Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Journal of Biomedical Informatics 108 (2020) 103483

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

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

Agents and robots for collaborating and supporting physicians in healthcare scenarios

Francesco Lanza[a,*], Valeria Seidita[a,b], Antonio Chella[a,b]

[a] Dipartimento di Ingegneria, Università degli Studi di Palermo, 90128 Viale delle Scienze, Palermo, Italy
[b] Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR), Consiglio Nazionale delle Ricerche, 90146, Via Ugo La Malfa, 153 Palermo, Italy

ARTICLE INFO

Keywords:
Multi-agent systems
Robots in therapy
Robots in Emergency Care for COVID-19
Patient monitoring
Human-robot interaction

ABSTRACT

Monitoring patients through robotics telehealth systems is an interesting scenario where patients' conditions, and their environment, are dynamic and unknown variables. We propose to improve telehealth systems' features to include the ability to serve patients with their needs, operating as human caregivers. The objective is to support the independent living of patients at home without losing the opportunity to monitor their health status. Application scenarios are several, and they spread from simple clinical assisting scenarios to an emergency one. For instance, in the case of a nursing home, the system would support in continuously monitoring the elderly patients. In contrast, in the case of an epidemic diffusion, such as COVID-19 pandemic, the system may help in all the early triage phases, significantly reducing the risk of contagion. However, the system has to let medical assistants perform actions remotely such as changing therapies or interacting with patients that need support. The paper proposes and describes a multi-agent architecture for intelligent medical care. We propose to use the *beliefs-desires-intentions* agent architecture, part of it is devised to be deployed in a robot. The result is an intelligent system that may allow robots the ability to select the most useful plan for unhandled situations and to communicate the choice to the physician for his validation and permission.

1. Introduction

Today there are many clinical scenarios in which a physician does not rely solely on himself, his knowledge or experience or his presence, to solve a patient's problems.

Current scenarios are made more complicated by the increase in the average life expectancy of citizens, especially in Western countries, which leads to an ever-increasing demand for healthcare systems. It is also remarkable today that societies are committed to ensuring access to care and well-being to all citizens [1]. There are also cases in which patients live in poor or not easily accessible places or in general cases in which the physician must provide, or receive if he is not on-site, fast and reliable diagnoses to be able to establish a therapy or otherwise solve a problem. This last is the case of the emergency due to the COVID-19 pandemic. The problem often faced by emergency room physicians was not to have the means for early identification of infected cases. This fact caused a lot of people infected and then dead also among doctors and nurses. Another case is the lack of professionals or, increasingly challenging, the presence of changing contexts. For instance, cases in which patients with the same disease but placed in a different family or social contexts have different characteristics and needs. Probably in these cases, a unique protocol cannot be applied, but doctors have to be able to decide on a case by case basis.

As evidenced by documents issued by the European Commission, the urgent need for intelligent systems for healthcare has not to be undervalued. Investigating the importance of AI & Robotics in healthcare is the current challenge for scientists and physicians, as illustrated by the Policy Department for Economic, Scientific and Quality of Life Policies in [2]. Here the main question is "Robots in Healthcare: a solution or a problem?". Following these research lines, we are investigating *"how can an intelligent system help a physician in making decisions, even in dynamic contexts?"*

Supporting physicians and patients in "complex" clinical contexts requires intelligent systems endowed with the ability to solve problems during interactions. During the execution, the system interacts with users and the environment that often change continuously as a result of the interaction.

These challenges may be solved by multi-agent systems able to self-adapt to changing situations and deliberate also in the total or partial absence of input data from physicians or patients.

These aspects cannot be faced and solved at the design time. Developers cannot identify and implement all the possible situations

* Corresponding author.
E-mail addresses: francesco.lanza@unipa.it (F. Lanza), valeria.seidita@unipa.it (V. Seidita), antonio.chella@unipa.it (A. Chella).

https://doi.org/10.1016/j.jbi.2020.103483
Received 30 January 2020; Received in revised form 1 June 2020; Accepted 8 June 2020
Available online 27 June 2020
1532-0464/

where a high level of autonomy is required. At best, they can identify several conditional statements and allow for a set of possible alternatives in the system behavior.

The more dynamism or uncertainty in clinical contexts exist, the more physicians may need to be supported by an intelligent system. In this situation, developers have to implement mechanisms that let the system autonomously monitor patients, retrieve important information, reason and suggest actions to physicians if necessary. All this may be done employing robots.

Thus, an efficient way to face the problem mentioned above is using the agent-oriented paradigm [3] and robots. The use of agent-oriented paradigm in complex systems, such as robotic platforms or internet of things applications, was studied and treated by scientific community [4–6] during these years. For instance, the application of the agent-oriented computing, for the IoT domain and the cyber-physical systems, is increasing day-by-day. Weaknesses and issues in these domains can be solved exploiting agent-oriented architectures and relative computing techniques as well discussed and highlighted in [7]. Another way to employ multi-agent systems for developing monitoring systems in the healthcare scenario may be seen in [8]. However, multi-agency has not been used for providing support in dynamic environments for healthcare domain. The contribution and the novelty of this paper lay in the creation of an agent-based architecture for healthcare systems. The architecture handles monitoring, knowledge management and deliberation module. The main novel idea we present is to employ the *Belief-Desires-Intentions* (BDI) paradigm and its reasoning cycle for implementing a multi-agent system able to deliberate and plan teleoperation activities to help physicians in making decisions. Even when information about plans to execute and goals to pursue are lacking or incomplete. The multi-agent system resulting from the proposed architecture has been conceived for being deployed in a robot to support physicians or patients in their activities.

The rest of the paper is structured as follows: in Section 2 some related work are explored; Section 3 briefly shows multi-agent systems topics, their features and how they handle the selection of plans during the execution phase; in Section 4 we discuss the architecture we propose for assisting physicians and how plans are created is illustrated with an example; in Section 5 we validate the proposed architecture and finally in Section 6 some discussions and conclusions are drawn.

2. Related work

The need to build a robotic system able to satisfy patients brought scientists to design robotic platforms to face this problem, as in the case of Koceska et al. [9]. Koceska et al. studied available platforms in the market of assistive robot systems. They designed and developed a low-cost assistive telepresence robot system for facilitating and improving the quality of life of elderly and people with disabilities. To improve the quality of life the robot system is able to interact with the environment (such as, moving small objects and measuring vital parameters) and can be controlled by a remote assistant. The robot merely executes the commands of the remote assistant. An advantage is that this is a low-cost robot that can be easily used in everyday life by patients and professionals.

The need for a telepresence robotic assistant has driven the interest of the EU itself, which has financed FP7 and H2020 European Projects for this purpose.

MARIO project[1] faces an important challenge into the field of telepresence robotic assistant. It aims to challenge loneliness, isolation and dementia in elderly people [10–13]. The project connects elderly people with their needs. It aims to be independent of robotic platforms. Moreover, it supports remote application installation and deployments onto the robotic platforms. The project uses data exchanged during personal or social interaction. Even if the project faced these challenges, no dynamic supports to patients or physicians were considered. It does not facilitate the interaction including new activities at runtime without the contribution of developers.

ENRICHME project[2] is another EU project financed through the H2020 Framework Program [14,15]. The project aims to face the cognitive decline of cognitive capacities in elderly people. The solution consists of an integrated platform for ambient assisted living, integrated with a mobile service robot. The project aims to realize long-term human monitoring and interaction system for letting elderly people stay independent in their home way. The strength of the project is to enable physicians and caregiver to analyze data for identifying changing in cognitive impairments and so early acting on them. This system does not take in consideration robots as a human supporter.

MOVECARE project[3] aims to realize *"an innovative multi-actor platform that supports the independent living of the elder at home by monitoring, assisting and promoting activities to counteract physical and cognitive decline and social exclusion"* [16–18]. The MOVECARE architecture monitors the frailty of the elderly specifically based on criteria identified and discussed by Fried et al. [19]. Like the previous approach, this framework provides a good means for monitoring; it adds some kind of intelligence during the reasoning process. The reasoning system can alert caregivers about the status of the patients and possible motivations exploiting the Fried frailty criteria. Nonetheless, the robotic platform is not designed for acting as a true caregiver.

Caresses project,[4] that stands for "Culture Aware Robots and Environmental Sensor Systems for Elderly Support", is an H2020 Project financed by EU for building culturally competent care robots [20–23]. The project aims to build the first robot that assists the elderly, adapting itself with the culture of the user. At the best of our knowledge, the platform is not able to handle runtime planning for executing therapies.

