Knowl Inf Syst
DOI 10.1007/s10115-013-0635-9

REGULAR PAPER

Context-aware multiagent system: Planning home care tasks

Juan A. Fraile · Yanira De Paz · Javier Bajo · Juan Francisco De Paz · Belén Pérez-Lancho

Received: 2 May 2012 / Revised: 21 September 2012 / Accepted: 16 March 2013

Abstract Context-aware systems are able to capture information from the context in which they are executed, assign a meaning to the gathered information, and change their behavior accordingly. As a result, the systems can offer services to users according to their individual situation within the context. This article analyzes the important aspects of context-aware computing such as capturing information for context attributes and determining the manner of interacting with users in the environment. Used in conjunction with mobile devices, context-aware systems are specifically used to improve the usability of applications and services. This article proposes the home care context-aware computing (HoCCAC) multiagent system that identifies and maintains a permanent fix on the location of patients in their home, and manages the infrastructure of services within their environment securely and reliably by processing and reasoning the data received. Based on the multiagent system, a prototype was developed to monitor patients in their home. The HoCCAC multiagent system uses a critical path method-based planning model that, in the present study, prepares the most optimal task-planning schedule for the patients in their home, is capable of reacting automatically when

J. A. Fraile
Pontifical University of Salamanca, c/ Compañía 5, 37002 Salamanca, Spain
e-mail: jafraileni@upsa.es

Y. De Paz · J. F. De Paz (✉) · B. Pérez-Lancho
Departamento de Informática y Automática, Facultad de Ciencias, University of Salamanca, Plaza de la Merced s/n, CP. 37008 Salamanca, Spain
e-mail: fcofds@usal.es

Y. De Paz
e-mail: yanira@usal.es

B. Pérez-Lancho
e-mail: lancho@usal.es

J. Bajo
Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus Montegancedo, Boadilla del Monte, 28660 Madrid, Spain
e-mail: jbajo@fi.upm.es

faced with dangerous or emergency situations, replanning any plans in progress and sending alert messages to the system. The results obtained with this prototype are presented in this article.

Keywords Context-aware computing · Home care · Multiagent system

1 Introduction

A search for software environments that better adapt to the demands of users and their environment takes us to context-aware systems. These systems store and analyze all of the relevant information that surrounds and forms part of the user environment. User preferences, tastes, location, state of mind, activity, surroundings, ambient temperature, lighting conditions, etc., constitute the information that can be classified from the onset as contextual information. Information contextual is any information that can be used to describe the situation of an entity [14]. As a result, a context not only stores information about the user, but also contains information about user preferences. However, these systems face the problem of managing information about the environment to provide an appropriate response with regard to its current state. These systems can be applied within different environments such as the case of home care for patients.

Context-aware systems provide mechanisms to develop applications that understand their context and are able to adapt to possible changes. A context-aware system uses its surroundings to modify its behavior to best satisfy the needs of its user. The information is usually obtained through sensors. Because of the large number of small and portable devices currently available, the most common form of displaying information to system users is to distribute it through various heterogeneous systems and information networks. These systems are incorporated into the user's daily life to the point of becoming so imperceptible that the users focus on the tasks that they need to perform without worrying about the tools they must use to do so. This integration is carried out in such a way that the inherent technology of these elements or objects does not interfere with the activities for which they are used, thus providing the most simple, useful, and comfortable use.

The previously described systems have wide-ranging applications. One of the environments in which they can be most useful is in home care. The high growth in the number of dependent persons and the advanced state of technological development has forced the need to generate new solutions for home care environments. Furthermore, recent commitments for satisfying the needs of this segment of the population indicate the need to modernize existing systems [1,11]. For this reason, home care environments require the use of context-aware systems. There are advanced applications that can be installed in the homes of dependent individuals in order to improve their quality of life. Home care requires the use of sensors, intelligent devices, and equipment to build a distributed environment in which basic home functions are automated. In this respect, multiagent systems can facilitate the development of home care environments. Multiagent systems [4] have been studied recently as monitoring system for the medical care [2] of persons who are ill or suffering from Alzheimer's [13]. These systems provide continuous support in the daily life of these individuals [13], predicting potentially dangerous situations, and managing the physical and cognitive support of the person being cared for [5].

