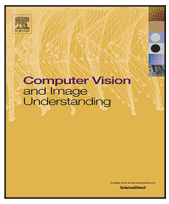
Computer Vision and Image Understanding 259 (2025) 104437

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu



CRDT-based knowledge synchronisation in an Internet of Robotics Things ecosystem for Ambient Assisted Living

José Galeas [a], Alberto Tudela [a], Óscar Pons [a], Juan Pedro Bandera [a], Antonio Bandera [a,*], Pablo Bustos [b]

[a] Dpto. Tecnología Electrónica, E.T.S.I. Telecomunicación, Universidad de Málaga, Campus de Teatinos s/n, 29071 Málaga, Spain
[b] RoboLab, Escuela Politécnica, Universidad de Extremadura, Cáceres, Spain

ARTICLE INFO

Keywords:
Ambient assistant living
Internet of Robotics Things (IoRT)
Delta conflict-free Replicated Data Type (δ-CRDT)
Publish/subscribe
Knowledge consistency maintenance

ABSTRACT

Integrating IoT and assistive robots in the design of Ambient Assisted Living (AAL) frameworks has proven to be a useful solution for monitoring and assisting elderly people at home. As a way to manage the information captured and assess the person's condition, respond to emergencies, promote physical or cognitive exercises, etc., these systems can also integrate a Virtual Caregiver (VC). Given the diversity of technologies deployed in such an AAL framework, deciding how to manage knowledge appropriately can be complex. This paper proposes to organise the AAL framework as a distributed system, i.e., as a collection of autonomous software agents that provide users with a single coherent response. In this distributed system, agents are deployed locally and handle replicas of the knowledge model. The problem of merging these replicas into a consistent representation, therefore arises.The δ-CRDT (Conflict-free Replicated Data Type) synchronisation mechanism is employed to ensure the eventual consistency with low communication overhead. To manage the dynamics of the AAL ecosystem, the δ-CRDT is combined with the publish/subscribe interaction protocol. In this way, the performance of the IoT, the robot and the VC, through the functionalities that depend on them, is efficiently adapted to changes in the context. To demonstrate the validity of the proposal, two use cases have been designed in which a collaborative response from the system is required. The first one deals with a possible fall of the user at home, while the second one deals with the problem of helping the person move small objects around the flat. The measured values of latency or consistency in the data show that the proposal works satisfactorily.

1. Introduction

Addressing the problem of an ageing population by increasing the number of places in nursing homes and health care services seems utopian. The already high percentage of the population over 65 years in developed and developing countries will continue growing in the near future. According to current projections, by 2050, this percentage will be over 25% in Europe and North America (Anon, 2022). Moreover, while this *silver society* will inevitably require a higher amount of health and social care services, the evolution of the Caregiver Support Ratio (CSR) points towards a very limited availability of carers to deal with this increasing demand (Ribeiro et al., 2022). One of the strategies proposed to address this situation is the *Active Ageing* paradigm, which aims to allow elderly people to remain as autonomous and active as possible for as long as possible (Anon, 2023a). The objective is to increase the person's welfare and keep her as a valuable asset in society while reducing the care or institutionalisation costs, as institutionalisation itself is delayed or even avoided. Active ageing policies must ensure privacy, safety and quality of care for elderly people (Anon, 2023a). They, therefore, include personalised treatments, long-term monitoring, and the integration of monitoring, communication and therapy technologies at home. There, elderly people remain in a familiar environment, with the consequent advantages at the psychological and physical levels (Ratnayake et al., 2022). Care for these people at home should follow a preventive approach, avoiding complications requiring care by medical professionals or transfer to a hospital as much as possible. The possibility for the elderly person to remain at home while monitoring their condition at all possible levels (physical activity, hours of rest, etc.) is now possible thanks to technological developments in ubiquitous computing and sensing. Specifically, an Ambient Assisted Living (AAL) environment integrates technologies (sensors, actuators, computational resources), solutions and services to

* Corresponding author.
E-mail address: ajbandera@uma.es (A. Bandera).

https://doi.org/10.1016/j.cviu.2025.104437
Received 4 March 2025; Received in revised form 14 May 2025; Accepted 24 June 2025
Available online 26 June 2025

enable assistance to the person (Cicirelli et al., 2021). They are everyday living environments, such as the home itself, and their deployment must enable the person to lead a full and autonomous life (Blackman et al., 2016).

An AAL environment is underpinned by the integration of a set of technologies. The development of the Internet of Things (IoT) is one of the most relevant. The idea of connecting objects around us in a real environment to the internet allows control and decision-making modules to have up-to-date information in near real time, thus anchoring a virtual model to the real environment. The application of this technology, together with other relevant technologies such as Big Data or digital twins, makes it possible to create intelligent AAL environments that improve the quality of life of the people who inhabit them (Hail and Fischer, 2015). The robot is another element that can be integrated into an AAL environment. Its combination with the Internet of Things (IoT) is known as the Internet of Robotic Things (IoRT) (Kara and Carlaw, 2014). In IoRT, information obtained from different sensors can be processed locally and fused and used to control the behaviour of an agent in the physical world. In this way, the IoT ecosystem can provide the robot with greater situational awareness, enabling it to perform its tasks better. Moreover, by being connected to the internet, the robot can access content that it can use in interacting with the people around it. Although the presence of a robot in such an ecosystem will increase the price, the advantages of their integration into the AAL environment can be relevant. In this sense, perhaps the most widely advocated is that the robot provides a more natural and intuitive interaction interface with the person. This is a debatable question, which depends on each user, and which competes with cheaper options such as a tablet or conversational assistants. Some studies do show that the robot has advantages over tablets or smartphones (Hammer et al., 2017; Deublein and Lugrin, 2020). The fact is that it is an animated entity, which can move around the environment, zoom in or out depending on the moment, be equipped with specific sensors, proactively engage people in social interactions, and facilitate these interactions with a multimodal approach (voice, touchscreen). In short, the difference between IoT and IoRT is that, in the latter, the system includes devices, robots, that not only act in the real physical world, but move through it and interact with people more closely than other devices could. Because of this, robots have been incorporated into many AAL proposals (Angulo et al., 2015; Coradeschi et al., 2014; Wengefeld et al., 2022). Another element that can be incorporated into the AAL environment is the virtual caregiver (VC) (Luperto et al., 2022). Their mission is to provide higher level reasoning skills, with the aim of detecting, for example, anomalies in the person's daily behaviour. In a typical deployment, this capability is placed in the cloud, addressing techniques that may include scheduling, spatio-temporal reasoning, or time series analysis (Vuono et al., 2018; Luperto et al., 2022).

The described scheme for AAL combines IoT and robotics, as well as Artificial Intelligence or Machine Learning techniques, which are necessary for the processing carried out on these elements and the design of the VC. In addition, mechanisms will be needed to allow them to transmit to or receive data from software agents implemented on different platforms. Although current devices try to be designed to communicate with any other, creating a network of interconnected elements, the lack of a standard and the continuous emergence of designs that seek to differentiate themselves from the rest with new protocols, makes that one of the challenges facing the design of an IoRT ecosystem is that of interoperability. Parallel to this is the problem of determining what information needs to be passed from one element to another for the system to generate the best response. To solve both issues, many proposals use widespread protocols (Naik, 2017; Luperto et al., 2022), such as MQTT (Message Queuing Telemetry Transport), to connect the elements of the ecosystem. In this framework, the knowledge that the higher-level components have of the context is limited, conditioned by the specific topics to which they subscribe. These components are also located in the cloud, which, in principle, ensures scalability in terms of computational and storage resources. However, this worsens response times and makes it necessary to improve security protocols as critical information leaves the local framework. This paper proposes an IoRT implementation for AAL. The IoT environment makes it possible to track a person's movement at home and monitor certain vital parameters. The robot integrated into the ecosystem has the hardware and software resources to move autonomously in the environment and interact with the person. The VC, which determines the behaviour of the IoT system and robot, is encapsulated in a collection of Behaviour Trees, from which the one to be executed is selected using a self-adaptation system that responds to changes in the context. The main novelty in our proposal is that all software agents, whether embedded in the robot, the IoT environment or as part of the VC, manage a unique knowledge representation. This representation, or working memory, does not exist as a single instance of a data structure accessed by the agents but as a set of local replicas owned by them that interchange messages every time a change is locally made to one of the copies. The synchronisation of these replicas is based on the use of Delta Conflict-free Replicated Data Type (δ-CRDT) (Almeida et al., 2018; Bustos et al., 2021) to guarantee consistency and of a publish/subscribe DDS middleware based on the RTPS (Real-Time Publish–Subscribe) protocol (Bustos et al., 2021). The main advantage of this approach is that the higher-level components have access to all the knowledge generated by the perceptual components, regardless of their location in the ecosystem. Additionally, they can use an agreed naming system for all concepts and instances represented in the working memory. Thus, for instance, the objects detected by the IoT system or the last detected position of the person are immediately accessible to the components running on the robot. In addition, the whole architecture adopts a very horizontal structure, and deliberate behaviour does not take full control without allowing more reactive behaviour to be executed when the context determines it is needed.

1.1. Contributions

In classic hierarchical architectures for IoRT, knowledge representation is distributed between layers, complicating the integration of software agents as they must decide in which particular layer they are placed and how to interact with agents that may be in another layer (Sayeed et al., 2022). In these situations, it is complex to decide what specific knowledge should be shared between layers (e.g. what topics publish in a MQTT-based approach). The behaviour of the framework is typically organised into services. However, when it is decided to run a certain service, all resources (computational, sensors, and actuators) become controlled by this service. The occurrence of a priority situation is then not detected (unless the service in question considers this to be the case). In our proposal, the knowledge is encoded as a single blackboard whose replicas are managed by all software agents present in the AAL (whether they are part of the robot, the IoT, or the VC). In this inner model, geometric and symbolic elements are integrated into a directional graph. The updating of the graph by the whole set of software agents will cause subgraphs (semantic relations) to appear. These relations automatically trigger reactive or deliberate action by other software agents. The main contributions of this work are

- The whole AAL framework is encoded following the guidelines of the CORTEX software cognitive architecture (Bustos et al., 2019). CORTEX is a mature approach based on the fundamental principle that any software agent added to it must interact with all other agents through updates in a shared runtime model (Romero-Garcés et al., 2022).
- The runtime model storing the knowledge is distributed, and each software agent manages a local replica of this model. Consistency of the copies is guaranteed using the δ-CRDT type (Almeida et al., 2018). Knowledge transfer is managed using a publish/subscribe DDS middleware based on the RTPS (Real-Time Publish–Subscribe) protocol.

- The proposed solution is able to address the challenge of interoperability in the deeply multi-platform framework of an IoRT solution. In the use cases proposed in this paper, we show how our proposal facilitates the integration of software agents developed in micro-ROS (with FreeRTOS), ROS 2, RoboComp or Mira, deployed using wired or wireless protocols.
- The deployment of the proposed complete system in a small flat is evaluated with use case examples co-designed with care professionals. The VC is responsible for determining the behaviour of the whole framework. A self-adaptation software agent is proposed to commutate between behaviours. Each specific behaviour is encoded as a Behaviour Tree (Romero-Garcés et al., 2022).

1.2. Organisation of the paper

The rest of the article is organised as follows: Section 2 briefly introduces the problem of interoperability, an issue that continues to emerge as IoT ecosystems continue to integrate new sensors or devices without specific standards being defined. In addition, this Section briefly introduces the typical organisation in layers of software architectures for IoRT-based AAL and discusses the functionalities that the robot can provide when deployed in such a system. The distributed runtime model employed for managing the shared knowledge and guaranteeing consistency is described in Section 3. Section 4 illustrates the instantiation of our solution for AAL. This section emphasises how CORTEX eases the integration of different technologies and protocols. Section 5 present a pair of simple use cases in which the IoRT system makes use of the robot as a natural interaction element. It also provides a brief analysis of representative quantitative metrics that allow for assessing the performance of the proposed system. Section 6 discusses several key aspects of the proposal and provides insights into current and further related research. Finally, Section 7 concludes the paper.

2. Related work

2.1. The multi-platform challenge

AAL technologies cover a wide variety of platforms that are in use in a wide range of contexts. If we focus the application framework on the care of the elderly, the active ageing paradigm sets the key objectives for these technologies at monitoring, communication, and therapy (Anon, 2023a). Within a framework of prevention rather than continuous action, monitoring the elderly person at home allows professionals to assess the person's condition and its evolution over time, and greatly reduces reaction times in case of emergency. The most immediate way to monitor the state of a person living at home is to deploy sensors there. The devices can be wired or wirelessly networked and, depending on the computing power required, are usually microcontrollers or single-board microprocessors. By incorporating the robot into the AAL ecosystem, we can improve this monitoring (Anon, 2015; Slavisa et al., 2022). On the one hand, the robot acts as a mobile sensor station. On the other hand, the robot can also talk to the person, thus obtaining data that the sensors cannot capture. These interaction capabilities also allow the robot to act as an interface for the person to communicate with family and friends, as well as with external experts or caregivers. By incorporating the robot, the repertoire of functionalities provided by the AAL ecosystem is increased (Slavisa et al., 2022): The robot can be used for therapeutic purposes, to manage rehabilitation through serious games, or to capture comprehensive geriatric assessment (CGA) data. To address these issues, the robot often receives computational support from an external server so that the ecosystem can incorporate new frameworks. In the work of (Vuono et al., 2018), this external support is provided by the so-called virtual caregiver (VC). In their proposal, this VC is the responsible of (i) analysing the data provided by an IoT system and an Activity Centre, accessible through a tablet (the so-called Community-Based Activity Center (CBAC)); and (ii) generating suggestions to the users with the aim of improving their quality of life (activities, games...). The VC is located in the cloud and is powered by MQTT, through specific topics (Luperto et al., 2022). The end result is that the IoRT environment deployed for AAL typically includes IoT and autonomous robots, but also intelligent connectivity, distributed and federated edge/cloud computing, and Artificial Intelligence (AI). In some cases, it may also consider digital twins (DT), distributed ledger technologies (DLT), virtual/augmented reality (VR/AR) and swarm technologies. Clearly, it has a strong cross-platform nature, which requires complex software architectures to cope with it (Modoni et al., 2017).

Consequently, the implementation of an IoRT-based AAL ecosystem faces many scientific and technical challenges (Modoni et al., 2017; Papadopoulos et al., 2020). One of these relevant issues is the lack of interoperability between the various elements involved. This problem can result in significant data captured by some sensors not being taken into account in the analysis carried out by certain algorithms. In short, although the data are captured, they are not incorporated into the knowledge that the system stores and manages of the entire context. The problem is due both to the existence of different communication interfaces (as devices are diverse in hardware, operating system, or programming language) and to the need to define a knowledge representation mechanism that integrates all sources of information, regardless of the formats they use. Some frameworks employ standard platforms such as universAAL,[1] ECHONET Lite,[2] and Matter (Singh, 2023). UniversAAL is an EU-FP7 project launched in 2010. The ECHONET Consortium includes more than 200 Japanese companies. Matter was released in 2022 by the Connectivity Standards Alliance (CSA). It uses the Zigbee Cluster Library as the application profile. [Moreover], integration efforts among protocols/platforms have been carried out to improve interoperatibility. For instance, the CSA affirms that currently there are available more than 150 certified Matter bridges. It can also include robotics platforms: the SwitchBot Floor Cleaning Robot S20 Pro is the first Matter certified robot vacuum,[3].

The design of a bridge between two platforms can be a complex challenge, especially when the problem is approached as one of integrating devices using one protocol into a different ecosystem (Pham et al., 2024). By not specifying encryption primitives, ECHONET Lite offers a plug-and-play mechanism that allows devices to join its network without the need for additional configuration steps.[4] The scheme is based on the device having a Network Address, which provides a unique identification for the node, and the Profile Object, which serves as the entry point for interacting with the device. The [functionality of the device] is specified as one or more Device Objects. Another option for integration is to maintain networks that internally use different protocols and define bridges that allow each network's knowledge of the context to be moved. In the proposal by Bui et al. (Bui and Chong, 2018), a Pepper robot is integrated into an ECHONET-based smart house. Additional home applications and user interfaces are also added. To solve the integration issue, universAAL nodes and middleware are employed for bridging. Thus, the Robot Controller, Home Gateway, and the Cloud Server install a universAAL middleware module (Fig. 1(Left). Each element in the AAL ecosystem provides specific functionalities. The reason for using universAAL is that it can run on different operating systems and support different communication protocols. In the proposal by Luperto et al. (2022), the challenge of communicating the IoT, robot, and VC of an AAL ecosystem is addressed using MQTT. The scheme is illustrated in Fig. 1 (Right). Each component of the framework maintains its own knowledge representation, being the relevant information transferred using MQTT messages. This scheme is very popular for IoT systems, with known advantages and disadvantages (Naik, 2017).

