




# Pooling of Heterogeneous Computing Resources: A Novel Approach based on Multi-Edge-Agent Concept

Florent Carlier<sup>1</sup>, Virginie Fresse<sup>2</sup>, Jean-Paul Jamont<sup>3</sup>, Loic Pallardy<sup>4</sup> and Arnaud Rosay<sup>4</sup>

<sup>1</sup>Centre de Recherche en Éducation de Nantes and Le Mans University, Le Mans, France

<sup>2</sup>Laboratory Hubert Curien UMR CNRS 5516 and University of Jean Monnet, Saint-Étienne, France

<sup>3</sup>Laboratory LCIS, Univ. Grenoble Alpes, Grenoble INP, Valence, France

<sup>4</sup>STMicroelectronics, Le Mans, France


**Keywords:** Architectures for Vehicular, Multi-Agent Systems, Edge Computing Approach, IoT-a, Avatar.


**Abstract:** Advanced driving assistance systems are major innovations in vehicles requiring more and more electronic systems both inside and outside cars and trucks and using vehicle to vehicle and vehicle to infrastructure communications. Electric/Electronic architecture of modern vehicle is based on clustering of Electronic Control Units (ECU) either by domain or by physical location. Innovation in the field of transportation is often related to the introduction of new software requiring the addition of hardware. This is a barrier to disseminate innovation in existing vehicles. The most appropriate solution to overcome this problem consists in fully exploiting under-utilized computing resources rather than adding new ones. In this paper, we propose a novel approach to manage a pool of resources by introducing the concept of EdgeAgent model. Pooling of resources is managed by three types of EdgeAgents: Mediator, Allocator and Processor. The resulting system architecture is based on IoT-a (agents as close as possible to hardware) and Avatar (virtualizing the representation of the hardware in high level: Cloud and Edge computing). The result of this work enables extension of vehicle management and functionalities while considering the environment for vehicles of the future.


## 1 INTRODUCTION


More and more intelligence is integrated inside vehicles to enhance driving safety and increase engine performance efficiency. The objective of car designers and manufacturers is not only to propose autonomous driving vehicles but also to add new features, increasing the level of automation from the level zero, where humans do the driving, up to level five through driver assistance technologies up to fully autonomous cars. Adding more and more features leads to raise the electronic computing power. The basic mundane solution is to increase the number of electronics components to execute these applications. As an example, Renesas sells hardware and software platforms to cover the full product range from the premium class to the entry level. These platforms con-

tain Systems On Chip (SoC), an integrated circuit integrating several components on one single substrate (SA, 2019). This was a viable solution as SoCs consume much less power and take up much less area than multi-chip designs. Adding SoCs for new features was possible in vehicular electronics in the past but cannot be considered in this way anymore. The use of lightweight materials within road vehicles has been considered for many years. Challenges in car manufacturing are to lightweight the cables, reducing copper wiring looms by grouping multiple legacy buses on a single Ethernet backbone, and now to optimize electronics to reduce energy consumption and weight of vehicle. Although the number of electronic features is set to grow significantly, the actual value of components should not significantly change. With the high number of microprocessors and micro-controller in vehicles, distributing the workloads is the new challenge. Such challenge requires to propose new approaches to manage the vehicle electronics in the present and in the future. Similar approaches have been already proposed in literature and in industry

<sup>a</sup> <https://orcid.org/0000-0003-0314-3667>

<sup>b</sup> <https://orcid.org/0000-0002-9944-0174>

<sup>c</sup> <https://orcid.org/0000-0002-0268-8182>

<sup>d</sup> <https://orcid.org/0000-0001-5937-5331>

for other application contexts. The emergence of the Cloud Computing, Internet of Things and Edge computing approaches aim at virtualizing, sharing and deploying resources to optimize the workloads and resources usage. The aim of this paper is to propose a similar approach in vehicle electronics, by predicting the evolution of vehicle architectures and power requirements from nowadays and future solutions.