This article presents the multiagent architecture home care context-aware computing (HoCCAC), which can supervise and monitor persons in specific contexts. The goal of HoCCAC is to facilitate the assistance of dependent users in their own home. The architecture provides a novel mechanism that integrates a task-planning model based on the critical path method (CPM) [3] into intelligent agents. The CPM method connects a series of related activities in the most optimal manner to reach a specific objective. By using the CPM task-planning method, the agents respond proactively within their environment. HoCCAC also uses information from the context-aware environment to predict user needs and provide effective solutions. HoCCAC incorporates case-based reasoning—belief, desire, intentions (CBR-BDI) agents [6] that can learn by building upon their initial knowledge base. The CBR-BDI agents interact autonomously with the environment and the system users to adapt to the needs of the environment. Incorporating the CPM task-planning method into the CBR-BDI agents enables the HoCCAC intelligent system to improve its planning and learning over time. HoCCAC has a model of context-aware data defined to optimize information management. The simple integration and interaction of intelligent agents, planning methods, sensors, devices, and a defined data model permits us to propose the HoCCAC architecture. HoCCAC is a novel system compared to other studies related to context aware [9,21,32]. These studies concentrate on collecting location data for the user. Other studies, such as [25], in addition to locating users within the context, attempt to improve the communication in a hospital center between patients and medical personnel by capturing context attributes such as time, the patient's state, or the user's role. However, the new services offered by HoCCAC allow for a closer, more natural, and implicit interaction with the user.

The remainder of the article is organized as follows: Sect. 2 presents the problem of context-aware computing and introduces the need for developing new systems that improve the living conditions of patients in their homes. Section 3 describes the proposed system, the interaction between agents and the HoCCAC system devices, the defined data model, the solutions proposed by the interpreter agent, the CPM method task planning, and a detailed design of the interpreter agent. Section 4 presents a case study in which the HoCCAC architecture was applied at the home of a patient with chronic pulmonary obstructive disease (COPD), with special emphasis on the HoCCAC context-aware capabilities. Finally, Sect. 5 presents the results, Sect. 6 the conclusions obtained from the home care scenario prototype and suggests future lines of work to improve the system and Sect. 7 a discussion.

2 Context-aware computing

Context-aware systems were first introduced by Want et al. [33] when they presented their active badge location system, generally regarded as the first context-aware application. It is a location system for individuals in an office environment, in which each person carries a badge that uses a sensor network to send signals containing information about each person's location to a centralized services area. Midway through the 90s, several tourist guide location-aware systems emerged [26,30], which provided information about the user's location. Location information is by far the most used attribute of context-aware systems. Recent years have seen considerable growth in the use of other attributes of context-aware information. It is difficult to describe the term *context aware*, and many researchers attempt to provide their own description and the relationship of attributes that are included in it. The term itself first appeared in printed form by Schilit and Theimer [29]. Some authors describe context aware as the location or identification of persons or objects [16,18]. These descriptions of the term *context aware* are often used in the initial stages of research in these systems. One of the most precise definitions was provided by Dey and Abowd [14]. These authors refer to context as any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the applications themselves.

There are various location-aware infrastructures that are capable of gathering positional data [9,21,32]. These systems include satellite global positioning system (GPS), mobile phone towers, proximity detectors, cameras, magnetic card readers, and barcode readers. These sensors can provide information about position or proximity and differ only in their precision. Some require a clear line of vision; other signals can penetrate walls, etc. The previously mentioned systems only use one context attribute: information on the position of the object or person. The use of different context attributes such as noise, light, and location makes it possible to combine context objects at a higher level. These elements are needed to construct systems that are more useful, adaptive, and easy to use. One example of this type of context-aware infrastructure is the system presented by Muñoz et al. [25], which improves communication by adding context awareness to the management of information within a hospital environment. Each user (in this case, doctors, nurses, etc.) is given a mobile device to write messages that are sent upon completion of a given set of circumstances. The context attributes included in this system are location, time, roles, and the state of the user or entity being analyzed.