[1] https://www.universaal.info/

[2] https://echonet.jp/

[3] https://vacuumwars.com/switchbot-reveals-k20-pro-the-multitasking-robot-and-s20-pro/ Accessed on 25 February 2025

[4] https://echonet.jp/

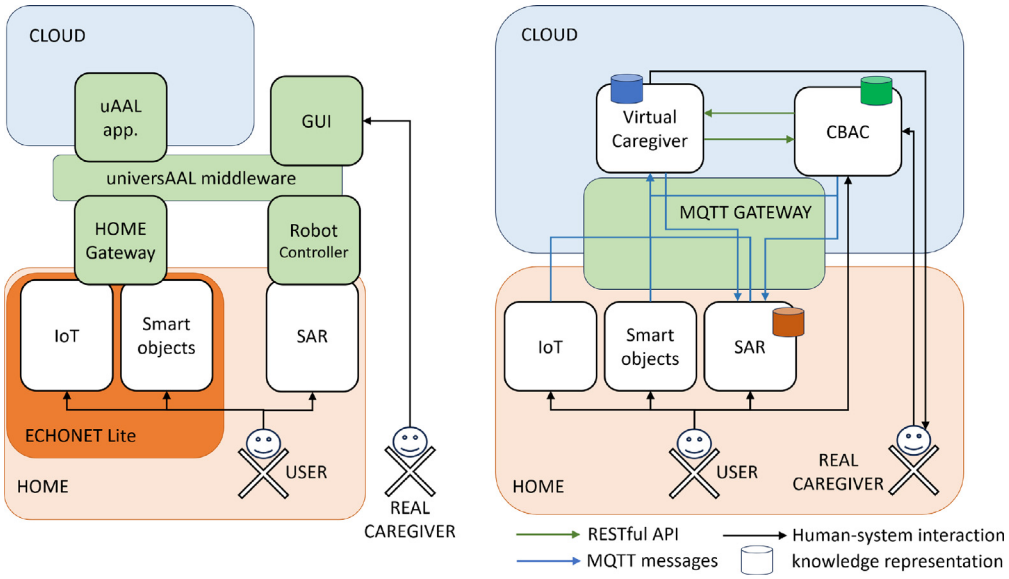


Fig. 1. (Left) Integration of a Pepper robot into the iHouse (Bui and Chong, 2018); and (right) AAL scheme proposed in the Movecare EU project (Luperto et al., 2022) (see text for details).

2.2. Organisation and architectures for IoRT-based AAL

The previous section has outlined how software architectures try to address the multi-platform problem. From a more generic perspective, an IoRT system is organised in a five-layer hierarchical structure (Sandhu et al., 2024) (Fig. 2). The physical layer contains the sensors and actuators, as well as the hardware of the robot itself. The goal of this layer is to connect the physical world to the digital world by converting physical signals into signals that are intelligible to the algorithms and procedures of the higher layers. The Data link and Network layers are responsible for converting the captured data and managing the transmission of data between devices, respectively. Both layers use control and communication protocols for data transfer. The transfer is completed at the Transport layer, which is also responsible for functions such as error control, data flow control and user interface.

Finally, the data obtained is used by the Application layer to present the processed information to users and caregivers. The data are processed and transmitted to a central server that controls the intervention mechanisms to provide assistance in real time, using robots if necessary.

This scheme, which is replicated in IoRT systems (Vermesan et al., 2020), has the disadvantage of keeping decision making far away from the Physical layer, so that any problems in the communication channel delay the intervention. In previous work using CORTEX (Calderita et al., 2014), between the Physical layer and the Network layer, there is an action layer, which allows the robot to be more autonomous in its behaviour and to help the person. These are actions that allow the robot to behave in a deliberative and/or reactive manner, but always bounded by a codified protocol designed in collaboration with the medical experts. The Application layer still exists, both to present the information to the user and to host high-level data processing and decision-making algorithms. This proposal follows this scheme of work, but addresses an optimised implementation of knowledge management, necessary to work with a distributed IoRT ecosystem, which includes more hardware and software processing elements.

2.3. Expected services of IoRT in AAL

Robots adequately integrated in AAL environments can significantly increase the amount of services provided by these environments (Rasch et al., 2019). More than ten years ago, the GiraffPlus project (Coradeschi et al., 2014) already proposed to equip the person's environment with a network of sensors to monitor her condition. The robot can communicate with relatives or friends in case certain states are detected in the person (nervousness, anxiety...). The main limitation of this robot is that it is teleoperated. Filling the communication gap between users, relatives and caregivers is also the aim of other projects (Wengefeld et al., 2022) and recent commercial solutions (Anon, 2023b), where a robot is used as a communication interface. Moreover, a robot able to communicate with the user can also remind her to take medication (Linner et al., 2015), provide information and stimulation to perform everyday activities (Meyer and Fricke, 2020; Embarak et al., 2021), or announce upcoming events (Iglesias et al., 2024). Unlike other solutions, the robot can look for the user around the environment when an incoming call comes in and accompany her in her daily activities while the call is being made. This means that relatives who live far away can use a smart remote control to keep an eye on things in the house or use telepresence to assist them with certain activities (Anon, 2023b). In Bui and Chong (2018), the user communicate with a robot to control devices in the smart home. The proposal employs an ECHONET standard home network that integrates the robot and the environment sensors and actuators. Human activity recognition (HAR) is a crucial task for an AAL ecosystem. The sensors mounted on the robot may be useful to improve the HAR but, at the same time, it may be that the HAR, carried out by the IoT sensors, allows the robot to improve its interaction with the user (Karim et al., 2024a). Thus, the use of smart clothing and Deep Learning emerges as an alternative to recognise human activity instantly and accurately (Nyangaresi and Shanshool, 2024). The data provided by devices embedded in the body area of the person should be able to be integrated into the knowledge model managed by the IoRT system in the same way as static sensors do. The RiSH proposal (Do et al., 2018) addresses human body activity detection using static sensors and those embedded in the robot. Proposed applications are human position tracking and human activity monitoring. The recognition of activities, normal or abnormal, is also the service provided by the AAL environment and the robot in Mojarad et al. (2023)'s work. The proposed framework includes long-term memory models (LSTM) and reasoning based on Probabilistic Answer Set Programming (PASP). The camera that can be carried by the robot can be used as the basis for carrying out HAR. In the HADE architecture (Karim et al., 2024b), the proposed models are built on a dataset of images captured mainly from smartphone cameras. The system obtains a very relevant accuracy at the cost of

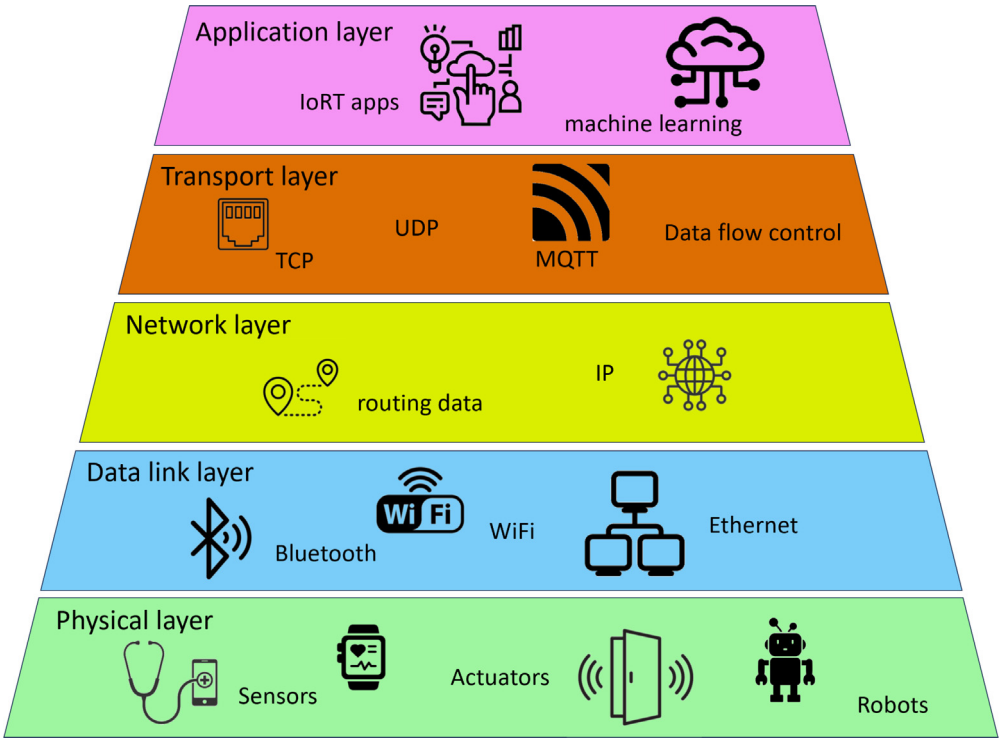


Fig. 2. Typical organisation in layers of an IoRT-based AAL.

using an NVIDIA Tesla V100 GPU. In case this GPU cannot be easily integrated into the robot (e.g. because of its high power consumption), the system could deploy this device outside the robot. The robot would send the images for processing in the IoT environment. Fall detection is also a very relevant functionality for AAL environments (Antonello et al., 2017). Solutions implemented in AAL typically employ cameras, accelerometers, or fuse data from different sources. For example, Jiang et al. (2024) use an infrared array sensor, Liu (2023) fuses information from infrared sensors and accelerometers, or Rodrigues et al. (2023) use ultra-wide band (UWB) technology and inertial measurement unit (IMU) data. In all cases, deep learning algorithms are used to implement the detection process. The use of a robot allows for a fast initial check on the person's condition and immediate communication with caregivers or emergency services. Commercial solutions addressing this functionality are already available on the market (Anon, 2023b). In other proposals, the robot acts as a virtual therapist for physical or cognitive follow-up activities (Soldatos et al., 2021; Calderita et al., 2020). Other functionalities are meal assistance (Hanheide et al., 2017) or transport items within the home (as the Turtlebot robot deployed in the RmR AAL ecosystem (Linner et al., 2015)). In the ALMI project, the TIAGo robot uses both its speech interaction for voice instructions and its object manipulation capabilities to help a user with mild motor and cognitive impairments to prepare a meal. Speech interaction is also the main functionality deployed by the robot in the proposal by Gulzar et al. (2023).

Calderita et al. (2020) propose the use of CORTEX (Bustos et al., 2019) to build the software architecture that controls the entire AAL ecosystem. This implementation defines services that, like the human–robot interaction service or the social navigation service, handle their own internal representations, annotating more limited knowledge in the shared knowledge representation. Decision-making is carried out outside the robot. This paper describes a similar scenario, where CORTEX is adopted as the software architecture of the entire IoRT ecosystem. However, in our implementation, the software agents are more atomic, and handle knowledge that is annotated in a distributed, shared representation. The goal is to achieve the fastest possible response to any change in the context. As described in Section 3, the runtime model has been modified to improve its management by the set of software agents. The AAL is organised as a distributed system, with elements managing local copies of this knowledge model, whose consistency is ensured by Delta Conflict-free Replicated Data Type (δ-CRDT).

3. Using CRDT for maintaining knowledge consistency

Fig. 3 shows the proposed scheme for integrating the different elements that make up the AAL ecosystem. The scheme has some similarities with the EU Movecare project (Luperto et al., 2022), but, in our case, the knowledge representation is unique and shared by all elements in the AAL ecosystem. The AAL is organised as a distributed system, with elements managing local copies of the representation, whose consistency is ensured by CRDT. Although, as shown in Fig. 1 (right), both schemes could be the same (substituting the information transfer mechanism, MQTT for DDS), in the Movecare case it is not intended that the knowledge representation is unique, creating (through MQTT messages) a higher level representation that is shared (although not the only one for all elements). In our case, the aim is for the representation to be unique. If the background to this scheme is sought, it replicates, in an AAL ecosystem, the guidelines that characterise the CORTEX cognitive architecture proposal (Bustos et al., 2019; Marfil et al., 2020).

CORTEX is a multi-agent architecture designed to facilitate the creation of information flows between different types of memories and modules (Bustos et al., 2019). These flows are pumped in and out by software agents responsible for different tasks, reactive or deliberative, and of different granularity (from sensors that measure heart rate to those that make decisions using automatic planning). In CORTEX, all agents are connected to a working memory $\mathcal{W}$, the so-called Deep State Representation (DSR). Software agents must edit the DSR to create and maintain an updated context that is accessible to the rest of the agents. The DSR is the only way for agents to communicate with each other. This forces the knowledge present in the DSR to reflect what the robot knows about the environment, its own state, or the tasks in progress, as

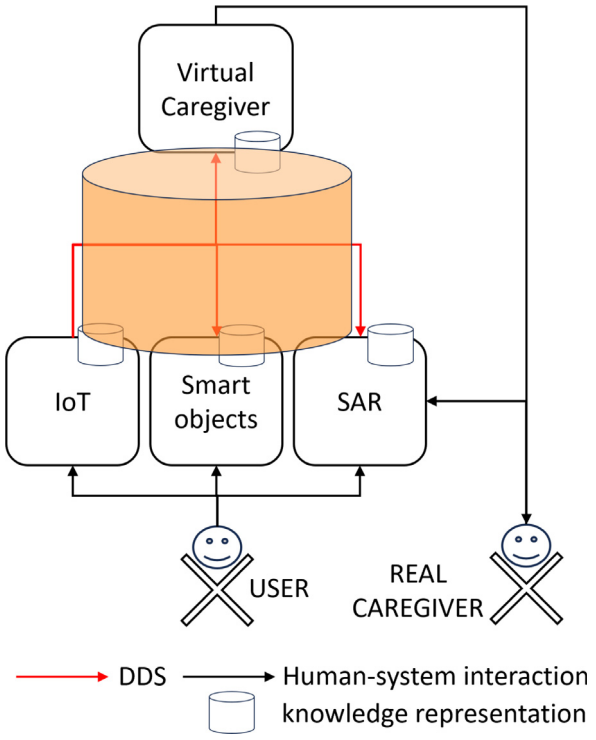


Fig. 3. AAL scheme based on CORTEX (see text for details).

perceived by the agents. The amount of context in the DSR at any time is controlled at this time by the software agents. Restricting inter-agent communication to changes in the DSR keeps the architecture simpler. It avoids inter-agent flows that can introduce unwanted side effects that are difficult to detect and debug. As all agents keep an updated copy of the DSR, all are promptly informed of changes in the broader context and can properly react to them (Bustos et al., 2019; Marfil et al., 2020).

The following sections present the structure of the DSR and how its synchronisation is managed in a distributed framework.

3.1. The deep state representation

As mentioned above, the working memory, DSR, is at the core of CORTEX and represents the current state, which the software agents use to create the AAL's behaviour. This representation is a distributed, directed, multi-labelled graph, which acts as a runtime model (Romero-Garcés et al., 2022). The graph vertices are elements of a predefined ontology, including raw sensor data, and the edges can encode probabilistic geometric transformations and constraints, logic predicates or meaning postulates. Thus, for example, the graph could maintain the robot's position with respect to the current room and all elements relevant to an ongoing interaction with a person. Both vertices and edges can store a list of attributes of a predefined type. Relative probabilistic poses between vertices are represented as rotation and translation vectors with their covariance. Fig. 4 shows a simple example. The **person** and **robot** vertices are geometrical entities, both linked to the **world** (a specific anchor providing the origin of coordinates) by a rigid transformation. But, at the same time that the geometrical relationship between both vertices ($RT^{-1} \times RT'$) can be computed, the **robot** is interacting with the **person**. Furthermore, an agent can annotate that currently the **robot** is _not **speaking**. Edges represent relationships among symbols. As the figure shows, two symbols may have several types of relationships. The overall representation of the geometric relations among the vertices must follow a kinematic tree structure, where each element has a unique parent and relative pose between both. This decision was made to facilitate the computation of coordinate frame transformations among distant vertices in the tree.