The paper is organized as follow. The futuristic motivating scenario and the predicted vehicle electronics are presented in section two to extract future challenges. In a third section, we propose an approach based on the use of edge agent to carry out mediation and task allocation on Processing Units. In a next section, we describe the system architecture that supports our approach. It is based in particular on the use of existing architectures: avatars and IoT-a. In conclusion, we propose to position ourselves in the context of the edge. We discuss the limits of the use of avatar and IoT-a architectures and end our discussion with recommendations for a new architecture that we wish to propose in the near future.

## 2 MOTIVATING SCENARIO

Most actual vehicles integrate a lot of in-vehicle services and a small number of out-vehicle services. Innovative services aim at enabling the car users to be better informed, be safer, more coordinated with its surroundings and to offer a smarter use of transport network. Therefore Intelligent Transportation System and Smart City will be more and more coupled to transform urban mobility. Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) and Vehicle-to-Roadsign (V2R) technologies will be rolled out, and some services oriented architecture such as which data filtering and fusion functionalities will be delegated either to the cloud or to external environment. In-vehicle services will have to consider context-aware applications. For example, effective infotainment system will use the latest (live) information for enhanced user experience. The connected car will serve as a communications hub that will transmit as well as receives data and information, for diagnosis and driving assistance system (by reinforcing deep learning application for example).

The trend toward connected cars will cause disruption and create new opportunities in areas for Advanced Driving Assistance System, Advanced Traffic Management System, Advanced Traveler Information System, Advanced Vehicle Control System, Advances Diagnosis and Maintenance System and Advanced Infotainment System.

Figure 1 depicts one created scenario for future urban mobility in dense and developed cities. The vehicle is connected to other vehicles and to external passive and active devices to help the driver.

All these systems interact and are connected together to offer relevant and appropriate services. For example, the steering system today interact with the suspension to ensure a smooth ride and in the future buildings will interact with the traffic management system to plan ahead the daily journeys.

The objective of the motivating scenario is to predict the electronic requirements of the vehicles of the future from actual electronics. The existing system is studied to consider the likely evolution of the system according to the predicted advanced system presented in the scenario.

Electronics in vehicle typically contains 100-300 micro-controllers or processors, 50 more complex Electronic Control Units (ECU). In general terms, we will talk about processing units (PU) distributed in the vehicle. Two representations co-exist for the grouping of these units in modern vehicles.

- Feature groups: the vehicle is composed of Domain Controller Units (DCU) with similar functionalities (ex: ADAS, Chassis, Body, etc.). This simplifies the manufacture and design of cars.
- Physical location: functionalities from the same location area are grouped together to form a Zone Controller Unit (ZCU) (ex: front left and right, rear left and right, center, etc.). The goal is to reduce wiring in cars.

In all cases, the DCUs and ZCUs are in the vehicle-specific area managed privately by the Gateway. The role of the gateway is to isolate vehicle control from external access to information. Thus the blocks of communication (Inter-Vehicular, Vehicle-to-Road, Infotainment) will have access to the Telematic Control Unit (TCU) to communicate with the environment without being able to influence the safety and security of the vehicle. The communication between the different blocks (DCU and ZCU) are made by an Ethernet link in order to gain speed and bandwidth. Figure 2 shows the two architectures (Feature Group with DCU or Physical localization with ZCU) in a modern vehicle.

Number of processing units (PU) is more and more increasing in the vehicle and will continue to increase in the future. These PUs will also communicate together inside the vehicle and in its surrounding. It is therefore necessary to propose a new approach to optimize this number of PUs and their workloads. Our approach is to be able to share our local computing resources in order to add new functionalities to