All previous projects consider several important aspects of telemedicine, assistive robotics and human-robot interaction for healthcare, but activities are not thought to support physicians by remote in a proactive manner.

Another approach in the literature, to perform robotic teleoperations for ultrasonic medical imaging, is the `OTELO System`. `OTELO` performs tele-echography [24] remotely. It supports the clinicians in making a diagnosis. This system even treats patients in their home, does not operate proactively and it needs for physicians to be used.

Progress in networking lets computer scientists build applications using a network to exchange data and information. Systems can exchange information using the network and the spread of wireless. In [25] authors propose a mobile-care system integrated with a variety of vital-signs monitoring, where all involved devices are endowed with a wireless communication module for data exchanging. The mobile-care system uploads data into a care server via the internet. Data stored into the remote server are available for physicians that need to check the health status of patients. The system serves as an alert system where interaction with patients is implemented.

Evolution of telemedicine systems brought at the definition of novel systems able to handle more complex scenarios. Recent works introduce a new technological paradigm for elderly people that live alone. In [26] authors consider IoT (Internet of Things) technology to connect devices in patient's home for data collecting and communication. Other studies involve the usage of robots, or better social robots, as telepresence systems [27]. In this work, the authors proposed a telepresence robot, built for a human-robot interactive experiment.

Each of these works aims to realize complex systems with the ability to supervise people at home letting them live without invasive instruments through the usage of robotics or sensors scattered into their

[1] https://cordis.europa.eu/project/id/643808.

[2] https://cordis.europa.eu/project/id/643691.

[3] https://cordis.europa.eu/project/id/732158.

[4] https://cordis.europa.eu/project/id/737858.

homes. Other approaches are present in literature, most of them resolve problems related to telepresence, monitoring and teleoperation.

We are now facing the following problem: each patient's environment is not equal to another leading to difficulty in handling new occurring situations by clinicians. Moreover, every patient answers therapies in different ways and so, every care system has to be endowed with the ability to adapt its behaviors taking into account this weakness. For that, scientists and engineers need for methods to facilitate reasoning operation at runtime.

Runtime reasoning methods arouse a strong interest in the robotics community for modeling complex domains, such as the clinical one. The state of the art of healthcare system presents solutions aimed at assisting patients and physicians in performing monitoring and other simple operations. Physician interacts with the system that interacts with the patients and vice versa. All these approaches do not solve problems related to changing clinical scenarios.

Additionally, our approach employs a multi-agent architecture that gives intelligent support to the physician. The architecture, deployed in a robot, monitor and assist the patient as a caregiver, also in cooperation and collaboration with the doctor.

3. Multi-agent system

An agent is an autonomous entity able to act in response to stimuli coming from the environment and proactively act towards a specific goal [3,28,29]. The agent paradigm was born to better understand and model complexity in software. Interaction among components is the main characteristic of complex software. Correctly engineering and implementing systems where complex components interact with each other is harder than engineering systems requiring the computation of single or simple functions. Agent theory is suitable for facing this kind of situation.

An agent has the following important properties: autonomy, reactivity, proactiveness and social ability. To understand what autonomy means when talking about agents let us refer to a functional program, a Java class, a compiler or something like that. All these kinds of software may be modeled by a function, they receive input and produce an output as the result of elaborating that input. Everything happens in this application is because we (the programmers or the designers) want it to happen and to happen exactly in that way. An autonomous agent is conceived to be at the exact opposite of the applications above. Research in the field of agents and multi-agent systems is going towards means for building agents to which we can delegate tasks. It is up to the agent to decide how to reach the objectives, they act on the base of plans we give them. A plan defines the set of actions an agent may perform to pursue an objective. An agent is reactive in the sense that it is able to perceive the environment and act in response to changes coming from it, actions are directed towards the agent's objective. One of the key points related to agent autonomy is that an agent may put together plans to achieve our goals. So they are making able to operate autonomously on the behalf of humans that delegate them their goals. Proactiveness is the ability "to exhibit a goal-directed behavior by taking the initiative". Social ability is the ability of an agent to interact with other agents and humans to purposefully reach its objective. This latest ability involves an important skill belonging to humans, i.e. communication abilities. To establish a society, the agent cooperates and coordinates activities with other agents and therefore it is able to communicate to the other (also humans) its beliefs, its goals and its plans.

Healthcare systems or other kinds of systems supporting physicians in the scenarios identified in the introduction are perfectly manageable with agents. An agent with the property above may be delegated to solve problems on behalf of the physician. It may be planned for monitoring patients and inform of all possible situations deviating from a specific protocol. The agent may autonomously establish an alternative plan or action to perform to pursue a specific goal and may communicate to the doctor to support him in the decision. Note that it is not a simple teleoperation ability but a real intelligent aid.

A widely recognized agency model in literature is the *Beliefs-Desires-Intentions* (BDI) model. This AI-based agent paradigm [30] involves a deliberation ability of the agent, based on a continuous sense-action loop and the evaluation of existing beliefs. It allows the agent to realize a desire with a plan available in its library. This model originates from the theory of practical reasoning by the philosopher M. Bratman [31]. *Practical reasoning* is a human ability, the reasoning is directed towards actions. *"Practical reasoning is a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes."* ([32], p.17).

In this section, we briefly outline the basic concept regarding the BDI agent paradigm and some ideas on how it might be useful in the clinical domain.

The definition of *practical reasoning*leads to taking into consideration two factors, the *belief* and the *desire*. *Beliefs* are all facts or information about the environment the agent considers valid. Beliefs constitute the actual knowledge of the agent. *Desires* are all the state of affairs the agent wants, or better would like, to reach. Desires relate to the goal the agent has been delegated to. Commonly, an agent has conflicting desires. Not all the desires it owns are devoted to be committed. Only intentions are pursued. The *intention* is the state of affairs, an objective also related to the state of the world, that the agent decides to reach, on which it commits to work on. We can imagine an agent having the goal of monitoring a patient for sending information to the physician, remembering or administering the patient some therapy he has to do, updating medical records, ensuring that the primary needs of patients are met. The agent is delegated to reach all those objective but it is up to it to decide which one to pursue in a specific moment and on the base on the information (all the beliefs) it has on the environment. When it chooses a desire, it commits to an intention and decides the course of action to pursue that intention. The course of action is called *plan*.

Moreover, humans actuating practical reasoning performs two important activities: deliberation and means-ends reasoning. The former is the action of deciding which intention to commit and the latter is about deciding how to act. BDI agents are developed to be deliberative and means-end reasoner. In a clinical scenario, having agents behaving in a human-like fashion, on the behalf of the physician, may be a great support. A software system, also deployed in a robot, that merely executes orders by physicians may be helpful. What if the software system is able to deliberate how to act when the situation is not the one envisioned by the physician? For instance, let us suppose that while the agent is administering a therapy, the patient's conditions can change. The changing produces a revision of the agent's beliefs, triggering other options, such as advise the physician and wait for his instructions.

This kind of support cannot be reached by using a simple reactive system. In the following section, we explain the agent architecture, involving BDI agents, we propose for physicians support in a clinical scenario.

The computational model implementing deliberation and means-ends reasoning contains four elements: B (beliefs), I (intentions), D (desire) and π (plan). At the beginning agent is endowed with a belief base, hence its knowledge about the environment, and a plan library, a set of possible plans for reaching objectives. A plan is selected and then activated only if some preconditions are true. The model develops on a loop for which an agent starts by updating the belief base, is something like getting aware of the world around, and choosing an intention. During the loop, the agent continuously perceives from the environment, if necessary it updates the belief base, determines desires and intentions by computing current beliefs and then generates a plan to reach the intention. Generating a plan means to select the best plan from the plan library, the one that fits with all the intention's precondition and executing all the actions it contains. After the execution

of each action, the agent pauses and observes the environment. The agent might need to update the belief base and reconsider the intention. In the worst case, if no plans or no actions exist in the plan library for reaching an intention the agent should be provided with other plans, hence its plan library has to be enriched. For instance, in the clinical scenario, if the robot does not know how to reach an intention, the robot could interact with the physician for a new plan to be added into the plan library, remotely.

Moreover, letting the agents reasoning on the belief base and on the current environment allows us to develop a robot able to explain the reason for failure. In so doing the interaction with the physician may reach a high degree of intelligent support.

Several technological approaches in the literature describe possible implementations of BDI agents. One of the most known and efficient ones is the *Jason* framework and its reasoning cycle [28,33]. *Jason* is a powerful instrument for realizing planning in uncertain environments.