To mathematically express the structure of the DSR, we will use the concept of quiver. A quiver is a quadruple consisting of a set V of vertices, a set E of edges, and two maps $s, r : E \to V$. These maps associate with each edge $e \in E$ its starting $\mathbf{u} = s(e)$ and ending vertex $\mathbf{v} = r(e)$. For simplicity, we can denote by $e = \mathbf{uv} : \mathbf{u} \to \mathbf{v}$ an edge with $\mathbf{u} = s(e)$ and $\mathbf{v} = r(e)$. In a quiver, a *path* of length m is a finite sequence $\{e_1, \dots, e_m\}$ of edges such that $r(e_k) = s(e_{k+1})$ for $k = 1...m-1$. A path of length $m \geq 1$ is called *cycle* if $s(e_1)$ and $r(e_m)$ coincide.

Using this notation, the DSR can be described as the union of two quivers (Bustos et al., 2019). One of these quivers is associated with the symbolic part of representation, $\Gamma_s = (V, E_s, s_s, r_s)$, while the other is associated with the geometrical relationships, $\Gamma_g = (V_g, E_g, s_g, r_g)$. Significantly, within Γ_s there are no cycles of length one. That is, there are no *loops*. Moreover, given a symbolic edge $e = \mathbf{uv} \in E_s$, we cannot infer the inverse $e^{-1} = \mathbf{vu}$. Finally, a symbolic edge $e = \mathbf{uv} \in E_s$ can store multiple values. On the other hand, the quiver Γ_g defines a directed rooted tree or rooted tree quiver (Katter and Mahrt, 2014). Within Γ_g there are no cycles (acyclic quiver), and any two vertices $\mathbf{u}, \mathbf{v} \in V_g$ can be connected by a unique simple path. For each geometric edge $e = \mathbf{uv} = RT$, we can define the inverse of e as $e^{-1} = \mathbf{vu} = RT^{-1}$. The *kinematic chain* $C(\mathbf{u}, \mathbf{v})$ is defined as the path between the vertices $\mathbf{u}$ and $\mathbf{v}$. The equivalent transformation RT of $C(\mathbf{u}, \mathbf{v})$ can be computed by multiplying all RT transformations associated with the edges on the paths from vertices $\mathbf{u}$ and $\mathbf{v}$ to their closest common ancestor $\mathbf{w}$. Note that the values from $\mathbf{u}$ to the common ancestor $\mathbf{w}$ will be obtained by multiplying the inverse transformations. Removing the no-cycles restriction in Γ_s would delve into a more complex mechanism to obtain point-to-point coordinate transformations, where multiple paths would be possible for each query, and the user would have to select which one to choose.

In Fig. 4, the creation and updating of the edges denoting a relative pose RT or a logical relation (in, interacting) is done by the agents. This is the typical scheme for updating the DSR graph. However, some of the knowledge used by the software agents or stored in the DSR is provided a priori. For instance, this is the case of the metric map of the environment in which the IoRT ecosystem is deployed and used by the navigation stack. This map is read from a file when the Navigation agent is started and could then be propagated to the others on demand. Vertices representing a room (e.g., the **bedroom** one in Fig. 4) are defined a priori from this metric map, shaping a topological map created offline using a SLAM algorithm and manual editing. These vertices are linked to the **world** vertex through in edges. The dynamic relation of the robot with its represented space is updated by the agent that tracks its localisation in the world frame. When the robot enters or leaves a room, the corresponding edge is updated, and all other agents receive the modification.

3.2. Guaranteeing the consistency of a distributed DSR

The overview of the AAL ecosystem described in Section 4 illustrates the complexity that CORTEX can handle, with blocks that, in the actual implementation, can be software stacks composed by different components. All agents contribute from their functional domains to maintain an updated representation in the working memory (DSR), which makes data access a potential problem. The DSR is a distributed data structure designed as a three-layered architecture. Fig. 5 shows a schematic representation of the main elements. The first layer uses a publish/subscribe DDS middleware based on the RTPS (Real-Time Publish–Subscribe) protocol and configured to use UDP (User Datagram Protocol) reliable multicast (Bustos et al., 2021). The second layer implements a series of conflict-free replicated data types (CRDTs) that guarantee eventual consistency among the replicas managed by the set of agents. Specifically, we have used the δ-CRDT (Almeida et al., 2018). The third layer provides a user-level API to facilitate the coding of

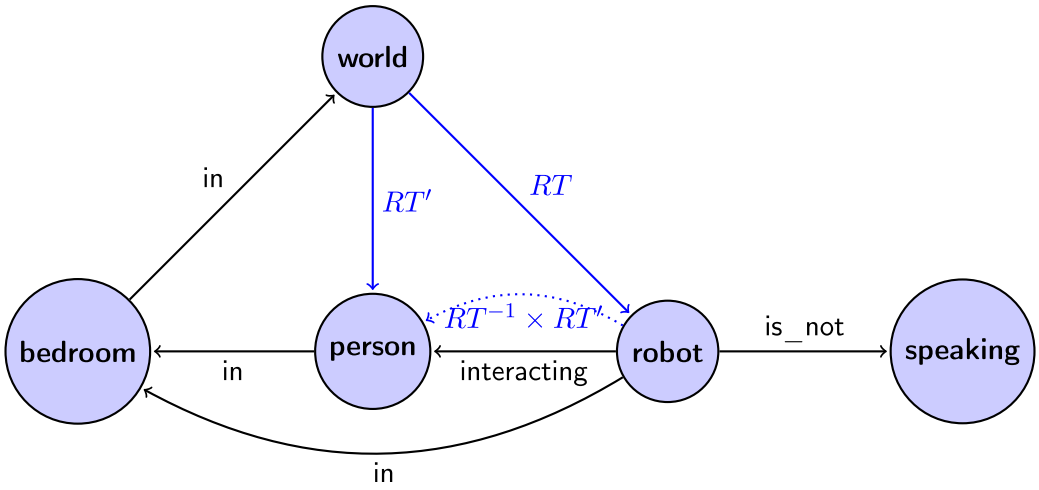


Fig. 4. Unified representation as a multi-labelled directed graph. Edges labelled as interacting and is _not denote logic predicates between vertices and they belong to Γ_s. Edges starting at **world** and ending at **person** and **robot** are geometric and they encodes a rigid transformation (RT' and RT respectively) between them. Geometric transformations can be chained or inverted to compute changes in coordinate systems.

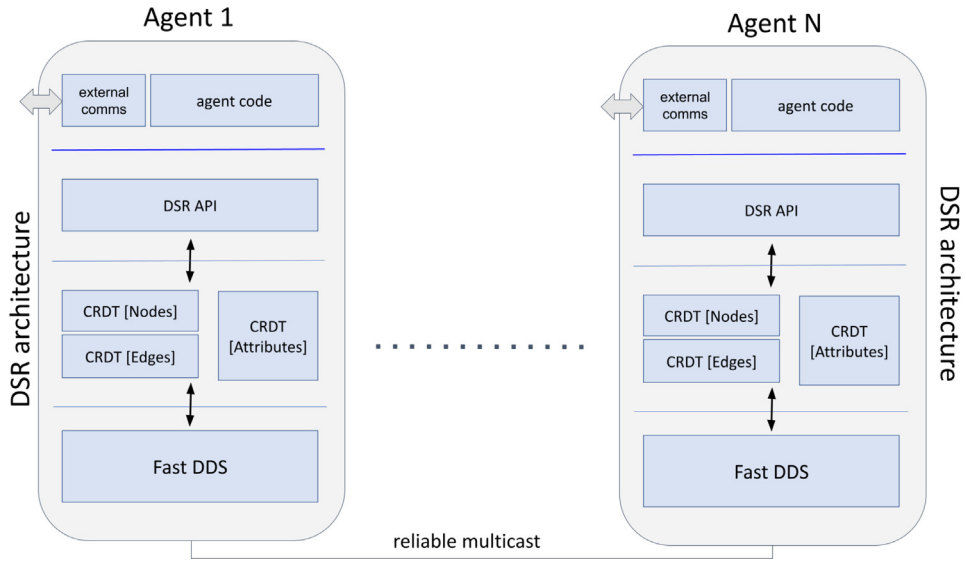


Fig. 5. Layered design of the working memory DSR.

new agents, giving them transparent access to the graph. Based on this design, the working memory does not exist as a single instance of a data structure accessed by the agents but as a set of local copies owned by them that interchange messages every time a change is locally made to one of the copies. This idea has several implications. First, access to the local copy of the graph is as fast as protected access to memory. Second, modifying the local copy, i.e., write access, implies the publishing through multicast of a topic containing only the part of the state that has changed, the so-called δ-mutations (Almeida et al., 2018). The logic underlying the CRDT included in our library takes care of the publishing and subscription to this topic and the local modifications to the graph that restore the synchrony among all copies. Third, reliable multicast provides a very efficient way of propagating messages among agents if all are hosted on the same machine.[5] Furthermore, increasing the number of agents does not increase the time required to propagate each message (Bustos et al., 2021). Finally, in stark contrast to a centralised server approach, our distributed design allows much faster access to data at the expense of the less strict synchronisation guarantee of eventual consistency. This condition implies that if all agents stop writing on the graph, all copies will come to a common state after a finite amount of time.

As mentioned above, eventual consistency is achieved using δ-CRDTs (Almeida et al., 2018). δ-CRDTs are a variant of state-based CRDTs in which, after executing a certain operation locally on the replica's state, the complete new version of the state is not sent to the rest of the replicas. Instead, only incremental states (i.e. deltas) are sent so that changes to the local state are incorporated in the rest of the replicas by joining the received deltas with their own states. Deltas are generated by so-called δ-mutators. A δ-mutator is defined as a function m^{δ} that, taking a given state X as input, returns a δ-mutation. Both the input and output state must belong to the join-semilattice space S of DSR graphs. With this restriction, only six types of δ-mutations are possible, which are presented, using a very simple DSR example, in Fig. 6.

Fig. 6 shows these δ-mutations as a transformation sequence of an input graph. The first of the mutations involves adding a new vertex

[5] The option of distributing the working memory among different computers is possible, but not used since the benefits of high-speed memory access would be lost and there exist other means to bring information to the DSR from distant machines using, for instance, point to point connections.

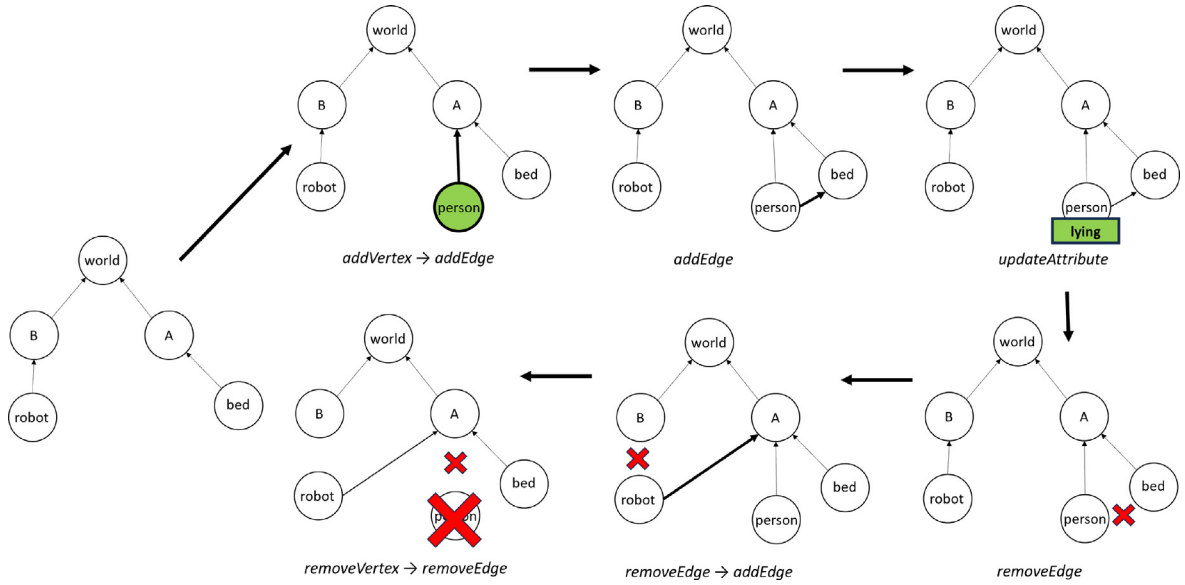


Fig. 6. The six types of δ-mutations that can be applied to a DSR graph.

(e.g. a **person** entering room **A**). The $addVertex$ operation will always force the addition of an edge, because in the DSR every vertex **u** (except the reference origin, marked as **world**) is connected to a vertex **v** by an edge $(\mathbf{u}, \mathbf{v})$. This edge $(\mathbf{u}, \mathbf{v})$ cannot exist without **u** in the state. However, as shown in the second δ-mutation, it is possible to add an edge between two vertices that already exist in the DSR. In addition to vertices and edges, a third relevant set present in the DSR network is the attributes. These cannot be added or removed as they are inherent to each vertex or edge added to the graph. Therefore, they can only be updated. In fact, these $updateAttribute$ changes, which do not modify the structure of the DSR network, will typically be the most frequent δ-mutations.

The fourth mutation is a $removeEdge(\mathbf{u}, \mathbf{v})$ that allows the associated vertices, **u** and **v**, to remain connected by a path of edges to the **world** vertex. This is not the case for any $removeEdge$. Thus, the fifth mutation is a $removeEdge(\mathbf{u}, \mathbf{v})$ which must be accompanied, in the same mutation, by an $addEdge(\mathbf{u}, \mathbf{w})$ (where **w** may or may not be distinct from **v**). The reason is that the vertex **u** (the **robot** in the example) cannot be unconnected (through a path) to the vertex **world**. The last possible δ-mutation that can be received is a $removeVertex$ which, for the same reasons as the $addVertex$ δ-mutation, has to come together with a $removeEdge$ (as there cannot be an isolated edge in the graph).

From a practical point of view, the graph elements (vertices, edges or attributes) are organised as multi-value registers, implemented using dot-based causality tracking (optimised δ-CRDT multivalued records (Almeida et al., 2018)). Our implementation has been built on the basis of the source code provided by Carlos Baquero, Paulo Almeida and Ali Shoker.[6] [The software implementation of the DSR is publicly available in a github repository.[7]

As discussed, the present scheme for managing a distributed runtime knowledge model is the core of the CORTEX software architecture. Each software agent will be able to work on a replica of the model, with the assurance that it will eventually be consistent. In the next Section 4, we will study both the specific CORTEX instantiation used to provide software support to the designed AAL environment and the environment and hardware used in the deployment that will be used to carry out the proposed use cases.

[6] https://github.com/CBaquero/delta-enabled-crdts

[7] https://github.com/grupo-avispa/cortex]

4. IoRT instantiation using the CORTEX architecture

Fig. 7 provides an overview of the CORTEX architecture instantiated in our AAL ecosystem. In this case, there is an IoT environment in which sensors are deployed, but also an actuator, an autonomous robot with navigation and user interaction capabilities, and a VC, in this case equipped with a self-adaptation system to the context, which allows it to modify the behaviour of the entire system. These behaviours are encoded in Behaviour Trees designed at design time. Obviously, the whole framework shown in the figure could be modified if necessary by adding new software agents or connecting other memories, such as long-term spatial memory and episodic, semantic or procedural memories that provide additional functionalities to the architecture.