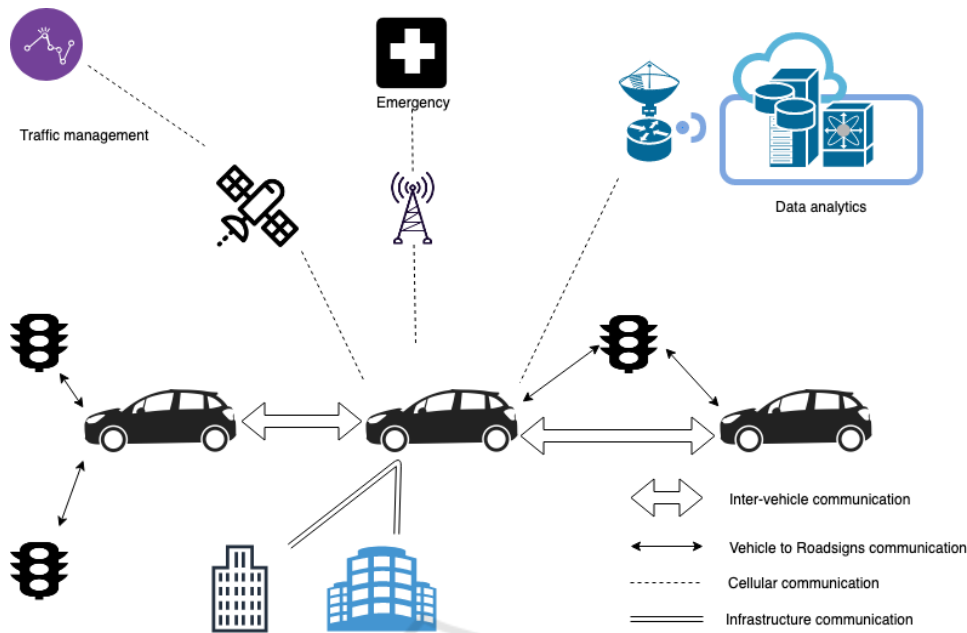


Figure 1: Illustration of a scenario with modern vehicle interactions.

the vehicle without having to redeploy PUs in DCUs or ZCUs. The use of edge computing is a new method to perform new tasks. In case you want to keep the local computing capabilities, it is possible to maintain a link to the cloud. This way of thinking allows us to work independently the availability of resources and thus make task delegation to other environments.

In the next section, we present these infrastructures and detail how they cope with EdgeAgent model.

### 3 A MULTIAGENT APPROACH BASED ON EDGE AGENTS

#### 3.1 Motivation for a Multi-Agent Approach

The motivation is the design of the future electronics platform enabling to compose the capacities of different resources (PUs) into services making sense for vehicle designers and users. Such platform requires to meet different challenges:

- discoverability (C1): allow to discover heterogeneous resources, to be able to plug and unplug resources to the platform;
- connectivity (C2): take into account several communication models (request/response, event-based, publish/subscribe, etc.) in order to allow applications to interact with various resources, as

well as support connectivity disruptions for mobile wireless connected resources;

- reactivity (C3): adapt its structure and behavior to its environment and any potential changes at runtime;
- safety (C4): be reliable and secure so that resources and applications are harmless and avoid privacy issues;
- interoperability (C5): allow any applications to run across heterogeneous resources, so that users can seamlessly interact with resources;
- delegation (C6): identify the most suitable location to execute each code module and deploy these modules on the resources processing unit or on any external infrastructure (V2V, V2I, V2R), instead of completely delegating computation tasks to cloud-based infrastructures;
- scalability (C7): cope with high numbers of resources, heavy calculation processes and/or high quantities of data;
- collaboration (C8): allow a set of resources to exhibit a collective behavior to achieve complex functionalities;
- usability (C9): provide high-level services, so that applications match vehicle-users' needs.

These constraints require decoupling decision-making as much as possible the nodes that make up the system. In other words, this point militates in favour of giving them autonomous decision-making.