In the next section, we detail the proposed agent architecture exploiting the said BDI features and we explain how we implement the architecture in a robot supporting and cooperating the physician for reaching a common goal.

4. A multi-agent system for healthcare

This section starts with the description of a pilot scenario, useful for deeply understanding the domain context.

4.1. Pilot scenario

Involved actors are listed in the following table (Table 1).

Alice is a nice elder woman with chronic bronchitis and she has to stay at home. She is assisted by MyRob robot. MyRob is in charge of monitoring Alice and her environment to collect her health-data.

Dr. Haus prescribes some medicines. MyRob daily collects and sends data to let Dr. Haus check Alice's health status or revise prescribed therapies.

The data is stored into a data repository where Dr. Haus, through an application, analyzes them and decides new therapies or confirms the previous one. If Dr. Haus, on the base of the analyzed data, finds the therapy good nothing changes and MyRob continues its tasks without interruptions otherwise MyRob may send feedback or recommendations to Dr. Haus.

So far, social robots used as a companion or virtual caregiver [34,35] satisfactorily face these kinds of duties.

What about if prescribed therapies are not adequate, because something unforeseen occurred in Alice's status, and Dr. Haus needs to revise them collaborating remotely with MyRob?

MyRob checks for alternative actions to apply to the context and asks Dr. Haus's authorization for proceeding, otherwise it contacts him to inform about the situation. In the latter case, Dr. Haus decides to change the therapy and through his terminal, he proceeds to send the new therapy to MyRob. So that, MyRob is allowed to change dynamically the therapies.

For instance, let us suppose that MyRob has been charged to open the window each time the quality of air lowers below a threshold and to administer timely previous medicines to Alice. MyRob is going to open the window but perceives an increase in Alice's body temperature. MyRob recognizes the right conditions for opening the window now lack. MyRob contacts Dr. Haus, informs him about the new Alice status and requires his consent for not opening the windows and administering paracetamol. If MyRob does not know this alternative action, it alerts Dr. Haus for receiving commands. Dr. Haus may suggest administering paracetamol.

Table 1
Actors involved in the pilot scenario.

Patient:	Alice
Robot Caregiver:	MyRob
Physician:	Dr. Haus

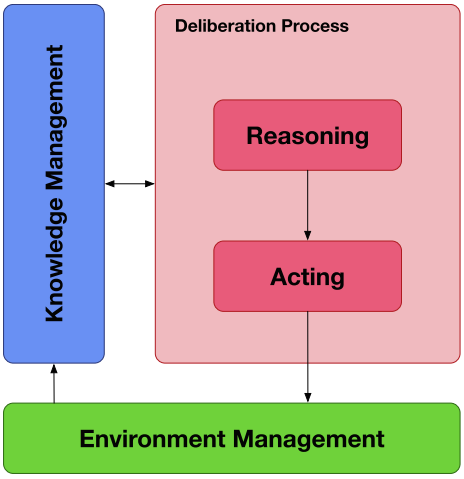

Fig. 1. The architecture for implementing the Boyd cycle to let agent observing, orienting, deciding and acting.

4.2. The system overview: architectural aspects

To accomplish the previous scenario we need to develop a robot that might serve as an intelligent object for monitoring and checking patients in their environment and also as a collector of data for enhancing diagnosis. The robot has also to be able to change its behaviors at runtime according to the physician's prescriptions.

The idea is to develop such a kind of robot as a part of a cognitive system realized employing BDI agents and on the base of the following architecture (Fig. 1).

The architecture is composed of four modules:

- **Environment Management** –it contains all useful elements for interfacing physicians, for monitoring the environment in which patients live and for acquiring data;
- **Knowledge Management** –it is the module for storing and managing data from the environment and patient;
- **Reasoning** –this is the module devoted to compute data stored into the knowledge module and to produce a series of actions to accomplish a task;
- **Acting** –given a set of actions, it extracts the proper action for a specific situation.

The Knowledge Management module depends on the Environment Management module, indeed only data resulting from monitoring form the knowledge of the system.

Data stored in the Knowledge Management module are used by the system for the reasoning process and, at the same time, all the results of the reasoning process update knowledge.

The Reasoning module produces a set of actions, useful to reach one or more system goals.

This set is the input for the Acting module where, it extracts the proper action sent to the system. The result of an action normally produces a change in the state of the environment. So, this module affects the monitoring one.

The architecture in Fig. 1 is implemented employing a BDI multi-agent system where a set of agents works to reach the system's goal.

4.3. A multi-agent approach for healthcare systems

The efficiency of the multi-agent paradigm in modeling and

handling complex systems has been widely discussed in the literature [3,36]. Handling robots via a multi-agent system means creating a multi-agent architecture able to handle components and planning abilities.

As said, our idea is to use the BDI agents paradigm for providing a intelligent support to the physician work, also allowing him to manage robot remotely. Beliefs, rules, desires and intentions for each agent are defined at design time by developers and users (physicians and patients in the clinical domain).

Beliefs are generally of two types: *(i)* perceptions acquired through sensors from the surrounding environment; *(ii)* information acquired through messages exchanged with the other agents into the platform.

Beliefs are critical for the agent's planner. Beliefs and rules push a plan to be selected as the best set of activities to do. Selecting one plan, compared to another, means changing the behavior of the agent during its operating cycle.

The agent's intelligence is defined with a descriptive file written in AgentSpeak(L) [30,37]. AgentSpeak(L) is a language that uses logical formalism to define a set of plans for handling situations to reach the agent's goal. The *agent programmer* writes initial beliefs, rules, goals and plans.

Each plan is composed of a head and a tail. The head is composed of a *event trigger* and a *circumstance*. The former is the trigger condition that launches actions defined into the plan and the latter is a set of parameters used for validating the context in which some actions are allowed. More in detail, a plan is launched when the event trigger is scheduled and the set of beliefs contained in the circumstance are solved by a unification process where a literal is unified with the data contained into the namesake belief and verified by a first-order logic process.

The tail of a plan is composed of sub-plans or actions that can be internal actions or external actions.

In the next paragraph, we show the architecture detailing how agents work, communicate and cooperate to support physicians.

4.4. Agents working in the healthcare architecture

Fig. 2 represents the multi-agent system that works in a typical healthcare scenario. In this scenario, two environments are involved, the patient's residence and the physician's office.

The two environments are connected through a multi-agent platform, each environment owns one or more agents, or better one or more agents are deployed in each environment. The multi-agent platform serves as a bridge among the patient and the physician. It is worth to note that the middle layer of this representation constitutes the intelligent part to be added to a simple teleoperated system (i.e. the two environments and the software interfacing robot and the physician). The contribution of this paper lays in how we conceive the intelligent bridge.

The physician (Fig. 2) uses his terminal to access the patient's environment and retrieves data about the patient's status and the surrounding context. The patient communicates directly with the robot and vice versa.

The cognitive model, followed by agents in the architecture figured out in Fig. 1, starts its reasoning cycle from the environment managing module. It (see Section 4.2) is responsible for acquiring information from the environments. Collected data are organized and stored to be saved in the remote server. The Knowledge Management module organizes data to be synced with the system and updates them with the last perceived one for the diagnostic purpose by physicians. The same data are handled from the agents into the deliberation process module. This module is composed of two sub-systems, the first for reasoning and the second for acting.

Multi-agent systems are distributed systems over the network. Each agent could be relocated to other computers that host a node or a set of nodes of the system infrastructure.

The multi-agent system can be distributed over remote sites connected via the internet. The system we propose uses the internet to share information, data and alerts between the physician's office and the patient's environment.

The physician's terminal is connected to the multi-agent system through the *Virtual Assistant* agent. It implements all necessary functionalities to be into agents' network and it receives data from the patient's environment via the internet.