Agents in the CORTEX architecture are linked with the DSR library to endow them with direct access to the distributed working memory through a simple C++ or Python API. Thus, each agent contributes knowledge to the DSR and makes it available to the other agents. As aforementioned, they provide a priori knowledge to the DSR in an initial step. In addition to metric or topological maps, the position of the charging station, to which the robot will autonomously go when its battery level recommends it, and the position of the sensors deployed in the IoRT environment are also known a priori. The Object agent can detect the objects (**table**, **chair**, **bed**...) from the set used to train a YOLO DNN network. Their positions in the environment are obtained using the depth plan in the RGBD camera, and their identity is tracked in time. The Person agent allows tracking people moving in rooms, as well as determining whether they are standing, sitting, or lying down. Both the Person agent and the Object agent are fed with images provided by azimuthal cameras. The positions of these cameras are known a priori. Other issues are also defined at design time. For instance, the Planner agent encodes the action in manually defined Behaviour Trees. Which of them is executed is a decision made by the Adaptation agent, depending on the evolution of the DSR itself.

In addition to the core level of connectivity with the DSR, agents can easily include other protocols and middleware to communicate with other processes that can be located anywhere. This is the basis of the technology presented in this paper. Some of these hybrid agents connect with other robotics frameworks such as ROS 2, RoboComp and Mira. Details of the implementation of these software agents are presented below, focusing on how communications are handled or on the functionalities they could provide. The framework illustrated in Fig. 7 has been instantiated in a small flat. Specific details of the hardware (robot, sensors) used in the specific deployment of the IoRT environment are also provided.

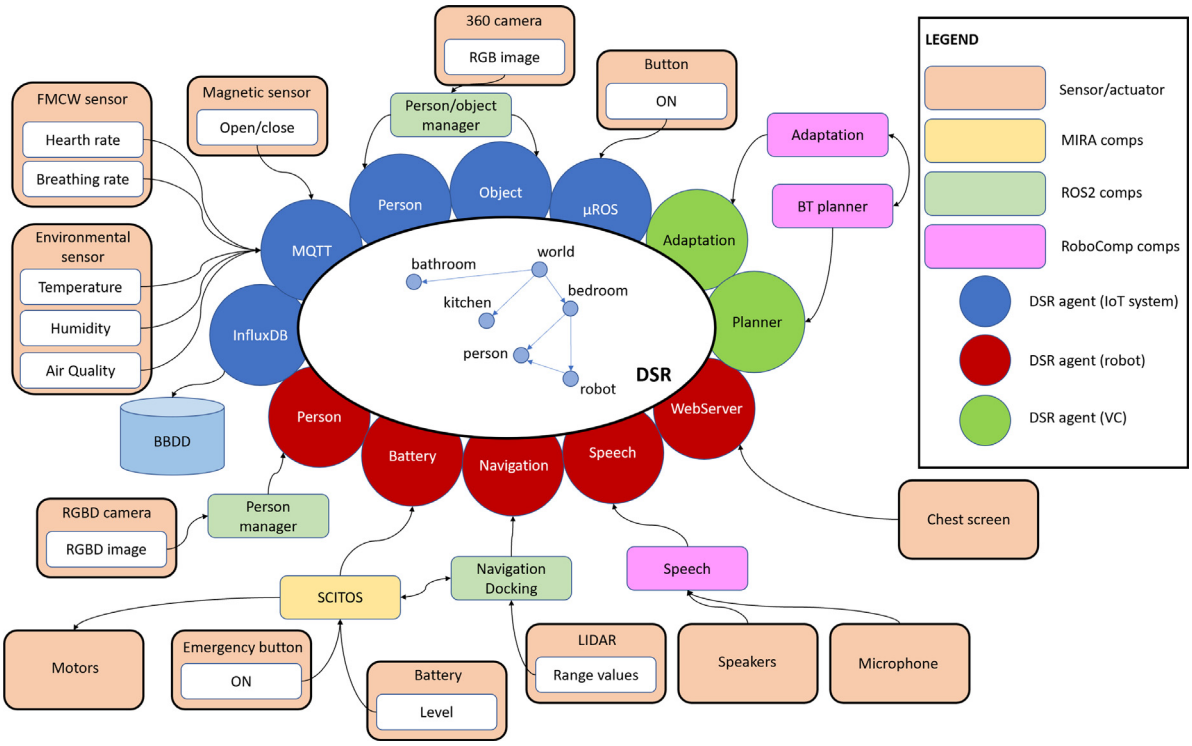


Fig. 7. The CORTEX architecture.

4.1. The IoT system

The AAL environment has been deployed in a small flat, with only three rooms: bedroom, dining room-kitchen, and bathroom. Fig. 8 illustrates the layout of the flat and the devices included in the IoT system. An Intel NUC i5 Mini PC is used to provide the computational resources needed to process the images from the cameras and also to store the software agents managing the DSR. Specifically, the devices in Fig. 8 are:

- **The 360° zenithal cameras**. To monitor the position of the person(s) and other objects in the two main rooms of the small flat (kitchen/dining_room and bedroom), two azimuthal cameras have been placed on the ceilings of these rooms. Specifically, the cameras used are the Panoramic Indoor Fisheye Camera with 6MP SHD from ReoLink. Fig. 9(top) shows what this camera looks like. Both cameras have been connected to a router, which allows captured images to be made available to the software stack in charge of processing them.
- **FMCW sensor for heart/breathing rates estimation**. In order to monitor the person's state of health, a frequency-modulated continuous wave (FMCW) radar sensor will be used to measure heart rate and respiratory rate data. The operation of this device is based on the emission of a continuous wave whose frequency is modulated in time. This modulation process makes it possible to establish a time reference. By receiving the echo of the emitted signal, the device can accurately estimate the movement of objects in front of the sensor. By working at very high frequencies (60 GHz), the device can detect very subtle movements, such as those associated with a beating heart. The sensor can therefore measure distance and radial velocity simultaneously. It is very accurate, non-invasive, and does not need to be precisely aimed at a specific point on the person. As Fig. 8 shows, two of these sensors were deployed in the small flat, one under the bed in the bedroom and one behind the back of the sofa in the dining_room. Fig. 10 shows the appearance of the device designed to house the FMCW sensor. Specifically, the sensor used is the MR60BHA1 from Seeed Studio. This sensor is connected to an ESP32C3 microcontroller, which includes the transceiver to connect via WiFi (MQTT) to the software agent that acts as a gateway to the DSR.
- **Environment, PIR and magnetic sensors**. To measure the most relevant environmental parameters (temperature, humidity and air quality), a board has been designed that uses the BME680 as a measuring element. The BM680 integrates gas, pressure, humidity and temperature sensors. The gas sensor can be used to assess indoor air quality, as it can detect a wide range of gases like volatile organic compounds (VOC). This device is connected via I2C to an ESP32-C3 microcontroller, which has a WiFi connection and implements MQTT to send the data to the software agent in charge of updating the DSR. The environment monitoring is complemented by a PIR sensor HC-SR501 located in the bathroom and magnetic sensors for monitoring the status (open/closed) of doors and windows. These devices are also connected, using an ESP32-C3 microcontroller, to the MQTT agent.
- **The button panel controller**. During the evaluation of this proposal, and as a proposal of the focus groups organised with care experts and end-users (Iglesias et al., 2024; Jerez et al., 2024), the IoT environment has been equipped with an actuator, with large buttons of different colours, which allows the elderly person to interact with the screens present either in the living room or on the chest of the robot. The aim of the button panel controller is to support the touchscreen, addressing scenarios where prolonged pressure on the screen could be uncomfortable and where users have limited mobility. Specifically, the controller consists of three mechanical buttons, each with an LED indicating when it is ready to be pressed (Fig. 11). These buttons are connected to an ESP32-WROOM32 microcontroller via the board's GPIO pins. This microcontroller communicates with the DSR via the μROS agent. Each button is mapped to a specific screen function, making the interaction more intuitive and user-friendly without the need for prior explanation. Our tests have shown that this combined interface significantly improved the user experience by allowing remote control of on-screen tasks from a more natural position.

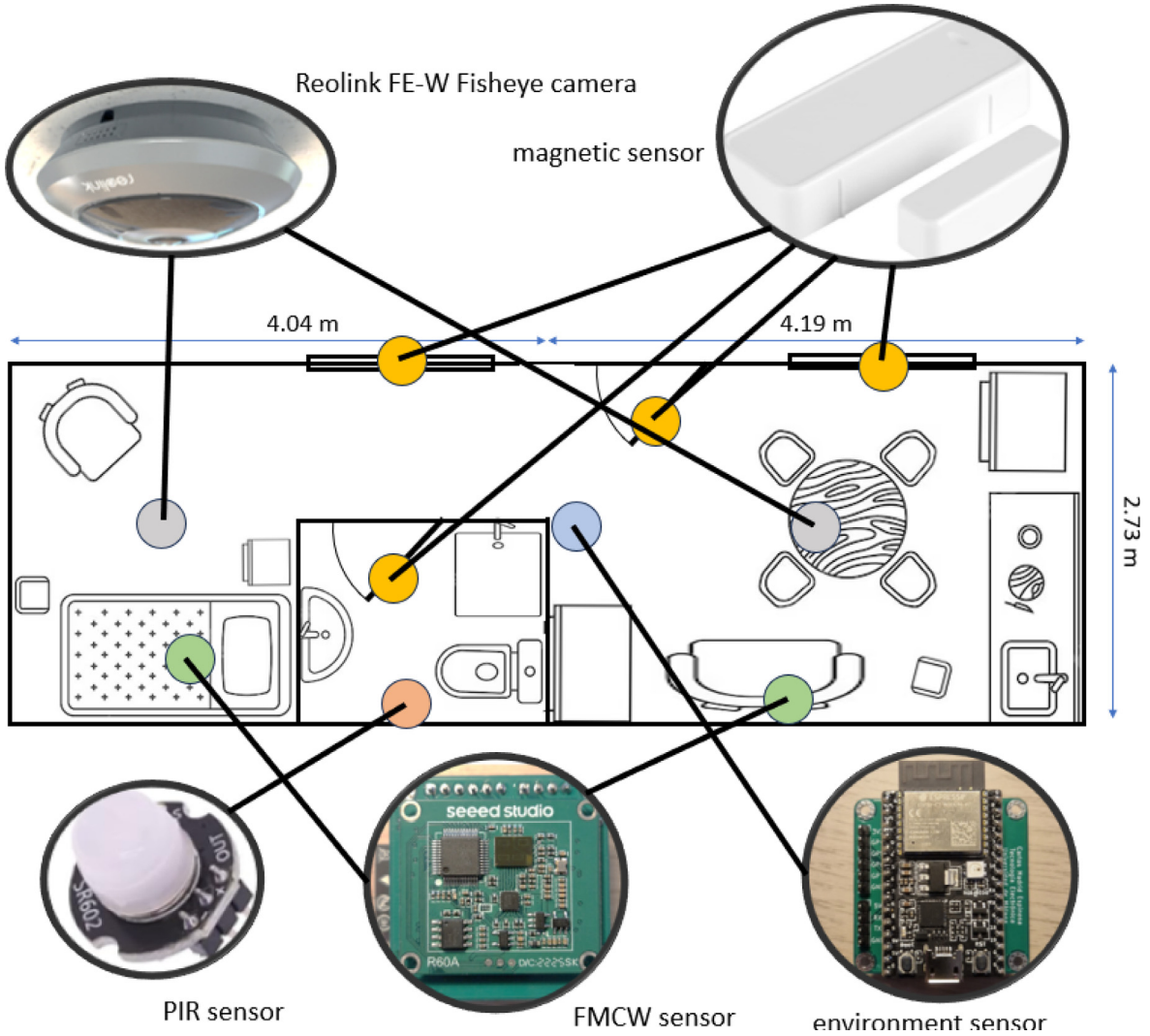


Fig. 8. Schematic layout of the small flat showing the distribution of sensors.

Fig. 9. (Top) The 360° Panoramic Indoor Fisheye Camera with 6MP SHD from ReoLink; and (Bottom) a rectified capture of the bedroom and the associated depth map.

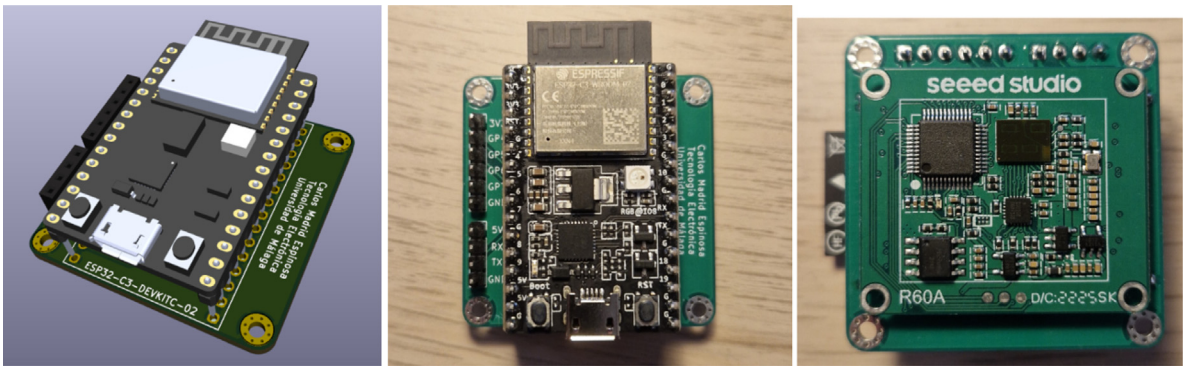


Fig. 10. Device mounting the FMCW 60 GHz sensor (MR60BHA1) from Seeed Studio and the ESP32C3 microcontroller: (Left) 3D model generated by Kicad 8.0; (Middle) Top layer showing the ESP32C3; and (Right) Bottom layer showing the MR60BHA1 sensor.

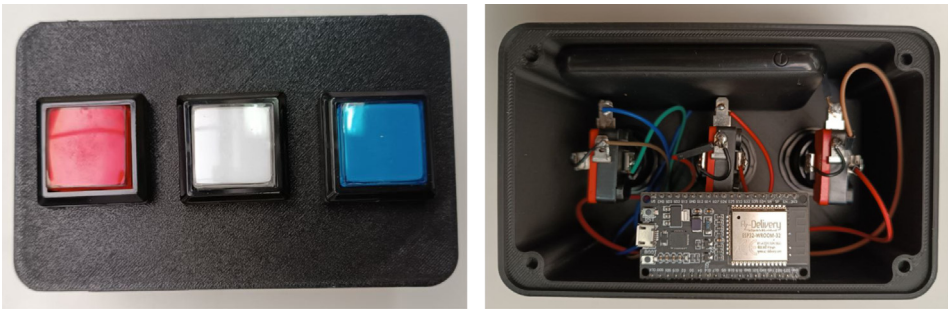


Fig. 11. (Left) The outside of button panel; and (right) the inside of the button panel.

As shown in Fig. 7, the interface of this set of devices with the DSR is based on five agents. The InfluxDB agent provides an interface between the DSR and an InfluxDB database.[8] [This database allows for solving the problem of the instantaneousness of the DSR (it would only have the last captured data) and to keep a history that helps in the decision-making of a caregiver or medical professional.] For instance, the heart and respiratory rate data captured while the person was lying in bed are stored in this database. The remaining software agents are described below.

4.1.1. The MQTT agent

The MQTT agent is a bidirectional interface used by many of the sensors deployed in the IoT system. These sensors are controlled by microcontrollers or single-board microprocessors. The only requirement is a WiFi transceiver. This agent receives data using the MQTT messaging protocol[9] and updates them in the DSR. The agent also sends relevant information from the DSR to the microcontroller or single board processor to modify its operation. The agent has been developed in C/C++ and has the following two parts:

- Connection with the DSR: A shared pointer is employed to connect this agent to the DSR. The agent also reacts to the following changes in the DSR, via signals and callbacks: (1) creation of a new vertex, (2) modification of a vertex attribute, (3) creation of a new edge, (4) deletion of a vertex, (5) deletion of an edge.
- MQTT Subscriber and Publisher: This protocol is implemented using the PahoC++ library from the Eclipse Foundation.[10] It connects to the Mosquitto broker as a client that subscribes to different topics to receive data and that publishes relevant changes in the DSR to notify external components.