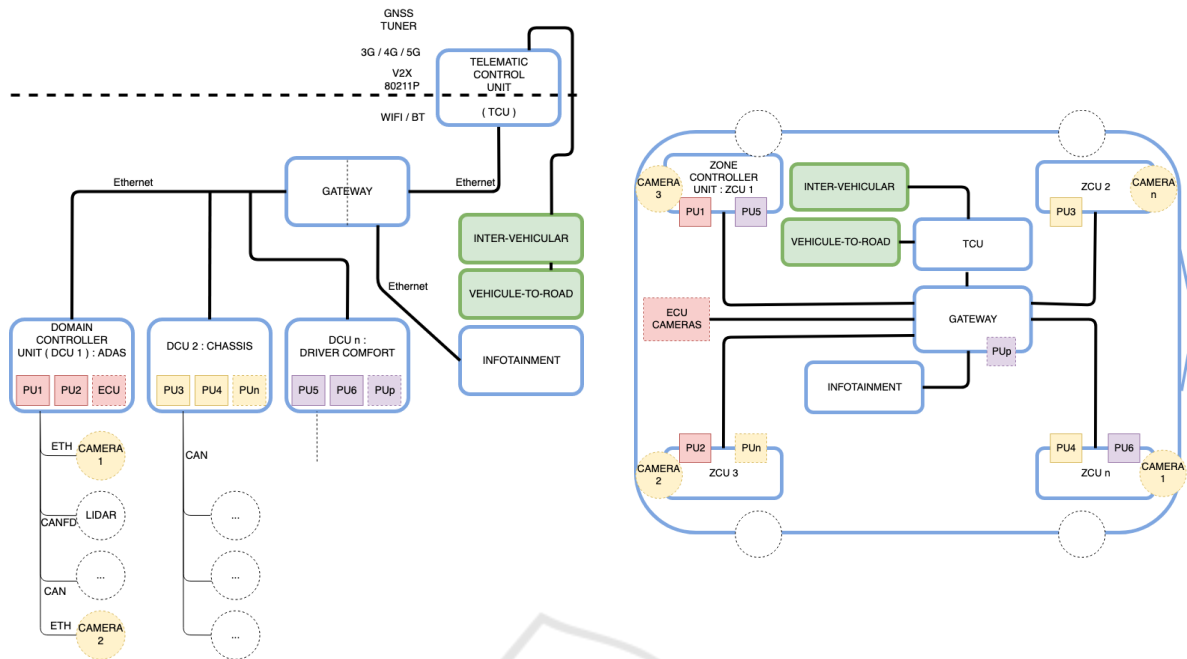


Figure 2: Modern Vehicular Architectures with ECU repartition, Left: Feature groups, Right: Physical localisation.

The corollary is that the overall decision-making process of the system is decentralized. The cloud-based model actually used for vehicle electronics, Figure 3 (Left) cannot cope to the previously presented challenges as resources are centralized. It is necessary for these resources to cooperate and these resources do not need to have all knowledge and skills to meet their individual (local level) and collective (global system level) objectives. Putting in cooperation autonomous agent systems is at the heart of multiagent paradigm. In our context, Edge computing can leverage computing resources that cannot be all connected, as depicted in Figure 3 (Right).

### 3.2 Proposed Approach

In the vehicle approach, the execution of algorithms on electronic vehicle must consider:

- Algorithms that can be deployed in the cloud for execution,
- Algorithms that cannot be deployed in the cloud and must be executed inside the vehicle,
- Algorithms and tasks that can be executed on any ZCU or DCUs as long as the target PUs contains the code/architecture,
- Algorithms and tasks that must be executed on a specific ZCU or DCU for safety, security reasons or for any other hardware constraints.

Sharing resources for future electronic vehicle must consider that each algorithm may have some execution and allocation constraints and the electronic platform must consider all of them. Resources sharing requires a model to supervise all resources, evaluate the workloads and allocate the functions according to the available resources. The proposed model is an EdgeAgent model with three types of EdgeAgent, as depicted in Figure 4:

- EdgeAgent Mediator: receives all requirements and looks for the appropriate and available resources in each ZCU and DCU to negotiate which resources will be used,
- EdgeAgent Allocator: knows the states of each PUs and allocate the tasks,
- EdgeAgent Processor: supervises the execution of the functions on the target PU.

In this model, the EdgeAgent Mediator has a global overview of the state and availability of each resource at any time. The EdgeAgent Mediator allocates functions to resources after EdgeAgent Allocator exchanges. For example, when the parking assistance system is on, the associated algorithm must be executed inside the vehicle on any resources. The EdgeAgent Mediator is looking for available resources and requests the EdgeAgent Allocators to know what resources are available. The EdgeAgent Mediator decides for example as resources in DCU1 will be used and the EdgeAgent Allocator associated to DCU1 allocates the functions to DCU1 resources.