Data are stored using an OWL ontology, they are accessible from the physician by a terminal for diagnostic purpose. Data are also stored in remote servers for safety reasons. A memory manager holder organizes acquired perceptions and collected information by the system in an OWL ontology [38]. The knowledge is split into a long-term and working memory; in this way, we avoid problems such as memory leak or memory explosion from agents when they have to use it. So the memory is handled as described in [39] and the agent which was delegated to manage knowledge is deployed into the system with the goal to keep all data synced and updated. The knowledge of the system is

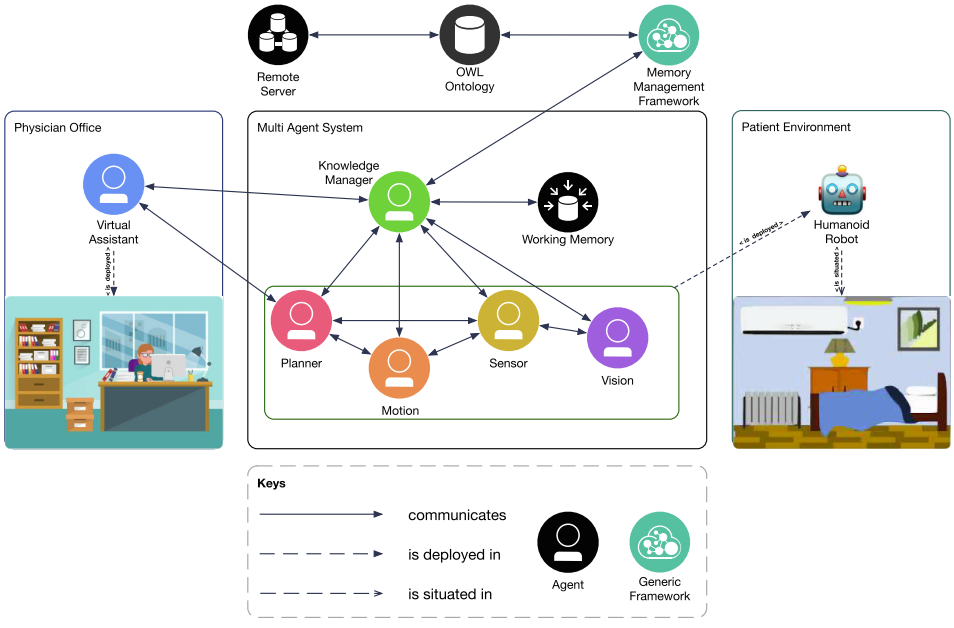

Fig. 2. The multi-agent system for a robot in a healthcare scenario. The system shows circles for software modules such as agents and other used frameworks and three kinds of relations (arrows). The continuous arrow indicates communication between modules, dashed arrows are activities that identify on which nodes agents and robots work. The relation: *(i)* `< is deployed >` is used for software parts; *(ii)* `< is situated >` for cyber-physical system.

Table 2
The table summarizes the role of the agents into the proposed multi-agent system, figured out in Fig. 2.

Agent	Role
Knowledge Manager	It is a part of the multi-agent system used for managing the system's knowledge. It uses a double kind of knowledge model, an ontology for storing information and gathering them as concepts and relations and a knowledge base where beliefs are stored for acting in the working domain.
Planner	It is a part of the multi-agent system used for planning operations. This agent is delegated for selecting which plan is more adapt to pursuing the goal. Generally, the system goal is defined at design time but it can be handled at runtime. This agent listens for information gathered and it deliberates the set of actions and plans that each agent has to execute using the agent's communication module. This agent communicates directly with the virtual assistant agent. This last is responsible for upgrading the planner's plan library to add plans and enabling it for revising plan at runtime; this means that the multi-agent systems can act dynamically.
Vision	It is a part of the multi-agent system used for vision task operations.
Sensor	It is a part of the multi-agent system used for collecting data from sensors. These data contribute in decision-making phase, to select the plan that best fit with the current situation.
Motion	It is a part of the multi-agent system used for handling robot motion. This agent implements modules for letting robot move and executing into the environment.
Virtual Assistant	It is a part of the multi-agent system used for letting physician be part of the system. This agent operates as a dashboard console on which the physicians retrieves data monitored by the robot and collected from the system. The agent lets physicians communicate new therapies to the robot that they will be translated into plans and added into the plan library of the planner agent. Therapy will be executed during the next agent's reasoning cycle if pre-conditions signed by the physicians will happen. If this does not happen, an alert will be sent to the dashboard console.

organized to be always synced, updated and shared on the entire system and to avoid possible conflicts given the nature of beliefs. Indeed, beliefs may change during the interaction with the environment or with the patient so it is necessary to guarantee the right knowledge synchronization among all the modules.

The *Virtual Assistant* agent is deployed into the physician's personal computer as a computer application. It works as a remote controller application and it endows the physician for accessing the patient's environment.

The *Virtual Assistant* agent assists physicians in remotely collaborating with the robot located in the patient's environment. Admitted operations let physicians manage the robot, such as supervising the robot's behaviors, stopping or restarting it if necessary.

As said in Section 2, nowadays, there exist several teleoperated robots that let the physician operate into the patient's environment, no one of them works in autonomy during the interaction with the patient. Autonomy, in this case, means to be able to recognize or retrieve the best action to perform in a particular situation, propose the action to the physician and wait for his command. In this sense, the robot is autonomous and it can suggest some actions to the physician. Indeed, in our system physicians can teleoperate in the sense of changing previous therapies or adding new ones to enhance the quality of life of the patient. The robotic caregiver should not be seen as a professional robotic avatar but as a valid instrument useful for monitoring and assisting patients in the place of clinicians.

Virtual Assistant agent endows physician for checking, revising, updating a therapy or a set of therapies initially prescribed and provided to the robot for taking care of the health status of the patient. Therapies are plans in the sense of the agent's plan that the *Planner* agent selects when the context is verified.

The *Planner* agent is responsible to realize the intelligence of the system. It communicates with the *Knowledge Manager* agent to obtain data synced, updated and stored into the ontology for selecting the best plan that fits the current situation. To satisfy the system goal, the *Planner* agent tries to find the right solution using the context of a logical formula; once the context is verified a combination of actions and other plans let the system move toward the goal.

Logical formulas, in multi-agent systems, are also called plans and each plan is generally pushed into a repository of plans called Plan Library. Plans use perceptions and information acquired through the *Vision* and *Sensor* agents and are put in action through the *Motion* agent.

Some instances of *Vision* and *Sensor* agents are deployed into the robot, other instances that perform heavy computational processes are distributed on the same node of the *Knowledge Manager* agent or in other nodes if necessary, thus giving a high level of scalability to our system.

In Table 2 is summarized the description of the role of the agents that are into the system.

4.5. How the robot is employed in the clinical scenario

The robot situated in the patient's environment works as a robotic caregiver.

The multi-agent architecture contains modules for handling knowledge, planning, motion, vision and sense for robots. Each agent implements its logical model to accomplish established goals that represent the desires of the agent.

The robot uses algorithms for implementing obstacle avoidance and motion. The agent may navigate in unknown indoor places using an adapted version of a state-of-the-art algorithm for robot SLAM (Simultaneous Localization And Mapping) [40].

The *Motion* agent is strictly connected with the *Vision* agent. While the former deals with the motion of the robot, the latter deals with computer vision tasks. The *Vision* agent, continually, senses the environment using RGB camera to discover and to recognize objects located into the environment.

The object recognition module works using the YOLO deep neural network[5] [41]. YOLO is a state-of-the-art system for implementing real-time object detection and recognition. Recognizing objects is useful for motion's tasks, such as obstacle avoiding or to enhance the localization algorithm using some objects as landmarks or fixed-points.

The *Vision* agent is also able to detect the status of the patient, such as understanding when the patient is sitting or lying down or the patient is standing. *Vision* agent takes also care of other tasks; a deep description of this module is out of the scope of the paper.

Other agents are deployed into the multi-agent platform to sense the environment and to enhance the quality of perceptions. An agent, delegated for handling information acquired by sensors, continually senses the environment with the aim to keep updated telemetries about the surrounding context. Data acquired are stored and shared between all agents.

Moreover, several vision tasks and heavy computational processes used by the robot are distributed over other nodes in the architecture for balancing the robot's workload.

Beliefs shared into the platform by agents are used to select the best plan that fits with the goal of the system. Methods are implemented as internal actions or operations for the agent. Perceptions are acquired through sensors and cameras. Sensors acquire information from the surrounding environment and each perception is translated into the

[5] YOLO v3 website: https://pjreddie.com/darknet/yolo.

