the same university. Currently, he has been Top rated plus freelancer on Upwork for more than three years during which he had the chance to work with international software development teams across the globe. His primary focus is scientific research, that is why he volunteers with research teams when he gets a chance.



François Pêcheux is Full Professor at the Sorbonne Université, Paris, France. He is currently heading the Polytech Sorbonne Engineering School. His research activities focus on the modelling and simulation of digital-centric heterogeneous systems. He participated in the development of numerous CAD tools for electronic design automation, especially event-driven and analogue simulators. He published more than 80 journal and conference papers in this domain.