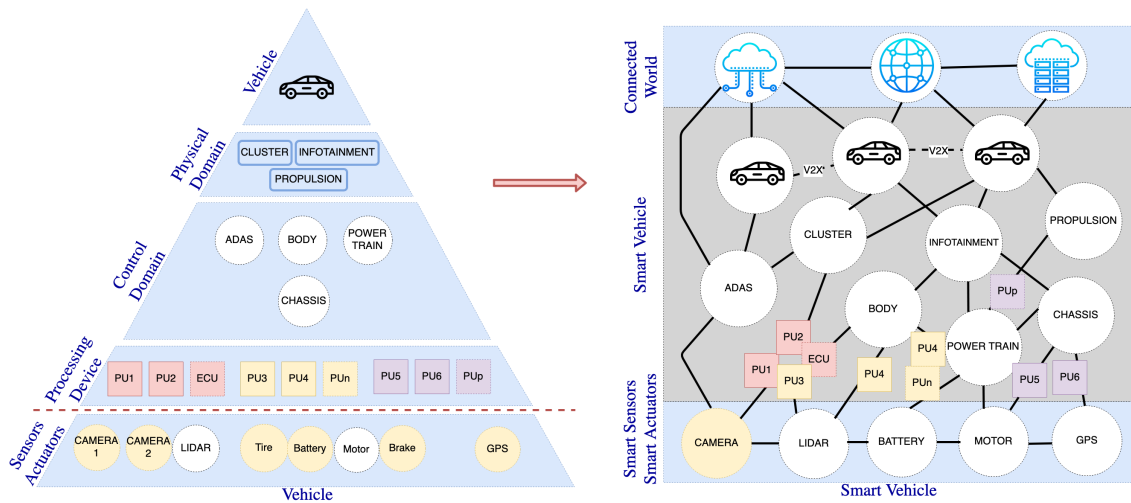


Figure 3: Vehicle evolution with the Edge Computing.

The EdgeAgent Allocator has a constant and precise view of the state of the resources it is in charge. The EdgeAgent Mediator has a global view of all resources of the system whereas the EdgeAgent Allocator knows precisely the states of the associated resources. The EdgeAgent Allocators communicate with the EdgeAgent Mediator so it can decide how and where to execute the functions. The role of the EdgeAgent Processor is to supervise the execution of the functions after deployment by the EdgeAgent Mediator and Allocator. These three types of EdgeAgent will be put in the Edge to manage resources according to services, algorithms and hardware and software requirements. The number and localization of these Agents depend on the electronics systems and their role inside the Edge. First assumption can be done:

- The EdgeAgent Mediator has a global overview of resources and zones and should be in the gateway level,
- The EdgeAgent Allocator has a precise overview of the states of its resources and can be in the gateway level or in the zone level,
- Several EdgeAgent Allocator exchange with an EdgeAgent Mediator,
- EdgeAgent Allocators communicate to EdgeAgent Processors,
- The EdgeAgent Processor should be near resources, the closeness depending on the PUs management,
- EdgeAgent Allocator makes the link between EdgeAgent Mediator and EdgeAgent Processor.

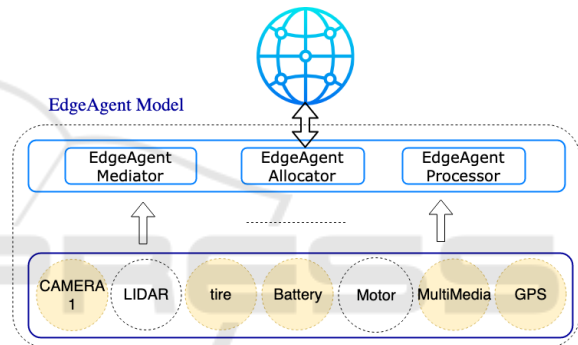


Figure 4: EdgeAgent model.