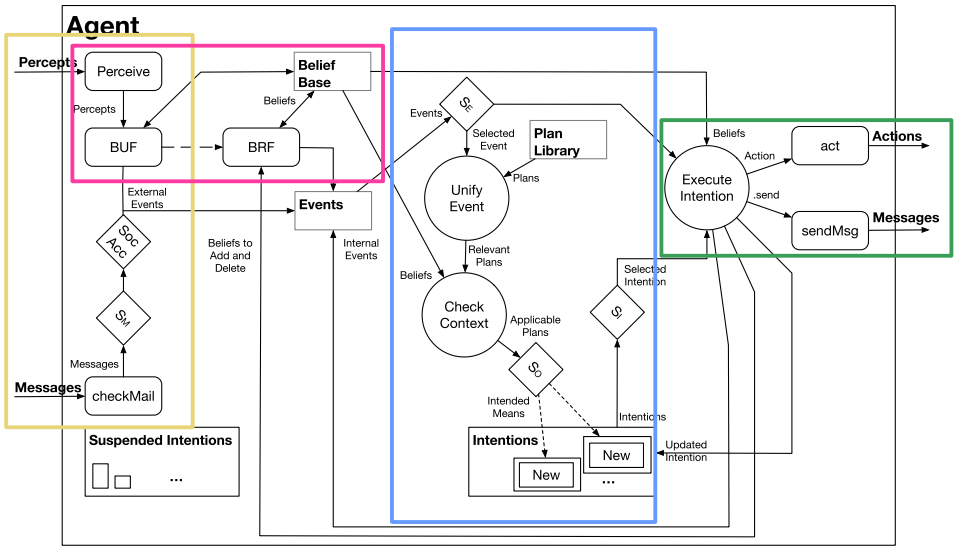

Fig. 3. The Jason reasoning cycle represents the reasoning process behind mechanisms that move the intelligence of the agent platform. This picture figures out the modules composing the `Agent class` in `Jason` framework. In `yellow` are marked `modules` for perception phase, in `pink` are highlighted beliefs revision `modules`, in `blue` are emphasized `modules` for decision-making phase and in `green` the acting `modules`. The reasoning cycle is redrawn from [29]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

corresponding belief and added to a set of beliefs handled by the knowledge manager as described.

4.6. Details towards the implementation

Our multi-agent system is implemented using the Jason framework [33,29] and the agent's reasoning cycle is well described in [29,42,43].

Briefly, each Jason agent reasoning cycle (Fig. 3) involves four phases: (*i*) sensing phase (in yellow); (*ii*) beliefs revision and updating phase (in pink); (*iii*) deliberative phase (in blue); (*iv*) acting phase (in green). Each cycle starts perceiving the environment and ends with acting operations in the surrounding context.

Once the agent has perceived something from the environment, internal knowledge handling is executed. The core of the reasoning cycle is on how the agent selects the plan that best fits with the context. The reasoning cycle is responsible for the rational process for robot acting.

As said before, plan is composed of three parts:

triggering event: context ← sub plans; actions; messages.

and a unification process over beliefs and a first-order logic is applied from the reasoner.

The first-order logic evaluates logical conditions over boolean expressions. The formalism used in Jason is inherited from the reasoning system developed using AgentSpeak(L) [37]. A list of plans is written at design-time by *agent programmer* according to the goal that the agent has to pursue. Each plan is evaluated in the order in which it is written into the agent description file, written in AgentSpeak(L) language.

The order in which plans figure into the Plan Library is fundamental for the decision-making process. In fact, the Plan Library is handled as a queue, where the first plan deployed into the Plan Library is the first that is select to be checked and executed if it is valid.

The Plan Library contains all plans written at design-time and can be handled during agent life-cycle adding, revising or removing predefined plans. In this sense, the first plan in the queue that best fit with the condition is executed.

Executing plans results in actions over the environment or exchanging information with other agents in the system.

The *Planner* agent communicates with the *Knowledge Manager* and *Virtual Assistant* agents for teleoperating activities and with the *Motion* agent to move into the environment for performing operations. The *Planner* agent's *plan library* can be manipulated and it is partially revised for changing behaviors at runtime. Ideally, this agent is divided into two parts: (*i*) plans for robot acting, (*ii*) plans for administering therapies. The former is configured for permitting to operate and this section of plans are written at design time by *agent programmer*, the latter is the section of plans that the system activates to administer therapies to the patient to serve the physician and taking care of the patient.

In the next section, we describe how to revise therapies remotely starting from diagnosis and we introduce the mapping process between therapies and agents' plans.

4.7. Handling diagnosis and therapy by means of plans

Diagnosis represents the status of the patient in a particular moment of his life and the triggering condition for administering a therapy.

Therapies or medical treatments are prescriptions aiming to heal a person with health problems. Each therapy contains a list of indications and contraindications for the patient and it can be effective or not. There are several types of therapies, in our approach, we take into consideration only the following type: *Procedure and Human Interaction – They consist of procedures and practices administered by humans. In this type falls therapies for counselling, family, education but also psychotherapy, cognitive behavioral therapy, rehabilitation, physical therapy such as vision therapy, massage therapy. Lifestyle modification and life-coaching fall also in this category.*

Each therapy is supported by a set of conditions to apply it. Conditions are made by individuating data acquired from the physician, observing patient's behaviors and activities.

Let us consider the Pilot Scenario. Alice is affected by bronchitis, so Dr. Haus inserts his diagnosis and the related therapy by using his remote controller application. The mock-up in Fig. 4 represents the interface of the remote controller application. Handling this interface is part of the work of the *Virtual Assistant* agent. From the point of view of the agent, the diagnosis is translated into a belief or a set of beliefs useful for selecting a plan. Instead, therapy is translated into one plan. The system is programmed in a way that lets MyRob to constantly monitor Alice, administer therapy and update the physician through the *Knowledge Manager* agent.

Suppose that something new happens, Alice worsens and a sudden fever appears. MyRob detects the fever interacting with her and it signals the *Virtual Assistant* agent notices the event and an alert is shown into the Dr. Haus terminal.

Even if Dr. Haus has just deployed some plans for handling fever and MyRob could select the best plan to face this new situation, it alerts him and waits for a consent for administering the alternative therapy. Dr. Haus can confirm previous therapy and lets MyRob administer the

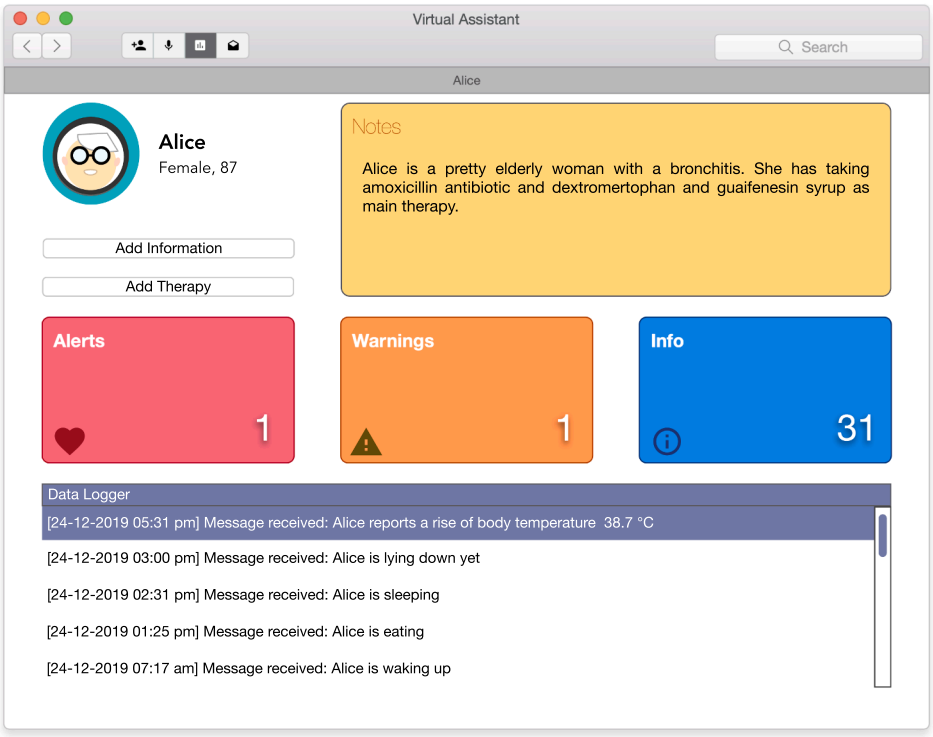

Fig. 4. A mock-up of the Virtual Assistant agent deployed into the physician's terminal. The tab contains details about the patient. Alerts, warnings and info are shown into the namesake pills, the data-logger contains exchanged messages.

recovery plan or inspect data for changing therapies. If Dr. Haus decides to consent previous therapy, the *Virtual Assistant* agent sends an ack to the *Planner* agent and MyRob continues working applying it, otherwise, Dr. Haus has to produce a new therapy. The multi-agent system allows Dr. Haus to insert all the needed therapies for facing the disease. He does this by evaluating information acquired by MyRob through its components.

Dr. Haus compiles the new therapy following some basic rules in order to be compliant with the system. Rules are necessary for translating therapies in the "AgentSpeak Language". This task is also in charge of the*Virtual Assistant* agent.