The implementation of these agents allows full integration in CORTEX.[11] In order to better describe the functionality of this MQTT agent, Fig. 12 shows how a particular sensor is informed to start/stop measuring. The sensor in this example is the FMCW sensor, which is placed under the bed for estimating heart and respiratory rates. A zenithal, 360-degree camera is also deployed in the bedroom. It captures the complete room and is connected to the DSR by a ROS 2 agent (the Person agent). Initially, the person is in the bedroom (Fig. 12a). For the sake of simplicity, the DSR shows that only the **person**, the **bed**, the **360 camera**, and the **FMCW** radar sensor are in the **bedroom**. When the zenithal camera detects that the person is lying down, and that the person's position coincides with that of the bed, the Person agent notes in its replica of the DSR that the **person** is _in **bed** (Fig. 12b). As an attribute of the person, lying is also noted. Both events are encoded as δ-mutations and sent to the rest of the replicas. When the MQTT agent detects this situation, it publishes the message 'OnSensor' through the MQTT Control topic so that the microcontroller that manages the radar sensor activates it and starts taking measurements. The event is encoded as an attribute update in its replica of the DSR and also managed as a δ-mutation. The average of these measurements is sent in the opposite direction, i.e. through an MQTT message to the agent, which finally updates the attributes 'heart rate' and 'breath rate' of the person vertex with the values received (Fig. 12c). Updates are managed as δ-mutations. Finally, when the person gets out of bed, the Person agent updates the DSR (by removing an edge and updating an attribute vertex). Both δ-mutations are sent to the rest of the replicas. When the replica of the DSR in the MQTT agent is updated, this agent updates the DSR and publishes the message 'OffSensor', so that the measurements are finished (Fig. 12d). This avoids erroneous measurements that could cause confusion and false alarms. The example shows that all mutations incorporate the information necessary to be correctly added in the rest of the DSR replicates.

[8] https://www.influxdata.com/
[9] https://mqtt.org/
[10] https://eclipse.dev/paho/files/mqttdoc/MQTTAsync/html/index.html
[11] https://github.com/grupo-avispa/dsr_aal

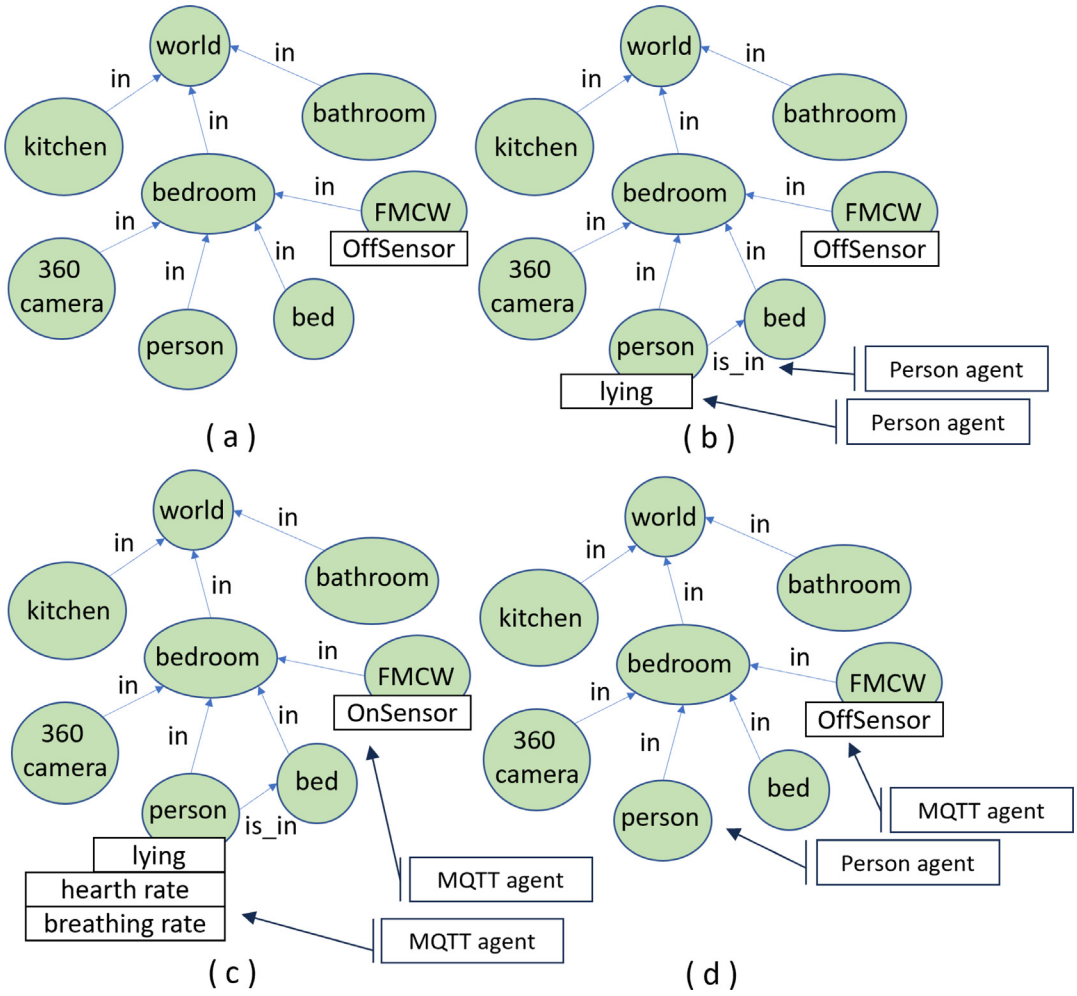


Fig. 12. Partial view of the DSR showing how it evolves when a person lies in bed (see text for details).

4.1.2. The person and object agents

To detect the objects in the rooms and also the presence of people, we have trained a YOLOv11 (large) model on a custom dataset consisting of almost a thousand images. Specifically, our 360 cameras detect people, their position (sitting, standing, or lying down), and a set of objects (the ones presented in the small flat). Before conducting the detection procedure, 360 images must be rectified (Fig. 9).

The YOLO solution encloses each detected item in a bi-dimensional (2D) bounding-box. However, in our representation, objects and people are managed as three-dimensional (3D) objects. To estimate the 3D position of the detected entities, we compute the depth using the Depth Anything model (Yang et al., 2024). Detections and depth are merged, and the information about people is extracted and updated in the DSR. For addressing this pipeline, we have created a ROS 2 wrapper for the YOLO models, composed by a node with an image subscriber (original RGB image), a Detection2DArray message[12] publisher, and a marked detections image publisher (bounding boxes drawn on the original RGB image). The bounding boxes associated to the detections are matched with a depth image obtained using our ROS 2 wrapper for Depth Anything model.[13] The 2D-to-3D conversion is developed with a synchronised subscriber (for managing 2D detections and depth image at the same time) and a Detection3DArray message publisher.[14] Information about people is extracted from this Detection3DArray message and a set of attributes are selected to be published in a custom person message. It can be noticed that all nodes of this perception pipeline have been converted to ROS 2 composable nodes and encapsulated under a single container. This allows for zero-copy communication, which saves computational resources and reduces latency (Macenski et al., 2023). [Source code as well as a more detailed description of this ROS 2 perception pipeline is stored in a publicly available repository.[15]

4.1.3. The micro-ROS agent

The idea of micro-ROS.[16] is to integrate microcontrollers into the framework by deploying a micro-ROS agent on the NUC in ROS 2 and a micro-ROS client on the microcontroller. The API on the client is based on the standard ROS 2 Client Support Library (rcl) and a set of convenience extensions and functions (rclc). This combination is optimised for microcontrollers, providing the basic concepts of ROS 2, such as publish/subscribe, node graph, lifecycle, etc. The main targets of micro-ROS are mid-range 32-bits microcontroller families. Usually, the minimum requirements for running micro-ROS in an embedded platform are memory constraints. Specifically, in the deployment described in this paper, the ESP32-WROOM32 microcontroller was used, combining micro-ROS and FreeRTOS functionalities. [Source code is available in a public github repository[17]

[12] https://github.com/ros-perception/vision_msgs/blob/ros2/vision_msgs/msg/Detection2DArray.msg

[13] https://github.com/grupo-avispa/depth_anything_v2_ros2

[14] https://github.com/ros-perception/vision_msgs/blob/ros2/vision_msgs/msg/Detection3DArray.msg

[15] https://github.com/grupo-avispa/ros_perception_pipeline]

[16] https://micro.ros.org/

Fig. 13. The Morphia robot navigating in the small apartment.

4.2. The autonomous robot Morphia

The assistance robot used in our AAL is a Morphia from MetraLabs GmbH. Developed in the course of the joint research project MORPHIA,[18] it is the result of a collaboration among the Technical University of Ilmenau, CIBEK Technology und Trading GmbH, SIBIS Institut für Sozial- und Technikforschung GmbH, the University of Osnabrück, AWO Alten- Jugend- und Sozialhilfe (AJS) GmbH, and YOUSE GmbH. Morphia is not a humanoid-looking robot, but it is designed to solve certain tasks that can help an elderly person living alone at home. For example, it has a tray that allows it to accompany a person using a walker and bring them a plate of food or a glass of drink. [It is also equipped with a tablet, which] is large enough so that even elderly people can interact with it without any problems.

Fig. 13 provides some snapshots of the Morphia robot in the small apartment where the IoRT has been deployed. The Morphia robot is built on the TORY differential base, also from Metralabs. Navigation uses a SICK s300 range laser scanner and an Intel RealSense D435i RGB-D camera. The detection of possible impacts while navigating is solved with a circular bumper. The perception of the environment is built on the basis of three Valeo 2MP cameras and a Microsoft Azure Kinect RGB camera. The latter camera is also used for human interaction, for which a tablet and speakers are included. Image processing is carried out using an NVIDIA Jetson Orin AGX. The entire system is controlled by an Intel NUC i7.

As Fig. 7 shows, there are a multitude of software agents running on the robot that are connected to the DSR to control and monitor the robot. These agents run on ROS 2, RoboComp, or MIRA.

4.2.1. ROS 2 agents

Most of the agents running on the robot have been developed using ROS 2. For its design, a package has been created that facilitates the generation of these agents, characterised by their integration into CORTEX by using a local copy of the DSR. Specifically, the software agents running on the Morphia robot are:

- Docking Agent: Responsible for carrying out the docking of the robot at the charging station.
- Navigation Agent: Facilitates the execution of navigation commands from the DSR.
- Person Agent: Publishes data on detected people within the DSR.
- Semantic Goals Agent: Invoke semantic goal actions in the DSR to retrieve random goals from a specified region of interest.
- TF Agent: Publishes the transformation tree as vertices in the DSR, providing a visual representation of transformations across different reference frames in the CORTEX environment.
- Battery Agent: Publishes all information related to the battery (percentage, voltage, temperature, etc.) as attributes belonging to the **robot** vertex of the DSR. This agent receives this data by subscribing to MIRA software topics.

A brief description and the implementation of these software agents is available in a public GitHub repository.[19]

4.2.2. The speech agent

This agent manages the software that allows the robot to communicate by voice with end users. It takes the text messages that are written down in the DSR and converts them into voice and plays them back using Pico TTS.[20] This system has two distinct parts:

- Text-to-Audio: The system generates speech audio from text using `pico2wave`. This utility is a lightweight tool designed to generate speech efficiently, especially on systems with limited resources. As part of the SVOX Pico TTS system, it supports multiple languages and converts text input into audio files in an uncompressed WAV format. Its strengths are simplicity and speed.
- Play audio: To process and play audio, the `play` command is used. This is a command-line utility from the Sox (Sound eXchange) suite, designed to play and process audio files. It supports a wide variety of audio file formats and provides numerous audio effects, making it a versatile tool for sound playback and manipulation.

A typical example of execution that combines both parts is:

```
pico2wave -l es-ES -w ''<filename>'' ''<text>'' && play -qV2 ''<filename>'' treble 18 gain -l <volume>
```

[17] https://github.com/grupo-avispa/infrared_uros]
[18] https://www.morphia-projekt.de/
[19] https://github.com/grupo-avispa/dsr_ros/tree/main/dsr_agents
[20] https://github.com/ihuguet/picotts

This command uses `pico2wave` to synthesise speech from the given text and save it to a WAV file, and then plays the audio using `play`, applying treble enhancement and volume adjustment. It can be customised with different languages and parameters as needed. The Speech agent is publicly available in a github repository.[21]

4.2.3. The WebServer agent

The WebServer agent is responsible for the interconnection between the graphical user interfaces (GUI) that appear on the robot's screen and the DSR. It should be noted that the control of these interfaces is carried out either through the touch panel of the screen itself or through the button panel described above. Therefore, this agent allows both the touch screen (external or attached to the robot) and the external button panel to communicate with the DSR.

The agent is based on the WebSockets communication protocol. This protocol provides full-duplex (bidirectional) communication channels over a single TCP connection. It is designed for bidirectional, real-time data transfer between a client (usually a web browser) and a server, allowing both parties to send and receive messages independently and simultaneously.[22] The programming of this agent has been done in C/C++[23] and is used in two different ways, listening and writing:

- Listening from DSR: To make the GUI as useful as possible (showing, for example, the status of the task the robot is executing), certain changes in the DSR cause changes in the graphical interface. This agent is responsible for communicating this change to the display (either the one carried by the robot or an external one), so that the interface can update subtitles or images, or completely change the interface to a more appropriate one.
- Writing to the DSR: Using the touch buttons on the GUI (or wireless buttons on the button panel), the user can answer certain questions or express preferences or needs. This agent converts these suggestions into changes in the DSR. These changes can range from updates to the person's attributes (if the robot asks how he/she is) to changes in the robot's behaviour (executing an exogenous use case when a button is pressed).

As an example of how this agent works, we can describe how the robot allows a user to select their lunch menu. While the robot is interacting with the person, when the planner proposes that this is the moment to ask the user what he/she wants to have for lunch, it does so by setting the show option as the action to be executed by the robot, and indicating the GUI to be shown as an attribute. The show option wakes up the WebServer agent and, in Listening mode, manages the control of what to show to the person (in this case the menu options). In Writing mode, the agent waits for the user's response, which can come from the touch screen or the button panel. Finally, when the user chooses the desired option, the client sends a message to the agent with the options, and the agent updates the DSR with an attribute of the person who saves what he/she want for lunch. Both modes, Listening and Writing, can be active simultaneously.

4.3. The virtual caregiver

The data collected in the DSR allows all actors to have an estimate of the context external and internal to the robot. This information can cause a particular agent, reactive or deliberate, to act (Marfil et al., 2020) (e.g., by annotating a certain text, the Speech agent plays it as audio). In the deployment described in this article, the two agents responsible for determining, at a high level, the task to be solved by the AAL environment have been grouped as Virtual Caregiver. The first of these agents integrates a context-aware self-adaptation system, responsible for determining the behaviour of the system. The second agent is responsible for executing the selected behaviour. More details on both agents are provided below. Source code of this framework is publicly available.[24]

4.3.1. The adaptation agent

This agent is in charge of selecting the use case that should be active. To determine the optimal course of action for the remaining cases, we employ a method inspired by preference-based learning (Fürnkranz and Hüllermeier, 2003). This area of machine learning emphasises classifying items based on preference data, often gathered from domain experts. In our approach, we collect input on use case preferences from individuals involved in robot development as well as staff from retirement homes (Iglesias et al., 2024). This data is used to train multiple classifiers that assign scores to each use case. The scores are then utilised to generate a ranked list, prioritising use cases from the highest to the lowest score. This agent is implemented using the RoboComp framework.

In the use cases illustrated in Section 5, the adaptation agent first evaluates whether the robot has sufficient battery power to tackle the tasks. If the battery level is deemed adequate, it checks the use cases with the highest priority values. For example, in the current version of our system, these are *Person fall* and *Follow*. If neither of these cases is selected, preference learning is applied. Our preference learning algorithm takes users' information into account to select the best use case for them (the *Explore* task is the default). However, this use case may not be active due to the time of day or the current location of the robot. An example of the smart use case switching algorithm is shown in Fig. 14.