#### 4 SYSTEM ARCHITECTURE

Embedded systems (ES) are defined as information processing systems integrated into dedicated products (Marwedel, 2018). In our context, ES contains PUs inside the vehicle and PUs outside the vehicle. The purpose of ES is to respond autonomously (in calculation, energy and memory) to a specific services. Lee (Lee, 2007) advances the notion of Cyber Physical Systems (CPS) and defines it as an embedded system to which is added the ability to capture information and/or act on its external environment. As CPSs are now able to communicate their results to each other, they become connected objects (Cervantes et al., 2018). We identify three types of physical objects:

- Complex Objects: These objects provide software services and dedicated communications that offers service interfaces. It is then often trivial to link these objects together or with other software services.

- **Lightweight Objects:** These objects cannot embed servers due to restricted computing capacity but it is often easy to link them to proxies. A proxy can embed a Web server. The (object, proxy) couple can be seen as a complex object that is physically distributed.
- **Bare Objects :** These objects are passive objects that can be detected. Passive objects receive data When such an object is in the range of a RFID reader, the reader receives a byte array. A logical link can then be established between the physical object and the byte array.

We propose two different types of agent architecture to embody the EdgeAgent: *avatars* will allow the most powerful implementation (to address strong requirements  $C_i$ ) while *IoT-A agents* will be the most economical in terms of energy/memory/CPU/bandwidth consumption. An IoT-A agent system can be abstracted by an avatar agent.

#### 4.1 IoT-a Model

In previous works (Carlier and Renault, 2016; Renault and Carlier, 2016), we propose the concept of IoT-a for Internet of Things-agent. Objects are connected and interact on the basis of a common language. The diversity of architectures requires an agent integration model capable of adapting to different hardware levels.

According to the IoT-a model (Figure 5), we propose four main detailed configurations for the implementation of agents within an IoT or PU. These configurations, respecting the constraints  $C_i$  (Ref. 3.1), can be combined because they are independent. The more complex an IoT or PU is, the more it can integrate different agent configurations.

The *configuration 1* proposes the integration of an agent at the hardware level. This first configuration is applicable to connected objects that can be based on a hardware architecture such as microcontrollers or an ASIC (Skhiri et al., 2017). The component integrates a material agent into the silicon and becomes an additional and autonomous function.

In the *configuration 2* an agent is present in addition to the global software system (e.g. operating system). From this configuration, it is assumed that the system has sufficient hardware resources to host an operating system. This second configuration can be implemented on IoTs based on architectures such as ARM, PowerPC or x86 (STMicroelectronics, Broadcom, Intel, etc.). This is the first all-software configuration but it is as close as possible to the hardware and does not suffer the latency of an operating system. The execution of agents has the possibility to be

#### IoT-a Model

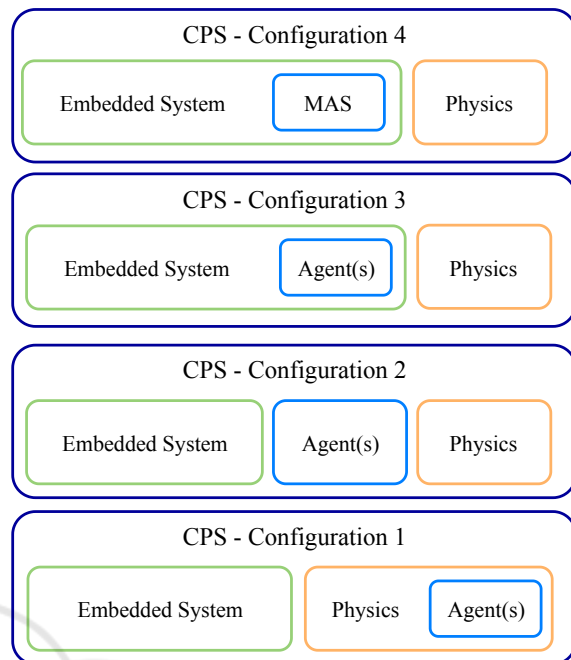


Figure 5: IoT-a traditional model.

real time and/or secure.

The *configuration 3* allows the integration of one or more agents into the software system. Agents are present either in the software kernel to take into account low-level or real-time events, or in the user space to respond to more complex problems that can be multitasking. These agents, like those mentioned in the first two configurations, are autonomous and require a multi-agent platform to exchange with other agents present on different connected objects.