The translation process follows some basic rules, resumed in Table 3:

- a therapy owns a triggering event, conditions for detecting it and a list of procedures to act when conditions appear and they are verified;
- the triggering event has to catch the status that the system has to attention for handling the situation. It is defined with a label that represents the *triggering event name*;
- conditions have to be identified with in mind a list of symptoms or observations that the architecture can retrieve from the environment. Symptoms are perceptions or beliefs and they are used in the unification process and logical inference for validating a context;
- the procedures list has to contain actions, behaviors or other therapies (plans) that try to resolve the issue acting or interacting with the agent;
- a procedure can be a simple action or a more complex structure that involve the usage of other therapies;
- an alternative therapy can be added into the plan library keeping the same *triggering event name* and changing conditions for validating it. If conditions of the default therapy fail, an alternative therapy can be executed.

Table 3
Mapping table between therapy and plan. The table contains the one-to-one correspondence that we identified for letting remote deployment of therapies in robotic caregivers that use the proposed architecture.

Therapy	Plan
Diagnosis Identifier	Triggering Event
Condition to administer therapy	Context Pre-Conditions
Indications and Treatments	Actions, Sub-plans, Messages

For adding therapies, Dr. Haus has to choose a triggering event name used to add the therapy into the cognitive system. To do this, Dr. Haus uses the interface (Fig. 4) shown by the *Virtual Assistant* agent. Before, he has to add information useful for applying the therapy that works as conditions for administering the therapy.

In this case, Dr. Haus evaluates that one pill of paracetamol is a good therapy to cure Alice of the fever. Dr. Haus clicks on `Add Information` button and he writes:

```
alice; current_diagnosis; fever;
fever; paracetamol; 1;
```

Once aforementioned information has been added to the knowledge base, new beliefs are registered. The new beliefs are necessary for the new treatment. Now a new therapy can be added clicking to the namesake button in the interface. Dr. Haus choices a *triggering event name* that represent the health problem. Dr. Haus clicks on the `Add Therapy` button and assigns the name for this therapy through a textbox. He decides to call it `to_handle_fever`. A new screen is launched and Dr. Haus has to select which conditions (beliefs) must be satisfied to perform the therapy, next to Dr. Haus has to compose the list of indications to specify which behavior MyRob has to show to perform the therapy. The list of behaviors is shown and Dr. Haus indicates the order of action execution. The list of indications for the new therapy is translated in plan and the ordered list is made by Dr. Haus. It looks like:

```
to_handle_fever: doctor(D) & D=='Dr. Haus' &
patient(A) & A=='Alice' & current_diagnosis(F) & F=='fever' &
fever(P,Q)
 <-.check_bodytemperature(A);
.search_medicine(P);.prepare_medicine(P,Q);
.take_drug_to(A);.check_therapy_taken_by(A);
.signal_to_physician(D).
```

The physician can select actions that have been previously programmed for a specific robot.

Summarizing, Dr. Haus interacts with *Virtual Assistant* agent for inserting new therapies for Alice (Fig. 4), this agent translates therapies in plans and sends plans to the *Planner* agent. The *Planner* agent pushes new plans into its own Plan Library. The *Motion* agent makes sure MyRob can operate for administering the therapies.

5. Discussing the architecture quality and validation

The contribution of this paper lies in the multi-agent architecture defined for solving problems related to healthcare. In this section, we validate the architecture employing one of the mainly assessed methods for analyzing and validating software architectures: the SAAM, Software Architecture Analysis Method [44].

Validating complex systems, in terms of the related software architectures, has boosted the interest of scientists and engineers. The main goal of the evaluation process depends on how much software systems are capable of fulfilling its quality requirements and identifying risks [45–47]. So, what makes an architecture good is in how much it fits the goals and needs of the organization that is going to use it.

A recent survey [48] presents a comparative analysis of software architectures evaluation methods, their result is a taxonomy of evaluation methods. From this taxonomy, it arises that comparing and validating software architectures is a hard task due to the different languages and notation used for defining architectures. The method we choose provides a notation for describing the architecture, mainly highlighting its structural perspective. Also, this method starts from considering the goals underpinning the creation of the software systems and allows them to link them to some quality attributes. In so doing, we can focus on the quality concerns satisfying some software quality factors: maintainability, portability, modularity, reusability and robustness.

The activities reported in SAAM [44] for evaluating an architecture are:

1. *Characterize a canonical functional partitioning for the domain.*
2. *Map the functional partitioning onto the architecture's structural decomposition.*
3. *Choose a set of quality attributes with which to assess the architecture.*
4. *Choose a set of concrete tasks which test the desired quality attributes.*
5. *Evaluate the degree to which each architecture provides support for each task.*

The first activity implicitly led us to identify roles and responsibilities to assign to agents in the architecture in Fig. 1. Functional partitioning consists of the separation of concerns between managing the environment and managing knowledge for the decision process. In Fig. 5, we represent the result of converting our architecture using SAAM notation. In so doing, we highlight the computational entities and, data and control connections among them. SAAM then prescribes to choose some quality attributes. In our case, maintainability, portability, modularity and reusability are intrinsic in the nature of an agent architecture. The agent paradigm allows designing a software system with a high degree of modularity and a low level of coupling among components that guarantee these quality factors. Going into details of this topic is out of the scope of this paper. We selected *robustness* for validating the architecture and we also based on FURPS [49,50] requirements category. *Robustness* [51] refers to *the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environment conditions*.

We validated our architecture in the spirit of understanding how much it can cope with errors and unknown situations during execution. Indeed, our need is having software able to adapt to changes and developing an architecture supporting such a kind of software. To validate this aspect, we identified some tasks with which evaluating the level of support provided by the architecture, as prescribed in SAAM. Among the tasks we used for the analysis, we report the following: (*i*) recognizing new patient's state, (*ii*) adding new suitable plans and (*iii*) reusing previous successful plans.

For validating the architecture in supporting these tasks, we have to refer to its representation in the SAAM notation (Fig. 5). The SAAM notation has been created for separating the control flow from the data flow. As can be seen, we identified three basic processes whose thread of control is assured by the control flow and the data flow among the computational components, the active and passive data.

Recognizing new patient's state is a scenario in which guarantying the quality of robustness is fundamental. This scenario involves only two processes and one thread involving at the same time one control flow and one data flow. Inside the Deliberation Controller process, the responsibilities of computation are divided among four computational components. Each component is in charge of one specific monitoring function that is directly transferred to the System Controller process. Computational components may communicate with each other; communications are direct and not mediated by other components and processes. This point assures that a change in the environment is immediately caught and transferred to the related computational entity and then directly communicated to the knowledge controller. Each computational entity is atomic and quite reactive, so robustness and ability to intercept changes is comparable to a portion of object-oriented code devoted to catching an event.

Adding new suitable plans, this task is supported by a single computational component inside the Virtual Assistant Controller computational component. Only one control flow and one data flow are necessary. The communication with the Planner computational component, devoted to managing the new plan, is direct. In this case, since the Virtual Assistant provides the right interface for inserting a new plan, an error is highlighted during the insertion process and immediately communicated, avoiding the risk of failure or unexpected input propagation.

Reusing previous successful plans is ensured by the system. These plans are stored in the Working Memory and are available to be selected when the patient's conditions fit with the plan's preconditions. Each time that the patient's state changes, the Multi Agent System Controller computational module takes control for actuating a plan, interacting with the user if needed through the Virtual Assistant Controller. Later on, it sends data and control to the Deliberation Controller process. In this task, all computational components are involved, but mainly only two are relevant, the Multi Agent System and the Deliberation Controllers. Supporting this task is more demanding since there are two data and one control flow exchange. The process involved in the thread of control is of two different levels where one comprises the second. Dependencies between processes could create an architectural coupling that may slow the thread of control for supporting that task. For instance, the scheduled plan might depend on other processes' results and the overall outcome is afflicted by the computational time needed. In so doing the reaction to the change may not be responsive and other sudden events may invalidate the chosen plan, affecting the system's robustness. Hence, this situation may make all the system asynchronous towards changes. We are investigating an intelligent support system based on scene understanding for forecasting events to face the problem. However, the task is supported, and adaptation to changes is guaranteed.