4.3.2. The planner agent

The planner agent is a software agent that loads and executes Behaviour Trees (BTs) to handle various use cases, selected dynamically by the Adaptation agent as previously explained. Built on BehaviorTree.CPP,[25] it enables modular, asynchronous, and reactive behaviour execution. One of the key advantages is its support for asynchronous execution, where actions are non-blocking by default. This makes it possible to run multiple tasks concurrently without unnecessary delays, improving real-time applications. The Behaviour trees are defined using a Domain-Specific Language (DSL) based on XML, which allows for runtime loading and modification. [To create, edit and visualise behaviour trees we use Groot as tool.[26] Also, some examples of XML files and how to upload them are shown in their official repository.[27]

5. Experimental results

The performance of the proposed architecture has been tested in the small living lab described in Section 4.1. Briefly, the use cases try to assess that the elements of the AAL ecosystem coordinate smoothly and that they generate the required action without relevant latencies for the execution of the action. The first of them is the attention to a possible fall of the user. This situation can be detected at any time by the static cameras fixed on the ceiling of the rooms, and is communicated to the robot so that it can talk to the person and act as predefined. The second is a more everyday case for people who move around the flat with a walker, for example. By having to walk with the walker, the person cannot move small objects around the flat. The robot, equipped with a small tray for this purpose, can be prompted by the user to help with this task. These use cases have been

[21] https://github.com/grupo-avispa/dsr_aal
[22] https://websockets.spec.whatwg.org/
[23] https://github.com/grupo-avispa/dsr_aal
[24] https://github.com/grupo-avispa/dsr_aal
[25] https://www.behaviortree.dev/
[26] https://www.behaviortree.dev/groot/]
[27] https://github.com/BehaviorTree/BehaviorTree.CPP

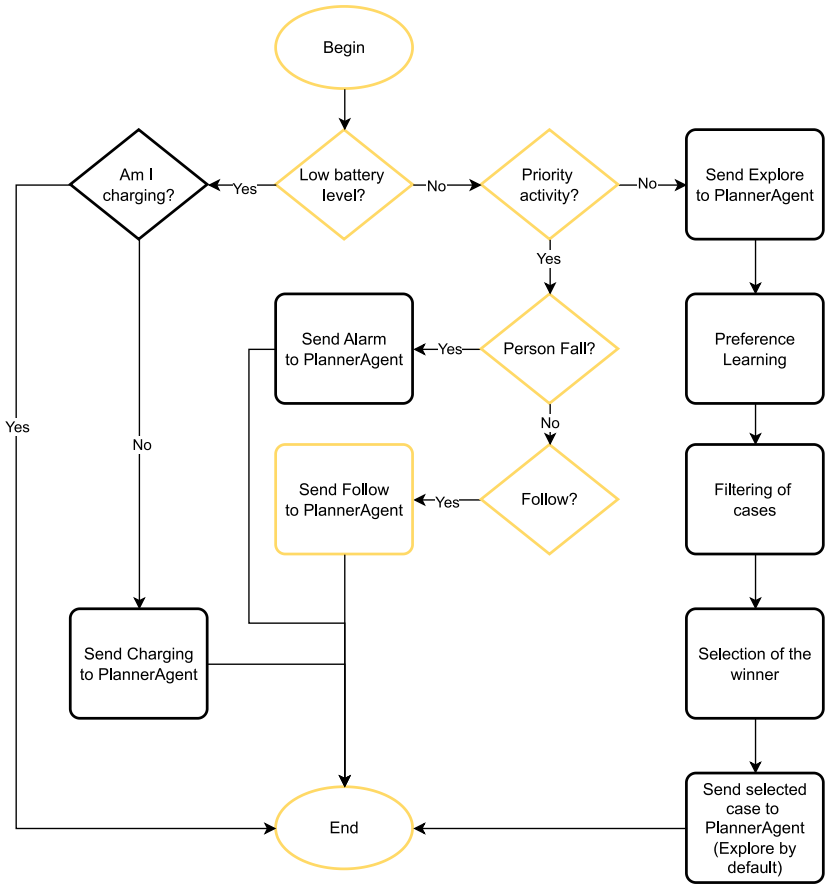


Fig. 14. Adaptation loop for smart use case switching.

co-created with representative end users in focus groups that included elderly people, caregivers, engineers and anthropologists (Iglesias et al., 2024; Jerez et al., 2024). Performed experiments focus on evaluating the feasibility and functional performance of the proposed software architecture in these use cases. An analysis of the user experience is beyond the scope of this paper, although the results discussed in Jerez et al. (2024) on a partial version of this architecture (involving only the robot and VC) provide a promising insight into the evaluation to be done. In addition to the qualitative evaluation that can be drawn from these two examples, the response times of the most critical software agents and, above all, the time it takes for local copies of the DSR to be fully synchronised after a δ-mutation have also been measured. These more quantitative results are reported in Section 5.3.

5.1. Assistance in the event of a possible fall

To show how the designed IoRT scenario works, different use cases have been implemented in collaboration with medical staff and caregivers from different facilities (Iglesias et al., 2024). Specifically, in this section, we are going to use one of them to show the IoRT system's ability to act quickly in case it detects a possible user fall. As mentioned above, a CNN has been trained to detect, using the images from the zenithal cameras, whether the person is standing, sitting or lying down. Using this knowledge, the VC system will ask the robot to stop any task in progress and approach the person it detects as lying down (as long as it is not in bed) to ask how they are and, if necessary, alert a caregiver to come and attend to the user.

Fig. 15 shows the state of the DSR at a given instant. Only information from the symbolic quiver is shown, from which certain attributes and objects (chairs, table, rocking chair, etc.) have been omitted for easier viewing. As shown by the person detections carried out on the image of the camera present in the kitchen-dining room, the person is in this room and remains standing. The robot is outside the flat, in the area marked on its metric map as north. The DSR shows the presence of sensors deployed in the IoRT environment, but no internal attributes are shown (e.g. the entrance door in the kitchen room is open).

At one point, the person falls to the ground (Fig. 16). The camera images capture that moment, and the person is tagged as lying. The Person agent also determines that he is not in bed. The information is annotated in its local copy of the DSR, and δ-mutations are sent to the rest of the agents. The change in the context forces the Adaptation agent to determine that the robot should try to talk to him (he could simply be looking for something on the floor). The task to be performed by the robot is changed to attending this specific alarm (**alarm_lying**). When the Planner agent updates its DSR and detects this change, it launches a different Behaviour Tree that manages how to act (Romero-Garcés et al., 2022). The first action in the plan is to approach the person. Thus, the robot starts to navigate. The planner agent informs the rest of the agents of the activities to be addressed by annotating the DSR in its local copy. When the updates are incorporated into the replicas in the rest of the agents, the behaviour is unfolded (Marfil et al., 2020). Obviously, this scheme forces the Planner to supervise the update of the DSR to capture when a subtask has finished and a new one must be triggered. Fig. 17 shows that the robot has arrived at the person's location (**robot** _with **person**). Both by voice and with a specific message on the tablet on its chest, the robot asks the user if he is well. If the person takes a few seconds to respond, the system sends a message and calls a caregiver, and if the caregiver demands it, can use the robot to see and try to talk to the user, as if setting up a video conference. All software agents are run locally, and changes to DSR copies are updated almost immediately (latency is in the order of milliseconds). From a functional point of view, the system behaves as if there is a single copy of the DSR.

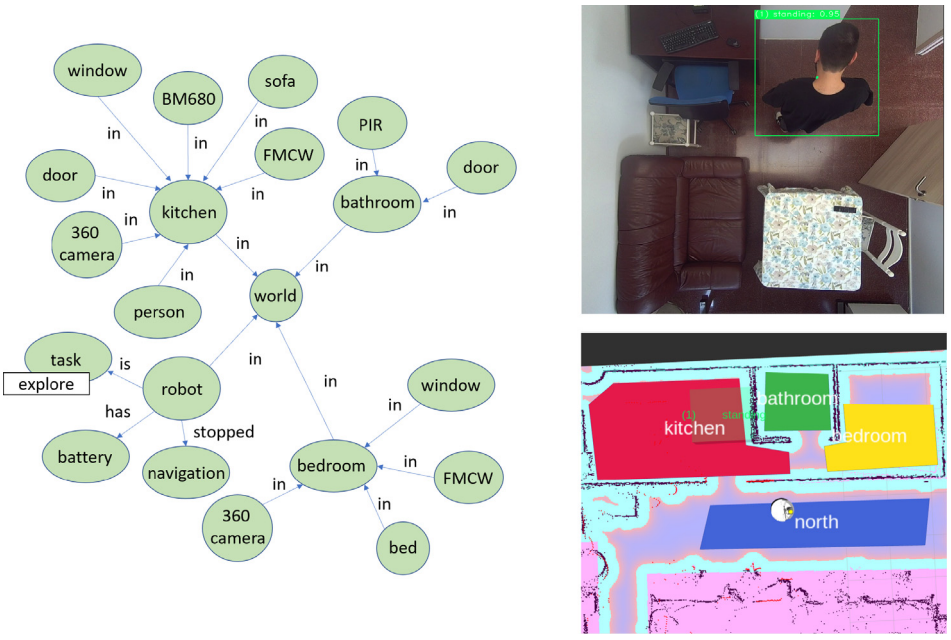


Fig. 15. (Left) A snapshot of the DSR running within the IoRT ecosystem (see text); and (right) person detections using the zenithal camera in the kitchen-dinning room, and metric map showing the robot and person positions, as well as the four marked areas.

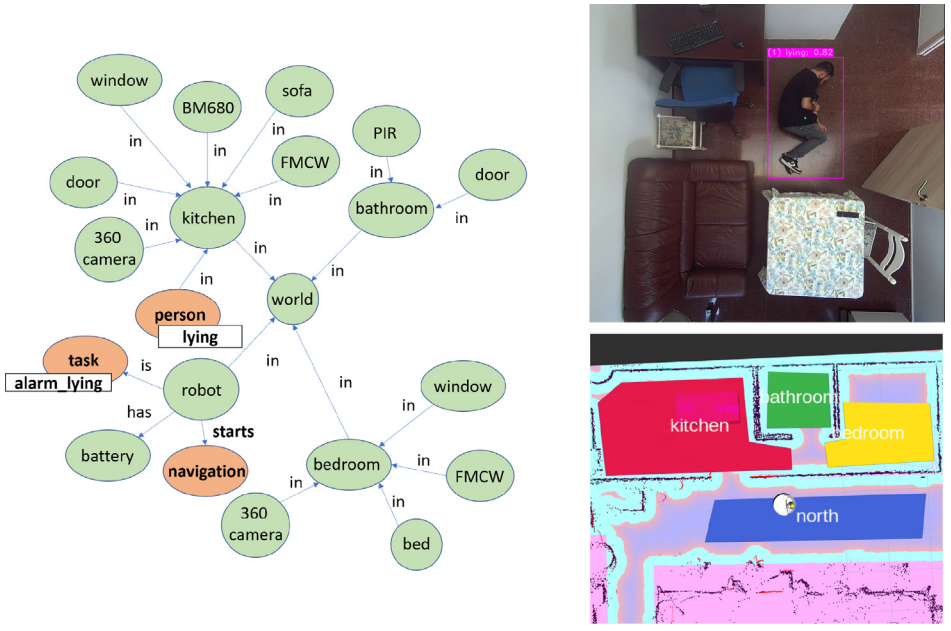


Fig. 16. (Left) The DSR is changed with respect to the situation in Fig. 15 when a person lying in the ground is detected; and (right) person detection, and metric map showing the new situation of the detected person.

This simple example shows how a common DSR is updated from its different replicas to orchestrate a given task. Sensors deployed in the environment allow continuous monitoring of the context while the robot provides a more natural and intuitive interaction interface for the user. The VC is able to change the task to reach according to the context. Although images are shown in the figures, it is important to note that sensitive data, such as images but also audio, are processed by the system locally, without being accessible externally. The system extracts from these information sources the knowledge that it injects semantically into the DSR as a continuous flow of δ-mutations.

In most of the environments in which this proposal has been deployed (be it this small flat or the Vitalia retirement home in Teatinos (Malaga, Spain)), the DSR comes to have a rather static representation (the rooms and objects hardly change), in which only the people show greater dynamism. This means that, as aforementioned in Section 3.2, the software agents mostly act on the DSR by changing attributes. The richness of these attributes has not been shown in the previous images. Fig. 18 illustrates the state of the DSR at a certain instant in time. The attributes on the edges, shown in previous figures, have been removed to simplify the image. These attributes are more complex than a single term. The figure shows part of the attributes of two of the vertices in the

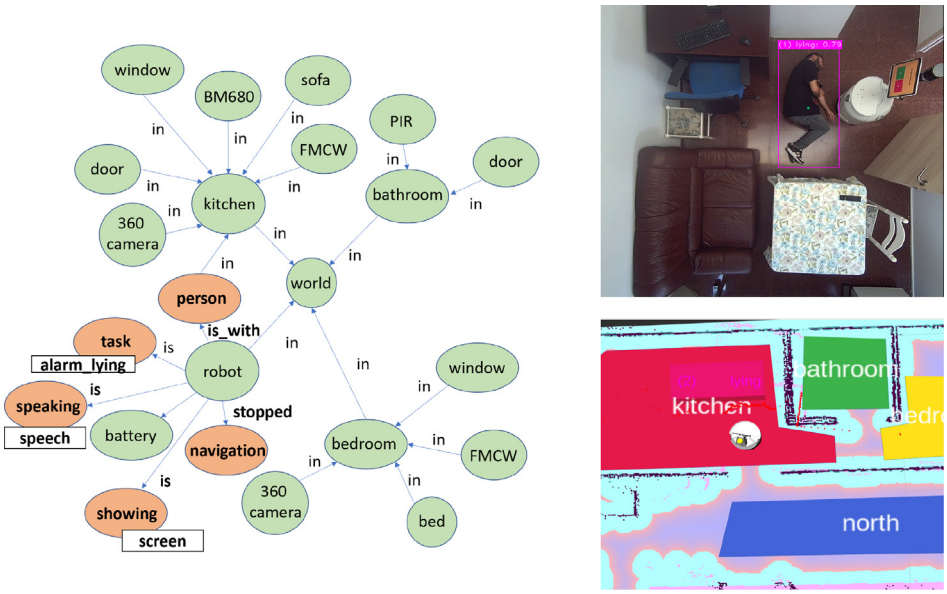


Fig. 17. (Left) The DSR shows that the robot is trying to interact with the person using voice and tablet; and (right) person detection, and metric map showing the new situation of the robot.

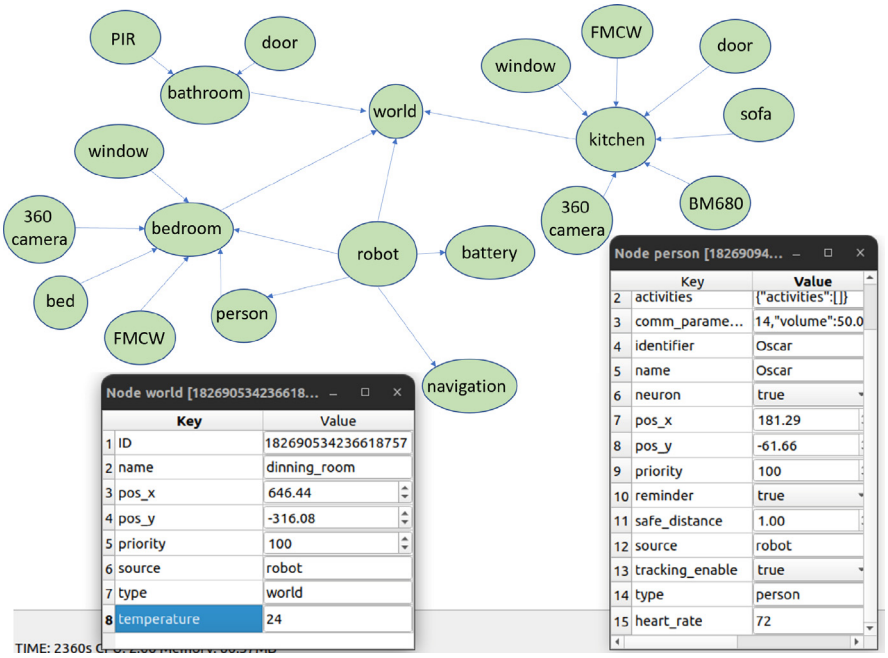


Fig. 18. A snapshot of the DSR running within the IoRT ecosystem.