Finally, *configuration 4* proposes that the PU host a complete multi-agent platform. The agents present in the object can interact autonomously and can be in large numbers.

The platform serves as a relay for agents and/or MASs distributed on other types of hardware configuration (or hybrid configuration) or other connected objects in its network. This configuration requires a Processing Unit with sufficient hardware resources to run multiple agents and support different communication protocols at different hardware levels.

#### 4.2 Avatar Model

An avatar is a virtual representation on the Cloud extending objects (Jamont and Ocelllo, 2015; Jamont et al., 2014; Mrissa et al., 2015) (Figure 6). Building this type of platform generally relies on proxy (a projection of a physical object into the Web). Concretely, it is a Web intermediary for requests from

clients seeking resources from other objects. Avatars are not simple proxies. An avatar is an autonomous entity (i.e. an agent). The increase of its knowledge and its skills comes from (1) the Web which is the avatar environment (so an avatar can access to the Web of data and Web Service) and (2) others avatars. Through their avatars, Cyber Physical Systems can be in interaction and particularly in cooperation.

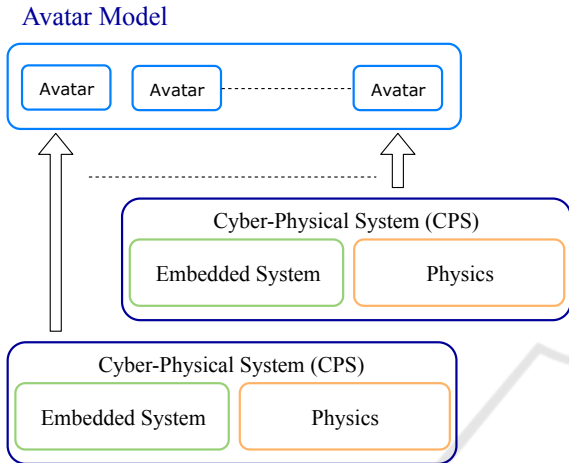


Figure 6: Avatar model.

Some components of the avatar architecture are dedicated to thing control and others implement the autonomous, self-adaptive and collaborative behavior of avatars. The physical setup is decoupled from its logical architecture: an avatar can dynamically adapt. The avatar components are divided into 8 functional modules:

- the *Core Module* includes components that are used in several steps of the avatar lifecycle. Each avatar embeds a Reasoner, used by other components to process semantic information pertaining on the capabilities, functionalities and context. So is the Local Cache, that stores semantic information from diverse sources (thing, repositories, external context) and reflects the current state of the avatar. In particular, the cache loads concepts from the semantic repositories, in order to make them available to other modules through the reasoner. This module is essential to address the multiple concerns targeted by the application through the avatar, while avoiding allocating unnecessary resources. It participates in addressing most of the requirements, and especially (C6).
- the *Interoperability module* provides the other avatar modules with a uniform interface to interact with the resources it is attached to (C1, C5). This interface consists of a set of *capabilities* that represent the thing API. It loads drivers from a

platform repository and uses them to identify the communication schemes understood by the thing; eventually, it uploads onto the thing the appropriate configuration.

- the *Filtering module* restricts functionality exposition and data exchanges according to privacy or security issues, some functionalities should not be achieved by the avatar, they will be filtered by the Privacy manager. The Context Manager has a more complex role.
- the *Communication module* ensures reliable communication with the resources. It selects the appropriate network interface and protocols according to communication purposes and performance needs (C4).
- the *Web service module* allows avatars to communicate with other avatars and with the external world.
- the *Local Functionality module* handles high-level functionalities achievable using the resource capabilities (C9). It relies on semantic technologies to map the resource layer (capabilities) with the application layer (functionalities) in a declarative and loosely coupled manner, ensuring application interoperability with various resources (Mrissa et al., 2014) (C5).
- the *Collaboration module* handles functionalities that require collaboration between several avatars (Cervantes et al., 2018) (C8).
- the *WoT Application module* provides and controls "Web of Things (WoT) application containers" that execute code modules implementing the different aspects of a WoT application (C9). Such containers can be replicated on the resource, on the gateway and on the cloud infrastructure thanks to the deployment manager, so that modules are executed on the appropriate location (C6).