We evaluated some other tasks that we do not report here because they are minor tasks, and the conclusion we drew was in the same line

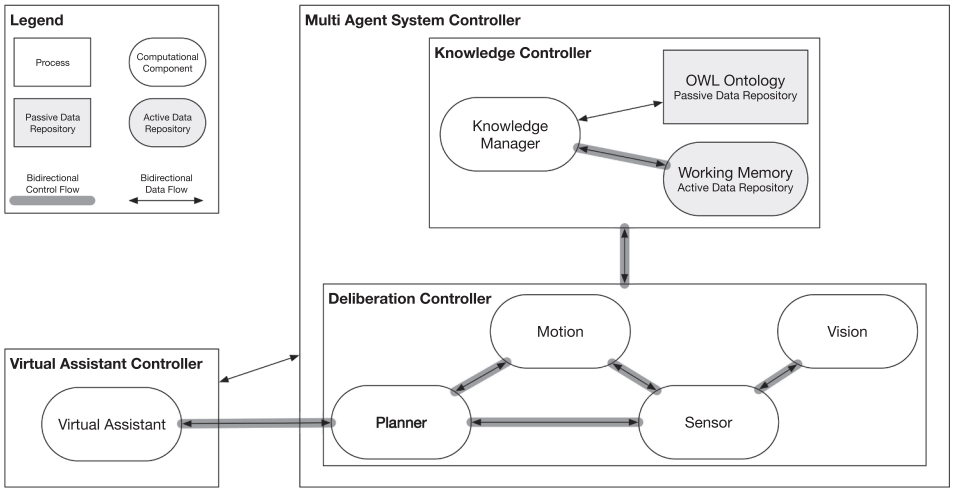

Fig. 5. A SAAM based representation of the architecture in Fig. 1.

as the previous ones.

From the validation process, we realized that the proposed agent architecture fits all the needs related to the functional requirements in Section 3. As said, architecture is not good or bad in general but in concerning some specific goals. In this case, SAAM revealed that the architecture is modular and decoupled enough for supporting in reacting to changing at runtime.

In the future, we will complement this validation with metrics and analysis results. We will inspect the flow of events and how they will be supported by the robotic platform. Now, this cannot be evaluated, because we tested modules independently from the operational scenario. Modules were tested onto the robot Pepper[6], simulating hypothetical input perceived from the environment.

6. Discussion and conclusion

From a state-of-the-art review, robots used in healthcare act mainly as teleoperators. We propose the use of robots and agent technology to provide the doctor with slightly more intelligent support than a simple teleoperator system. The robot does not make decisions but, programmed based on the agent architecture we propose, can interact with patients and doctors in a changing environment and alert the doctor or suggest strategies when necessary. The robot is endowed with the ability to handle unforeseen situations and to communicate and collaborate with the physician, thus providing him with intelligent support.

Using a robot produces several advantages, mainly having an avatar of the physician that acts on his behalf. Employing robots increases healthcare facilities' efficiency when the number of physicians is low, reduces the risk of infections due to pandemic such COVID-19 [52], because a robot is immune, speeds up the detection of new occurring situations.

The architecture we propose (shown in Fig. 1) resembles a cognitive model and it is implemented using the multi-agent approach. It has been inspired by our previous works in the Human-Robot Interaction (HRI) area [53,54].

The main contribution of this work is the creation of an intelligent bridge between the physician and the patient. It has been realized by fully exploiting the strength of BDI multi-agent systems.

The proposed architecture offers the advantage to split in the space all the system's functionalities. In this way, the overall system may be easily scalable and adaptable to any context. The agents, to be implemented based on the architecture, have been conceived at a higher level than the implementation one. Hence, thinking to their macro-level functionalities. It is for this reason that the combination of architecture plus agent system can be adapted and extended to other and more complicated clinical scenarios. For instance, specialized agents could support specialized physicians through intelligent software that uses a data-driven approach for building interfaces as in [55] or including techniques for data manipulation as in [56,57]. Moreover, everything related to monitoring can be deployed in various sensors to adopt IoT techniques for patient monitoring. In the next future, we are planning to realize a physical deployment on nursing homes to validate our system. We will evaluate trials using the experience of domain experts and metrics for software analysis on collected system results. Given our experience in the development and use of robotic systems, we claim that some disadvantages and limitations we might found so far are that the potential efficiency of the whole system clashes with technological problems related for example to the robot's abilities or the interaction of the robot with the human. In the future, we will fine-tune these last two aspects, especially the one related to interaction, including in this context elements of Human-Robot Teaming Interaction and the results that the Robotics Lab of Palermo has achieved so far in this field, such as the integration of the method proposed for learning plans [58] autonomously.

[6] https://www.softbankrobotics.com/emea/en/pepper.

Declaration of Competing Interest

The authors declared that there is no conflict of interest.

References

[1] World Health Organization, Healthy, prosperous lives for all: the European Health Equity Status Report (2019). URL <http://www.euro.who.int/en/publications/abstracts/health-equity-status-report-2019>.
[2] Policy Department for Economic, Scientific and Quality of Life Policies, Robots in healthcare: a solution or a problem? (2019). <https://www.europarl.europa.eu/RegData/etudes/IDAN/2019/638391/IPOL_IDA(2019)638391_EN.pdf>.
[3] M. Wooldridge, P. Ciancarini, Agent-Oriented Software Engineering: The State of the Art, Agent-Oriented Software Engineering: First International Workshop, AOSE 2000, Limerick, Ireland, June 10, 2000: Revised Papers.
[4] P. Stone, M. Veloso, Multiagent systems: a survey from a machine learning perspective, Autonomous Robots 8 (3) (2000) 345–383.
[5] J. Ota, Multi-agent robot systems as distributed autonomous systems, Adv. Eng. Inform. 20 (1) (2006) 59–70.
[6] J. Liu, J. Wu, Multiagent robotic systems, CRC Press, 2018.
[7] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović, G. Fortino, Agent-based internet of things: state-of-the-art and research challenges, Future Gener. Comput. Syst. 102 (2020) 1038–1053.
[8] C.-J. Su, C.-Y. Wu, Jade implemented mobile multi-agent based, distributed information platform for pervasive health care monitoring, Appl. Soft Comput. 11 (1) (2011) 315–325.
[9] N. Koceska, S. Koceski, P.B. Zobel, V. Trajkovik, N. Garcia, A telemedicine robot system for assisted and independent living, Sensors (Switzerland) 19 (4). doi:10.