DSR. Specifically, the person is identified as Oscar. While he was lying in bed, the FMCW sensor captured data on his heart and respiratory rates. In the list of attributes, you can see how the respiratory rate value (72 bpm) has been noted. On the other hand, the temperature of the dining_room (kitchen), for example, is 24 degrees. In general, agents will carry out continuous attribute updates, but few structural changes (i.e. inclusions or deletions of vertices and edges) will be made to the DSR network. These attribute updates also usually involve a single software agent (e.g., the robot's battery is updated by a single agent, its position on the map by another, etc.). Both factors facilitate the synchronisation of local copies of the DSR.

5.2. Assists the user in transporting small objects

A second use case is described in this section. In this case, the user, who needs a walker to move around, requires the help of the robot to carry certain objects in a different room in the flat. Fig. 19 shows the initial situation: the person is lying in bed and has, next to the bed,

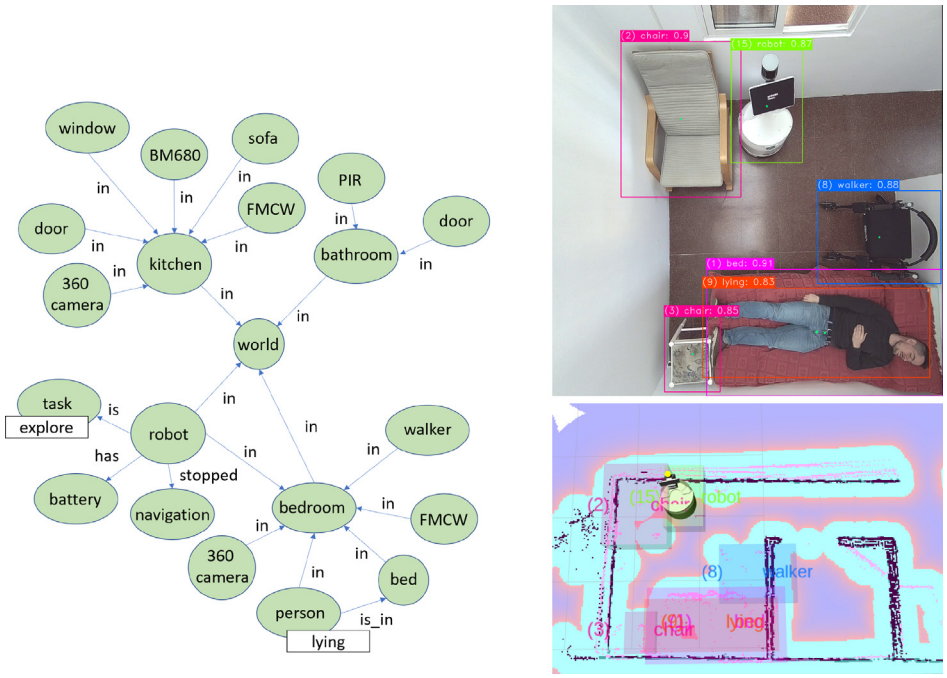


Fig. 19. (Left) The DSR shows that the person is lying on the bed. The robot is stopped and internally running the default task (explore); and (right) objects and person detection in bedroom (some of the detected objects are not shown in the DSR for clarity), and metric map used by the Navigation software agent.

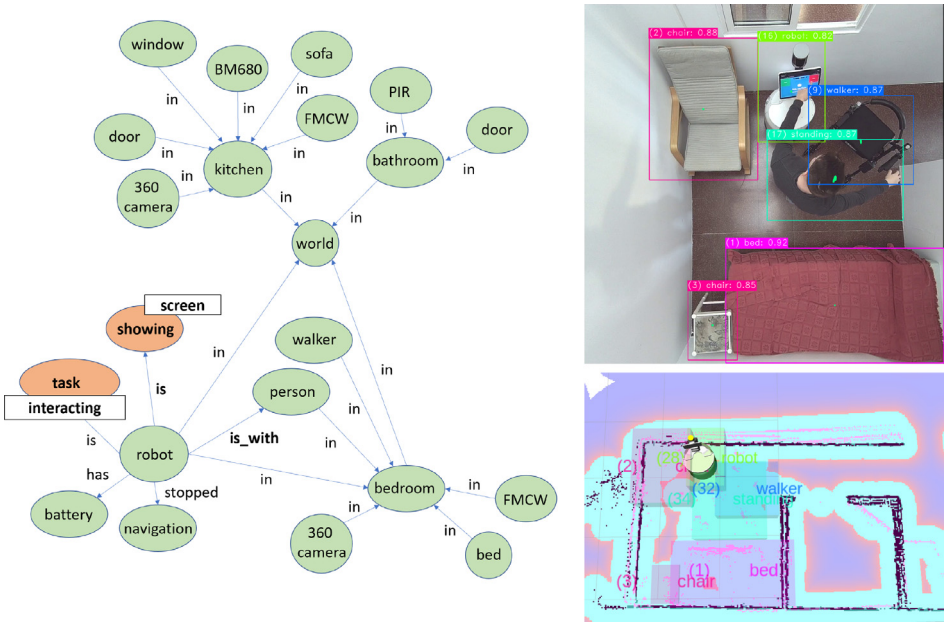


Fig. 20. (Left) The person uses the walker to approach the robot and ask it, using the touch screen, to follow him; and (right) objects and person detection in bedroom (some of the detected objects are not shown in the DSR for clarity), and metric map used by the Navigation software agent.

the walker. The robot remains in the same room (the default task is to explore). The person gets out of bed and, using the walker, approaches the robot and interacts with it using its touch screen. He asks the robot to accompany him as he is going to need its help (Fig. 20). The person interacts with the robot using the touch screen on its chest. The Webserver agent manages the interaction. The task that the robot runs changes to **interacting**. The person can ask the robot to accompany him.

In this case, the person moves, always with the walker, from the bedroom to the living room/kitchen (Fig. 21). When the person goes out of the bedroom, he is not immediately detected by the camera monitoring the living room. However, the vertex **person** is not removed from the DSR as the Person agent in the robot maintains that the **person** _with **robot**. However, the walker is removed. It is added again when person, robot and walker go into the living room. Person and robot are now linked to the living room. Contrary to the person, the robot is always linked to a space in the map (whether it is a room in the flat or an area outside the apartment). In the living room, the person approaches the central table and places a small object on the tray held by the Morphia robot. The robot follows the person to a

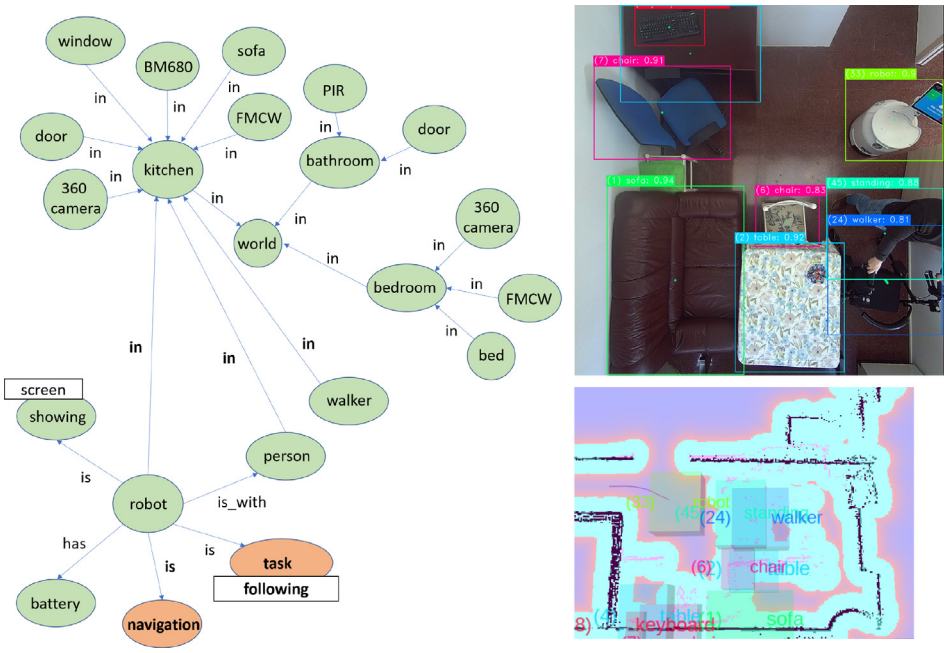


Fig. 21. (Left) The robot follows the person into the living room; and (right) objects and person detection in the living room (some of the detected objects are not shown in the DSR for clarity), and metric map used by the Navigation software agent.

second table, where the person sits down and can now take the object from the robot's tray.

5.3. Quantitative evaluation

Use case-based evaluation has been complemented by quantitative metrics that reflect system performance. In particular, the response times of the system have been measured. The most critical times are those associated with the detection of persons and objects, using the YOLO v11 network. Running on the Jetson AGX Orin 64 GB, information about these entities is updated every 143 ms (7 Hz).

After a software agent reports, by publishing a δ-mutation, a modification on the DSR structure, the representation takes, at most, 232 ms to be fully consistent in what is stored in all replicas.

Finally, with respect to resource utilisation, the aforementioned NVIDIA Jetson AGX Orin shows a GPU utilisation of 83.9% (816 MHz). The DSR and software agents are executed in a NUC7i7DNH processor with eight cores. All these cores are utilised at 50% approx. The NVIDIA Jetson consumes a power average of 26.0 W.

6. Discussion and further enhancements

The work currently being undertaken or expected to be undertaken in the short to medium term follows different lines of research. The following subsections list these future lines of work or improvement, grouped at the level of application, architecture or representation of knowledge and its management. Possible limitations of the proposed knowledge model scheme are also assessed.

6.1. Future work at application level

New use cases, designed in the framework of our collaboration with caregivers, medical professionals, and residents of the Vitalia residence in Teatinos (Malaga, Spain), are also continuously being evaluated. Robots are deployed in this residence, which may serve in the medium term as a space for evaluation of the complete architecture in a more complex environment. In fact, as described in some sections of this article, the system has been partially deployed in this residence for the last five years (in periods of time that are not always continuous). The analysis of the user experience during these deployments, in terms of acceptability, utility and accessibility, offer positive results that support the validity of the current proposal (Iglesias et al., 2024; Jerez et al., 2024). In this scenario, where more residents live, one of the problems is sometimes the identification of the user. For this purpose, we are working on a multisensory framework, to which we intend to add the FMCW radar sensor described above. Incorporated into the robot, there are previous works that show how to use this sensor for identification.

6.1.1. Data security

Protecting data security and user privacy is paramount, especially when dealing with sensitive or personal information. In our work, the only sensitive content corresponds to the images detected by the camera. In this case, the images are only processed locally on the machine and are deleted at every instant, so they are not saved at any time, guaranteeing the user's privacy. However, to ensure ethical considerations, we requested informed consent from all external subjects involved in the study (users and professionals). This informed consent was approved by the Provincial Ethical Committee of the Andalusian Public Healthcare system (Comité de Ética de la Investigación Provincial de Málaga). These protocols and ethical procedures have been successfully applied in previous evaluation projects involving a larger number of users (retirement home) and conducted with greater depth and scope of analysis (Jerez et al., 2024).

Finally, in order to extend the functionality of the research and the security of the people involved, the main methods and standards used to protect this information are discussed below.

- End-to-End Encryption: ensures that data is encrypted at the source and remains inaccessible to anyone except the intended recipient, even during transmission. This method provides a robust safeguard against eavesdropping or interception, making it ideal for sensitive communications. By implementing E2EE, the

research could emphasise protecting user data across potentially vulnerable channels.
- Compliance with data protections regulations: Compliance with regulations like GDPR and HIPAA ensures the lawful and ethical handling of sensitive data. These standards mandate practices such as obtaining user consent, anonymising personal information, and ensuring data security through encryption and access controls.

In conclusion, by implementing compliance with regulations such as GDPR and HIPAA, our research can ensure the safe and ethical handling of personal information. These frameworks provide clear, proven guidelines that are easy to adopt and robust enough to guarantee data security and user trust.

6.2. Enhancements at CORTEX architecture level

At the level of architecture, in the short term, new sensors or actuators are being incorporated into the IoRT ecosystem that will allow the robot, for example, to command the opening of a motorised door. However, in the medium term, the lines of work involve further research and innovation. All the knowledge that the system can store about the evolution of the context can enable the robot to explain to the person the reasons behind its behaviour or an unexpected event for the person. The integration of a theory of explainability into CORTEX will allow the robot to become a closer and more reliable tool for the people with whom it shares the environment.

Emerging technologies such as edge computing, digital twins, and 6G connectivity can significantly enhance the CORTEX architecture by addressing its computational, contextual, and communication needs. Edge computing can reduce the latency and improve the response times for the agents working with the CORTEX architecture. Also, this technology can maintain some functions locally even when the central system is temporarily unavailable, ensuring continuity of critical tasks.

Digital twins provide a virtual representation of physical entities that can mirror and predict their behaviour in real time. A digital twin of the robot could provide richer data for agents' decisions in the DSR, predicting future states, simulating alternative actions or estimating environmental changes. A recent research involving 5G and digital twins for anticipatory prediction using CORTEX has been presented (Barroso et al., 2024). 6G technology is based on ultra-low latency, high reliability and massive bandwidth, which are essential for real time systems. This technology could improve the DSR synchronisation between multiple robots or systems in a faster and safer way. Moreover, high bandwidth data (like video, images or lidar) is efficiently integrated into the graph.

In summary, these technologies align perfectly with CORTEX architecture improving its modularity and shared context, and enhancing its scalability and ability to handle dynamic environments.

6.3. Limitations and future work at the knowledge representation level

One limitation of the proposed approach is that, as the number of agents incorporated in the robot architecture and IoT ecosystem increases, the software complexity also increases, and the debugging and validation process becomes more tedious. This is not a prerogative of our system since size and complexity usually grow together, especially in distributed architectures. A way out is to force the agents to follow a common design pattern that ensures robustness and reliability. A code generator would provide the required quality guarantees. Another limitation is that, in its current implementation, all software components that exchange δ-mutations must be on the same machine to achieve the maximum bandwidth available in the motherboard. Otherwise, and due to the use of multicast, the transmission speed is limited to the slowest connection, i.e. a WiFi link. In our setup, this implies that the DSR must run whether in the external NUC i5 Mini PC, or in the robot. To mitigate this problem, we are currently testing a proposal in which the robot and the Iot system run separate instances of the CORTEX architecture while keeping a unicast connection between their DSRS that selectively updates both graphs. In this configuration, the hard eventual consistency constraint is lost between both instances, but they maintain an independence that facilitates debugging and maintenance. In addition, this would allow the software components that manage the knowledge model to be better distributed among the two instances that now make up the IoRT system.

7. Conclusions and future work

This paper proposes the deployment of an IoRT ecosystem for AAL based on the CORTEX software architecture. In CORTEX, all software agents communicate with each other using a blackboard or working memory. However, unlike other approaches, in CORTEX, the DSR is not just the way for agents to communicate with each other, but a runtime model characterised by three elements: an original to which the model refers, a purpose that defines what the model should be used for, and an abstraction function, that maps relevant and purposeful features from the original to the model (Romero-Garcés et al., 2022). The distributed nature of CORTEX's working memory provides both real-time access by the agents to the shared data in linear time, thanks to multicast routing, and a stable context built by gathering information from multiple sources that provide each agent with the means to take better decisions. In addition, its encoding allows the DSR to be processed at runtime. This paper emphasises precisely this character, as it is processed at runtime in a strongly multi-platform framework.

CRediT authorship contribution statement

José Galeas: Writing – original draft, Investigation, Data curation, Writing – review & editing, Software, Formal analysis, Conceptualization, Validation, Methodology. **Alberto Tudela:** Writing – original draft, Supervision, Methodology, Conceptualization, Resources, Formal analysis, Validation, Software, Investigation, Data curation. **Óscar Pons:** Validation, Investigation, Methodology, Formal analysis, Software, Conceptualization. **Juan Pedro Bandera:** Writing – review & editing, Supervision, Formal analysis, Writing – original draft, Funding acquisition, Investigation, Conceptualization. **Antonio Bandera:** Writing – original draft, Supervision, Formal analysis, Visualization, Funding acquisition, Writing – review & editing, Project administration, Conceptualization. **Pablo Bustos:** Writing – original draft, Project administration, Investigation, Writing – review & editing, Supervision, Funding acquisition, Conceptualization, Methodology, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been supported by grants PDC2022-133597-C4X, TED2021-131739B-C2X and PID2022-137344OB-C3X, funded by MCIN /AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR (for the first two grants), and "ERDF A way of making Europe" (for the third grant). Also by FEDER Project 0124_EUROAGE_MAS_4_E (2021–2027 POCTEP Program). Funding for open access charge: Universidad de Málaga/CBUA.