## 5 CONCLUSION

Through this article, we provide a new approach (EdgeAgent) of allocating task (Inguere et al., 2016) for PUs in a modern vehicle. Contrarily to our both separate initial model (IoT-a and Avatar), we propose a better flexibility to manage a computational resources. Now, depending on the complexity of the PU, we can choose IoT-a model for objects with computation resources or Avatar model for simple objects. In the vehicle, the minimization of cables leads the OEM to group the PUs by local area (ZCU) and not by functionality category (DCU). ZCUs are connected

by Ethernet cable to the gateway to increase the transfer rate and reactivity. We can pre-process data locally and provide new functionalities for data manipulation using the edge computing principle. In the Figure 3, the network approach is migrating from a pyramidal to a cubic architecture. The introduction of the concept of edge computing leads us to take the lead in delegating tasks locally in the vehicle. The generic tasks can now be executed on a different PU than the one assigned by default. In the event that the functionality cannot be achieved locally, we send the data to the cloud for cluster server processing.

Following of our work, we will investigate the possibility to extend our negotiation procedure to other cubic edge, allowing by the way a resource sharing between different hardware units. The STMicroelectronics also produces vehicular embedded systems. Later, we wish to distribute our agents on these various units to delegate tasks between these different resources (to vehicle, to road, to smart city).

## REFERENCES

- Carlier, F. and Renault, V. (2016). Iot-a, embedded agents for smart internet of things: Application on a display wall. In *2016 IEEE/WIC/ACM Int Conf. on Web Intelligence, The First Int. Work. on the Internet of Agents*, pages 80–83. IEEE Computer Society.
- Cervantes, F., Ramos, F., Gutiérrez, L., Ocelllo, M., and Jamont, J. (2018). A new approach for the composition of adaptive pervasive systems. *IEEE Systems Journal*, 12(2):1709–1721.
- Inguere, T., Carlier, F., and Renault, V. (2016). Flexible image processing in embedded systems using multi-agents systems. In *14th IFAC/IEEE Int. Conf. on Programmable Devices and Embedded Systems (PDeS 2016)*, pages 164–169.
- Jamont, J., Médini, L., and Mrissa, M. (2014). A web-based agent-oriented approach to address heterogeneity in cooperative embedded systems. In *Int. Conf. on Practical Applications of Agents and Multi-Agent Systems*, pages 45–52.
- Jamont, J. and Ocelllo, M. (2015). Meeting the challenges of decentralised embedded applications using multi-agent systems. *International Journal of Agent-Oriented Software Engineering*, 5(1):22–68.
- Lee, E. A. (2007). Computing foundations and practice for cyber-physical systems: A preliminary report. Technical Report UCB/EECS-2007-72, EECS Department, University of California, Berkeley.
- Marwedel, P. (2018). *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems and the Internet of Things*. Springer Publishing Company, Incorporated, 3rd edition.
- Mrissa, M., Médini, L., and Jamont, J. (2014). Semantic discovery and invocation of functionalities for the web of things. In *IEEE int. conf. on enabling technologies: infrastructure for collaborative enterprises*.
- Mrissa, M., Médini, L., Jamont, J., Sommer, N. L., and Laplace, J. (2015). An avatar architecture for the web of things. *IEEE Internet Computing*, 19(2):30–38.
- Renault, V. and Carlier, F. (2016). Triskell3S, une plateforme embarquée multi-agents pour les IoT-a. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2016)*, pages 181–190.
- SA, R. (2019). Automotive System-on-Chip (SoC). <https://www.renesas.com/us/en/solutions/automotive/soc.html>.
- Skhiri, R., Fresse, V., Jamont, J., and Suffran, B. (2017). Challenges of virtualization fpga in a cloud context. *IEEE Int. Conf. on Computational Intelligence and Virtual Environments for Measurement Systems and Applications*.