3390/s19040834.
[10] D. Casey, So what have the roman's researchers ever done for us-mario, in: 15th Annual School of Nursing & Midwifery Conference, NUI Galway, 2015.
[11] G. D'Onofrio, O. James, D. Scancarlo, F. Ricciardi, K. Murphy, F. Giuliani, D. Casey, A. Greco, Managing active and healthy aging with use of caring service robots, 6th Forum Italiano dell'Ambient Assisted Living, 2015.
[12] D.R. Recupero, A. Gangemi, M. Mongiovi, S. Nolfi, A.G. Nuzzolese, V. Presutti, M. Raciti, T. Messervey, D. Casey, V. Dupourque, et al., Mario: Managing active and healthy aging with use of caring service robots, EU Project Networking at ESWC: Portoroz, Slovenia.
[13] D. Casey, H. Felzmann, G. Pegman, C. Kouroupetroglou, K. Murphy, A. Koumpis, S. Whelan, What people with dementia want: designing mario an acceptable robot companion, International conference on computers helping people with special needs, Springer, 2016, pp. 318–325.
[14] C. Salatino, L. Pigini, M.M.E. Van Kol, V. Gower, R. Andrich, G. Munaro, R. Rosso, A.P. Castellani, E. Farina, A robotic solution for assisting people with mci at home: Preliminary tests of the enrichme system, AAATE Conf. 2017, pp. 484–491.
[15] C. Salatino, V. Gower, M. Ghrissi, A. Tapus, K. Wieczorowska-Tobis, A. Suwalska, P. Barattini, R. Rosso, G. Munaro, N. Bellotto, H. van den Heuvel, The enrichme project, in: K. Miesenberger, C. Bühler, P. Penaz (Eds.), Computers Helping People with Special Needs, Springer International Publishing, Cham, 2016, pp. 326–334.
[16] F. Lunardini, M. Luperto, M. Romeo, J. Renoux, N. Basilico, A. Krpic, N.A. Borghese, S. Ferrante, The MOVECARE Project: Home-based Monitoring of Frailty, in: 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI) (2019) 1–4. doi:10.1109/bhi.2019.8834482.
[17] M. Luperto, J. Monroy, F.-A. Moreno, J.R. Ruiz-Sarmiento, N. Basilico, J. Gonzalez-Jimenez, N.A. Borghese, A multi-actor framework centered around an assistive mobile robot for elderly people living alone, in: Robots for Assisted, Living Workshop-International Conference on Intelligent Robots and Systems (IROS), 2018.
[18] M. Luperto, M. Romeo, F. Lunardini, N. Basilico, C. Abbate, R. Jones, A. Cangelosi, S. Ferrante, N. Borghese, Evaluating the acceptability of assistive robots for early detection of mild cognitive impairment, 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2019, pp. 1–8.
[19] L.P. Fried, C.M. Tangen, J. Walston, A.B. Newman, C. Hirsch, J. Gottdiener, T. Seeman, R. Tracy, W.J. Kop, G. Burke, et al., Frailty in older adults: evidence for a phenotype, J. Gerontol. Ser. A: Biol. Sci. Med. Sci. 56 (3) (2001) M146–M157.
[20] I. Papadopoulos, A. Sgorbissa, C. Koulouglioti, Caring robots are here to help, arXiv preprint arXiv:1803.11243.
[21] I. Papadopoulos, C. Koulouglioti, S. Ali, Views of nurses and other health and social care workers on the use of assistive humanoid and animal-like robots in health and social care: a scoping review, Contemp. Nurse 54 (4–5) (2018) 425–442.
[22] I. Papadopoulos, C. Koulouglioti, The influence of culture on attitudes towards humanoid and animal-like robots: an integrative review, J. Nurs. Scholarsh. 50 (6) (2018) 653–665.
[23] V.C. Pham, Y. Lim, H.-D. Bui, Y. Tan, N.Y. Chong, A. Sgorbissa, An experimental study on culturally competent robot for smart home environment, International Conference on Advanced Information Networking and Applications, Springer, 2020, pp. 369–380.
[24] F. Courreges, P. Vieyres, Advances in robotic tele-echography services-the otelo system, The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vol. 2 IEEE, 2004, pp. 5371–5374.
[25] R.-G. Lee, C.-C. Lai, S.-S. Chiang, H.-S. Liu, C.-C. Chen, G.-Y. Hsieh, Design and implementation of a mobile-care system over wireless sensor network for home healthcare applications, in: 2006 International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE, 2006, pp. 6004–6007.
[26] J.-C. Liau, C.-Y. Ho, Intelligence iot (internal of things) telemedicine health care space system for the elderly living alone, in: 2019 IEEE Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS), IEEE, 2019, pp. 13–14.
[27] N. Dao, X. Hai, L. Huu, T. Nam, N.T. Thinh, Remote healthcare for the elderly, patients by tele-presence robot, 2019 International Conference on System Science and Engineering (ICSSE), IEEE, 2019, pp. 506–510.
[28] M. Wooldridge, Reasoning about rational agents, MIT press, 2003.
[29] R.H. Bordini, J.F. Hübner, M. Wooldridge, Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology), John Wiley & Sons Inc, USA, 2007.
[30] A.S. Rao, M.P. Georgeff, et al., BDI agents: from theory to practice., in: ICMAS, Vol. 95, 1995, pp. 312–319.
[31] M.E. Bratman, Intention, plans, and practical reason Vol. 10 Harvard University Press Cambridge, MA, 1987.
[32] M.E. Bratman, What is intention, Intentions Commun. (1990) 15–31.
[33] R.H. Bordini, J.F. Hübner, BDI agent programming in AgentSpeak using Jason, International Workshop on Computational Logic in Multi-Agent Systems, Springer, 2005, pp. 143–164.
[34] A. Vuono, M. Luperto, J. Banfi, N. Basilico, N.A. Borghese, M. Sioutis, J. Renoux, A. Loufti, Seeking prevention of cognitive decline in elders via activity suggestion by a virtual caregiver, Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 1835–1837.
[35] M. Luperto, M. Romeo, F. Lunardini, N. Basilico, R. Jones, A. Cangelosi, S. Ferrante, N. Alberto Borghese, Digitalized cognitive assessment mediated by a virtual care-giver, IJCAI International Joint Conference on Artificial Intelligence, 2018, pp. 5841–5843.
[36] J. Ferber, G. Weiss, Multi-agent systems: an introduction to distributed artificial intelligence Vol. 1 Addison-Wesley, Reading, 1999.
[37] A.S. Rao, AgentSpeak (l): BDI agents speak out in a logical computable language, in: European workshop on modelling autonomous agents in a multi-agent world, Springer, 1996, pp. 42–55.
[38] G. Antoniou, F. Van Harmelen, Web ontology language: OWL, in: Handbook on ontologies, Springer, 2004, pp. 67–92.
[39] A. Chella, F. Lanza, V. Seidita, Representing and developing knowledge using JASON, CArtAgO and OWL, Vol. 2215, CEUR-WS, 2018, pp. 147–152.
[40] F. Endres, J. Hess, J. Sturm, D. Cremers, W. Burgard, 3-d mapping with an rgb-d camera, IEEE Trans. Robot. 30 (1) (2013) 177–187.
[41] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767.
[42] A. Chella, F. Lanza, V. Seidita, Human-agent interaction, the system level using JASON, in: Proceedings of the 6th International Workshop on Engineering Multi-Agent Systems (EMAS 2018). Stockholm, 2018.
[43] A. Chella, F. Lanza, V. Seidita, Decision process in human-agent interaction: Extending JASON reasoning cycle, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11375 LNAI (2019) 320–339. doi:10.1007/978-3-030-25693-7_17.
[44] R. Kazman, L. Bass, G. Abowd, M. Webb, Saam: A method for analyzing the properties of software architectures, Proceedings of 16th International Conference on Software Engineering, IEEE, 1994, pp. 81–90.
[45] N. Lassing, D. Rijsenbrij, H. Vliet, The goal of software architecture analysis: Confidence building or risk assessment, Proceedings of First BeNeLux conference on software architecture, 1999, pp. 47–57.
[46] L. Dobrica, E. Niemela, A survey on software architecture analysis methods, IEEE Trans. Software Eng. 28 (7) (2002) 638–653.
[47] M.A. Babar, L. Zhu, R. Jeffery, A framework for classifying and comparing software architecture evaluation methods, in: 2004 Australian Software Engineering Conference. Proceedings., IEEE, 2004, pp. 309–318.
[48] A. Patidar, U. Suman, A survey on software architecture evaluation methods, in: 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), IEEE, 2015, pp. 967–972.
[49] P. Eeles, Capturing architectural requirements, IBM Rational developer works.
[50] R. Janoščová, Evaluation of software quality, IMEA 2012 (2012) 24.
[51] B. Bruegge, A.H. Dutoit, Object-oriented software engineering: Using UML, Patterns and Java (2003).
[52] G.-Z. Yang, B.J. Nelson, R.R. Murphy, H. Choset, H. Christensen, S.H. Collins, P. Dario, K. Goldberg, K. Ikuta, N. Jacobstein, D. Kragic, R.H. Taylor, M. McNutt, Combating covid-19—the role of robotics in managing public health and infectious diseases, Science Robotics 5 (40). arXiv:https://robotics.sciencemag.org/content/5/40/eabb5589.full.pdf, doi:10.1126/scirobotics.abb5589. https://robotics.sciencemag.org/content/5/40/eabb5589.
[53] A. Chella, F. Lanza, A. Pipitone, V. Seidita, Human-robot teaming: Perspective on analysis and implementation issues Vol. 2352 CEUR-WS, 2019.
[54] A. Chella, F. Lanza, V. Seidita, A cognitive architecture for human-robot teaming interaction, Vol. 2418, CEUR-WS, 2019, pp. 82–89.
[55] O. Gambino, L. Rundo, V. Cannella, S. Vitabile, R. Pirrone, A framework for data-driven adaptive gui generation based on dicom, J. Biomed. Inform. 88 (2018) 37–52, https://doi.org/10.1016/j.jbi.2018.10.009.
[56] E. Ardizzone, O. Gambino, A. Genco, R. Pirrone, S. Sorce, Pervasive access to mri bias artifact suppression service on a grid, IEEE Trans. Inf Technol. Biomed. 13 (1) (2009) 87–93, https://doi.org/10.1109/TITB.2008.2007108.
[57] A. Stefano, S. Vitabile, G. Russo, M. Ippolito, F. Marletta, C. D'Arrigo, D. D'Urso, M. Sabini, O. Gambino, R. Pirrone, E. Ardizzone, M. Gilardi, An automatic method for metabolic evaluation of gamma knife treatments, Lecture Notes Comput. Sci. (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9279 (2015) 579–589, https://doi.org/10.1007/978-3-319-23231-7_52.
[58] F. Lanza, P. Hammer, V. Seidita, P. Wang, A. Chella, Agents in dynamic contexts, a system for learning plans, Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 823–825, , https://doi.org/10.1145/3341105.3374083.