Data availability

Data will be made available on request.

References

Almeida, P.S., Shoker, A., Baquero, C., 2018. Delta state replicated data types. J. Parallel Distrib. Comput. 111, 162–173.

Angulo, C., Pfeiffer, S., Tellez, R., Alenyà, G., 2015. Evaluating the use of robots to enlarge AAL services. J. Ambient. Intell. Smart Environ. 7 (3), 301–313.

Anon, 2015. Robotics 2020 Multi-Annual Roadmap for Robotics in Europe. Technical report, SPARC: The Partnership for Robotics in Europe, euRobotics Aisbl, Brussels, Belgium.

Anon, 2022. World population prospects. Report, department of economic and social affairs. Population division (UN). https://population.un.org/wpp/.

Anon, 2023a. Implementation of the united nations decade of healthy ageing (2021–2030): note / by the secretary-general. Report, United Nations. General assembly.

Anon, 2023b. Temi automatic fall detection and alerts system. Temi robot - temi robots for business, healthcare robot, retail robot, hospitality robot, telepresence robot. https://temibots.com/product/temi-automatic-fall-detection-and-alerts-system/.

Antonello, M., Carraro, M., Pierobon, M., Menegatti, E., 2017. Fast and robust detection of fallen people from a mobile robot. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 4159–4166. http://dx.doi.org/10.1109/IROS.2017.8206276.606.

Barroso, Sergio, Zapata, Noé, Pérez, Gerardo, Bustos, Pablo, Núñez, Pedro, 2024. Real-time hazard prediction in connected autonomous vehicles: a digital twin approach. In: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 3182–3188. http://dx.doi.org/10.1109/IROS58592.2024.10802333.

Blackman, S., Matlo, C., Bobrovitskiy, C., Waldoch, A., Fang, M.L., Jackson, P., Mihailidis, A., Nygård, L., Astell, A., Sixsmith, A., 2016. Ambient assisted living technologies for aging well: A scoping review. J. Intell. Syst. 25, 55–69.

Bui, H.D., Chong, N.Y., 2018. An integrated approach to human-robot-smart environment interaction interface for ambient assisted living. In: 2018 IEEE Workshop on Advanced Robotics and Its Social Impacts. ARSO, Genova, Italy, pp. 32–37. http://dx.doi.org/10.1109/ARSO.2018.8625821.

Bustos, P., García, J.C., Cintas, R., Martirena, E., Bachiller, P., Núñez, P., Bandera, A., 2021. DSRd: A proposal for a low-latency, distributed working memory for CORTEX. In: Bergasa, L.M., Ocaña, M., Barea, R., López-Guillén, E., Revenga, P. (Eds.), Advances in Physical Agents II. WAF 2020. In: Advances in Intelligent Systems and Computing, vol. 1285, Springer, Cham, http://dx.doi.org/10.1007/978-3-030-62579-5_8.

Bustos, P., Manso, L.J., Bandera, A., Bandera, J.P., García-Varea, I., Martínez-Gómez, J., 2019. The CORTEX cognitive robotics architecture: Use cases. Cogn. Syst. Res. 55, 107–123. http://dx.doi.org/10.1016/j.cogsys.2019.01.003.

Calderita, L., Manso, L., Bustos, P., Suárez-Mejías, C., Fernández, F., Bandera, A., 2014. THERAPIST: Towards an autonomous socially interactive robot for motor and neurorehabilitation therapies for children. JMIR Rehabil Assist. Technol. 1 (1), e1. http://dx.doi.org/10.2196/rehab.3151.

Calderita, L.V., Vega, A., Barroso-Ramírez, S., Bustos, P., Núñez, P., 2020. Designing a cyber-physical system for ambient assisted living: A use-case analysis for social robot navigation in caregiving centers. Sensors 20.

Cicirelli, G., Marani, R., Petitti, A., Milella, A., D'Orazio, T., 2021. Ambient assisted living: A review of technologies, methodologies and future perspectives for healthy aging of population. Sensors 21 (10), 3549. http://dx.doi.org/10.3390/s21103549.

Coradeschi, S., Cesta, A., Cortellessa, G., Coraci, L., Galindo, C., Gonzalez, J., Karlsson, L., Forsberg, A., Frennert, S., Furfari, F., Loutfi, A., Orlandini, A., Palumbo, F., Pecora, F., Rump, S.von., Štimec, A., Ullberg, J., Ötslund, B., 2014. GiraffPlus: A system for monitoring activities and physiological parameters and promoting social interaction for elderly. In: Hippe, Z.S., Kulikowski, J.L., Mroczek, J. (Eds.), Human-Computer Systems Interaction: Backgrounds and Applications 3. Springer International Publishing, Cham.

Deublein, A., Lugrin, B., 2020. (Expressive) social robot or tablet? – on the benefits of embodiment and non-verbal expressivity of the interface for a smart environment. In: Gram-Hansen, S., Jonasen, T., Midden, C. (Eds.), Persuasive Technology. Designing for Future Change. PERSUASIVE 2020. In: Lecture Notes in Computer Science, vol. 12064, Springer, Cham, http://dx.doi.org/10.1007/978-3-030-45712-9_7.

Do, H.M., Pham, M., Sheng, W., Yang, D., Liu, M., 2018. Rish: A robot-integrated smart home for elderly care. Robot. Auton. Syst. 101, 74–92. http://dx.doi.org/10.1016/j.robot.2017.12.008.

Embarak, F., Ismail, N.A., Othman, S., 2021. A systematic literature review: the role of assistive technology in supporting elderly social interaction with their online community. J. Ambient. Intell. And611 Humaniz. Comput. 13, 1–14. http://dx.doi.org/10.1007/s12652-020-02420-1.

Fürnkranz, J., Hüllermeier, E., 2003. Pairwise Preference Learning and Ranking, in Machine Learning: ECML 2003. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 145–156.

Gulzar, H., Shakeel, M., Itoyama, K., Nakadai, K., Nishida, K., Amano, H., Eda, T., 2023. FPGA based power-efficient edge server to accelerate speech interface for socially assistive robotics. In: 2023 IEEE/SICE International Symposium on System Integration. SII, pp. 1–6.

Hail, M.A., Fischer, S., 2015. IoT for AAL: An architecture via information-centric networking. In: 2015 IEEE Globecom Workshops. GC Wkshps, San Diego, CA, USA, pp. 1–6. http://dx.doi.org/10.1109/GLOCOMW.2015.7414020.

Hammer, S., Kirchner, K., André, E., Lugrin, B., 2017. Touch or talk? Comparing social robots and tablet PCs for an elderly assistant recommender system. In: Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction. HRI'17, Association for Computing Machinery, New York, NY, USA, pp. 129–130. http://dx.doi.org/10.1145/3029798.3038419.

Hanheide, M., Hebesberger, D., Krajnik, T., 2017. The when, where, and how: An adaptive robotic info-terminal for care home residents - a long-term study. In: 2017 12th ACM/IEEE International Conference on Human-Robot Interaction. HRI, pp. 341–334.

Iglesias, A., Viciana, R., Pérez-Lorenzo, J.M., Ting, K.L.H., Tudela, A., Marfil, R., Qbilat, M., Hurtado, A., Jerez, A., Bandera, J.P., 2024. The town crier: A use-case design and implementation for a socially assistive robot in retirement homes. Robotics 13, 61. http://dx.doi.org/10.3390/robotics13040061.

Jerez, A., Iglesias, A., Pérez-Lorenzo, J.M., Tudela, A., Cruces, A., Bandera, J.P., 2024. An user-centered evaluation of two socially assistive robots integrated in a retirement home. Int. J. Soc. Robot. http://dx.doi.org/10.1007/s12369-024-01175-5.

Jiang, Y., Gong, T., He, L., Yan, S., Wu, X., Liu, J., 2024. Fall detection on embedded platform using infrared array sensor for healthcare applications. Neural Comput. Appl. 36, 5093–5108. http://dx.doi.org/10.1007/s00521-023-09334-x.

Kara, D., Carlaw, S., 2014. The Internet of Robotic Things. Technical Report, ABI Research.

Karim, M., Khalid, S., Aleryani, A., Khan, J., Ullah, I., Ali, Z., 2024a. Human action recognition systems: A review of the trends and state-of-the-art. IEEE Access 12, 36372–36390. http://dx.doi.org/10.1109/ACCESS.2024.3373199.

Karim, M., Khalid, S., Aleryani, A., Tairan, N., Ali, Z., Ali, F., 2024b. HADE: Exploiting human action recognition through fine-tuned deep learning methods. IEEE Access 1242769–1242790. http://dx.doi.org/10.1109/ACCESS.2024.3378515.

Katter, V., Mahrt, N., 2014. Reduced representations of rooted trees. J. Algebra 413, 41–49. http://dx.doi.org/10.1016/j.jalgebra.2014.03.014.

Linner, T., Güttler, J., Bock, T., Georgoulas, C., 2015. Assistive robotic micro-rooms for independent living. Autom. Constr. 51, 8–22.

Liu, Z., 2023. Detecting falls through convolutional neural networks using infrared sensor and accelerometer. In: 2023 IEEE 20th International Conference on Smart Communities: Improving Quality of Life using AI, Robotics and IoT. HONET, Boca Raton, FL, USA, pp. 152–155. http://dx.doi.org/10.1109/HONET59747.2023.10374742.

Luperto, M., Monroy, J., Renoux, J., Lunardini, F., Basilico, N., Bulgheroni, M., Cangelosi, A., Cesari, M., Cid, M., Ianes, A., González-Jiménez, J., Kounoudes, A., Mari, D., Prisacariu, V., Savanovic, A., Ferrante, S., Borghese, A., 2022. Integrating social assistive robots, IoT, virtual communities and smart objects to assist at-home independently living elders: the MoveCare project. Int. J. Soc. Robot. 15, 1–31. http://dx.doi.org/10.1007/s12369-021-00843-0.

Macenski, S., Soragna, A., Carroll, M., Ge, Z., 2023. Impact of ROS 2 node composition in robotic systems. ArXiv 2023, abs/2305.09933.

Marfil, R., Romero-Garces, A., Bandera, J., Manso, L., Calderita, L., Bustos, P., Bandera, A., Garcia-Polo, J., Fernandez, F., Voilmy, D., 2020. Perceptions or actions? Grounding how agents interact within a software architecture for cognitive robotics. Cogn. Comput. 12, 479–497. http://dx.doi.org/10.1007/s12559-019-09685-5.

Meyer, S., Fricke, C., 2020. Autonomous assitive robots for older people at home: An exploratory study: He is always there for me-and I for him too. Z. Gerontol. Geriatr. 53, 620–629.

Modoni, G.E., Veniero, M., Sacco, M., 2017. Semantic knowledge management and integration services for AAL. In: Cavallo, F., Marletta, V., Monteriù, A., Siciliano, P. (Eds.), Ambient Assisted Living. ForItAAL 2016. In: Lecture Notes in Electrical Engineering, vol. 426, Springer, Cham, http://dx.doi.org/10.1007/978-3-319-54283-6_22.

Mojarad, R., Chibani, A., Attal, F., Khodabandelou, G., Amirat, Y., 2023. A hybrid and context-aware framework for normal and abnormal human behavior recognition. Soft Comput. 28, 4821–4845.

Naik, N., 2017. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In: 2017 IEEE International Systems Engineering Symposium. ISSE, pp. 1–7.

Nyangaresi, V.O., Shanshool, A.M., 2024. Smart wearables powered by AI transforming human activity recognition. Babylon. J. Artif. Intell. 128—133. http://dx.doi.org/10.58496/BJAI/2024/014.

Papadopoulos, I., Koulouglioti, C., Lazzarino, R., Ali, S., 2020. Enablers and barriers to the implementation of socially assistive humanoid robots in health and social care: a systematic review. BMJ Open 10 (1), http://dx.doi.org/10.1136/bmjopen-2019-033096.

Pham, V.C, Nguyen-Mau, T., Sioutis, M., Tan, Y., 2024. Matter and ECHONET lite: Similarities, differences, and a bridge solution for interoperatibility. Internet Things 27, 101265.

Rasch, R., Sprute, D., Pörtner, A., Battermann, S., König, M., 2019. Tidy up my room: Multi-agent cooperation for service tasks in smart environments. J. Ambient. Intell. Smart Env. 11, 261–275. http://dx.doi.org/10.3233/AIS-190524.

Ratnayake, M., Lukas, S., Brathwaite, S., Neave, J., Henry, H., 2022. Aging in place: Are we prepared? Del. J. Public Heal. 8 (3), 28–31. http://dx.doi.org/10.32481/djph.2022.08.007.

Ribeiro, O., Aráujo, L., Figueiredo, D., Pául, C., Teixeira, L., 2022. The caregiver support ratio in Europe: Estimating the future of potentially (un)available caregivers. Healthcare 10 (1), 11. http://dx.doi.org/10.3390/healthcare10010011.

Rodrigues, M.J., Postolache, O., Cercas, F., 2023. Wearable smart sensing and UWB system for fall detection in AAL environments. In: 2023 IEEE Sensors Applications Symposium. SAS, Ottawa, ON, Canada, pp. 1–6. http://dx.doi.org/10.1109/SAS58821.2023.10254065.

Romero-Garcés, A., Hidalgo-Paniagua, A., González-García, M., Bandera, A., 2022. On managing knowledge for MAPE-K loops in self-adaptive robotics using a graph-based runtime model. Appl. Sci. 12, 8583. http://dx.doi.org/10.3390/app12178583.

Sandhu, M., Silvera-Tawil, D., Borges, P., Zhang, Q., Kusy, B., 2024. Internet of robotic things for independent living: Critical analysis and future directions. Internet Things 25, 101120. http://dx.doi.org/10.1016/j.iot.2024.101120.

Sayeed, A., Verma, C., Kumar, N., Koul, N., Illés, Z., 2022. Approaches and challenges in internet of robotic things. Futur. Internet 14, 265. http://dx.doi.org/10.3390/fi14090265.

Singh, S., 2023. Intercompatibility of IoT devices using matter: Next-generation IoT connectivity protocol. In: Mishra, A., Gupta, D., Chetty, G. (Eds.), Advances in IoT and Security with Computational Intelligence. ICAISA 2023, In: Lecture Notes in Networks and Systems, vol. 756, Springer, Singapore, http://dx.doi.org/10.1007/978-981-99-5088-1_5.

Slavisa, A., et al., 2022. State of the art of audio- and video-based solutions for AAL. http://dx.doi.org/10.48550/arXiv.2207.01487.

Soldatos, J., Kyriazakos, S., Ziafati, P., Mihovska, A., 2021. Securing IoT applications with smart objects: Framework and a socially assistive robots case study. Wirel. Pers. Commun. 117.

Vermesan, O., Bahr, R., Ottella, M., Serrano, M., Karlsen, T., Wahlstrøm, T., Sand, H.E., Ashwathnarayan, M., Gamba, M.T., 2020. Internet of robotic things intelligent connectivity and platforms. Front. Robot. AI 7, http://dx.doi.org/10.3389/frobt.2020.00104.

Vuono, A., Luperto, M., Banfi, J., Basilico, N., Borghese, N.A., Sioutis, M., Renoux, J., Loufti, A., 2018. Seeking prevention of cognitive decline in elders via activity suggestion by a virtual caregiver. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. AAMAS'18, Richland, SC, USA, pp. 1835–1837.

Wengefeld, T., Schuetz, B., Girdziunaite, G., Scheidig, A., Gross, H.M., 2022. The MORPHIA project: First results of a long-term user study in an elderly care scenario from robotic point of view. In: Int. Symposium on Robotics. ISR Europe Munich, Germany. VDE, pp. 1–8.

Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., Zhao, H., 2024. Depth anything: Unleashing the power of large-scale unlabeled data.