Lim et al. [23] propose a project to create a context-aware home that utilizes multiagent system to monitor and execute appropriate actions based on the current state of the house. The multiagent system learns and adapts the movements and actions of the occupants and makes predictions. They propose an architecture where there is an agent in each room that interacts with the sensors and a superagent that makes decisions and deals with risk prediction. Finally, the system is tested in a simulated environment. The system proposes agents that interact with sensors, depending on the room where the sensors are located, and does not provide neither external services to the user and relatives nor a planning method. Another authors as Uhm et al. [31] focus on the semantics aspects of the context and propose a context model that separates the upper and lower layer according to the characteristic of each class using the web ontology language (OWL). This context model is used by a multiagent system (CAMAS) to provide the solution of policy problem using the classification of rule and implementation of a rule-based engine. The use of a semantic model and a rule-based engine limits the learning and adaptation abilities of the system. The system does not provide a planning method. Kaluza et al. [20] present a context-aware multiagent system for care of the elderly that combines sensor technologies to detect falls and other health problems, and calls for help in the case of an emergency or issues a warning in cases not needing urgent attention. The system focus on detecting alarm situations and does not provide a planning recommender system. Moreover, it does not provide external services to the user and relatives. As noted in the previous studies, the attributes used by the majority of context-aware systems are the location or situation of persons, objects, or entities. Few systems are able to use information taken from different context attributes and relate different types of data to interact with users or patients. Moreover, it is important to interact with the user and provide not only surveillance methods, but also planning and recommender adaptable systems. The goal of the present study is to move one step further and, in addition to using the different context attributes and behaviors in the proposed system, store and process different types of data gathered by the system in order to improve the quality of life of dependent persons in their home.

Using a context model as our basis, we propose the addition of the HoCCAC multiagent architecture, which offers context-aware services to users in specific contexts, and includes a set of independent services that gather and interpret context data. The fundamental characteristic of the system is to improve patient care through the processing and reasoning of the data provided by the context. The system can easily develop context-aware services and applications in a variety of contexts. The system is independent, as it can be applied to various types of hardware devices and operating systems, and is based on Java technology. Patients are authenticated and located within the environment by a radio frequency identification (RFID) chip with JavaCard technology that they carry. HoCCAC defines a light framework for the execution of service-oriented applications. The functions of this system include management of the installation, activation, deactivation, update, and elimination of services, as well as the authentication, control, and monitoring of users at all times.

3 HoCCAC multiagent system

Multigent systems are distributed organizations where the components (agents) collaborate to achieve a series of goals. Agent and multigent systems have become increasingly relevant during the last decades and have gained relevance in different areas [7,8]. The agents are autonomous entities that can be characterized through their capabilities: autonomy, proactivity, reactivity, social skills, organization, mobility, etc. [12,28]. These capabilities make the agents very appropriated to be applied in business intelligence and risk management scenarios. In these scenarios, the agents can play different roles and establish an organizational model where the human behaviors and the management processes can be emulated. In this sense, it is possible to obtain different agent types, specialized in concrete tasks and behaviors, that can collaborate together to increase the productivity of the business and provide an effective risk management. The agents can act as an interface between the human users and the systems, trying to provide advanced facilities and personalize the access to the system, but also can act as autonomous entities that are proactive and can make decisions independently.

HoCCAC is a distributed system composed of intelligent agents that can reason out and carry out optimal plans of action. The HoCCAC architecture is primarily focused on tracking, control, and notification. HoCCAC is defined by the need to control distributed devices and gather user information in context-aware environments in a manner that is both non-intrusive and automatic [19]. Furthermore, HoCCAC performs the handling, storage, and reasoning of context-aware information, as well as the security for managing the access and administration of information. The HoCCAC system uses sensors to receive information from its environment. It takes the context data from this information so that the intelligent agents can processes the information received and respond accordingly. With the data they are provided, the intelligent agents create or carry out plans of action, always opting to maximize or improve the expected result. HoCCAC also combines the management of personal data with a daily activity model defined by intelligent agents, using the data provided by the sensors installed in the household network. The interpretation and processing of both the knowledge base and the daily models, which is carried out by intelligent agents, provides an added value to the system. As a result, HoCCAC anticipates the requests or possible incidents that users may confront within their context and offers solutions based on past experiences that can improve the expected results.

In order to obtain and process context data and provide the user with solutions, the HoCCAC system employs a multiagent architecture that is composed of various types of intelligent agents, as shown in Fig. 1. These agents understand and respond to their environment. Additionally, HoCCAC provides the agents with mechanisms to access the system services via mobile devices. The multiagent architecture is comprised of the following types of agents:

– Provider agents capture the data obtained by both internal and external heterogeneous context sources so that the interpreter and database agents can process and reuse the

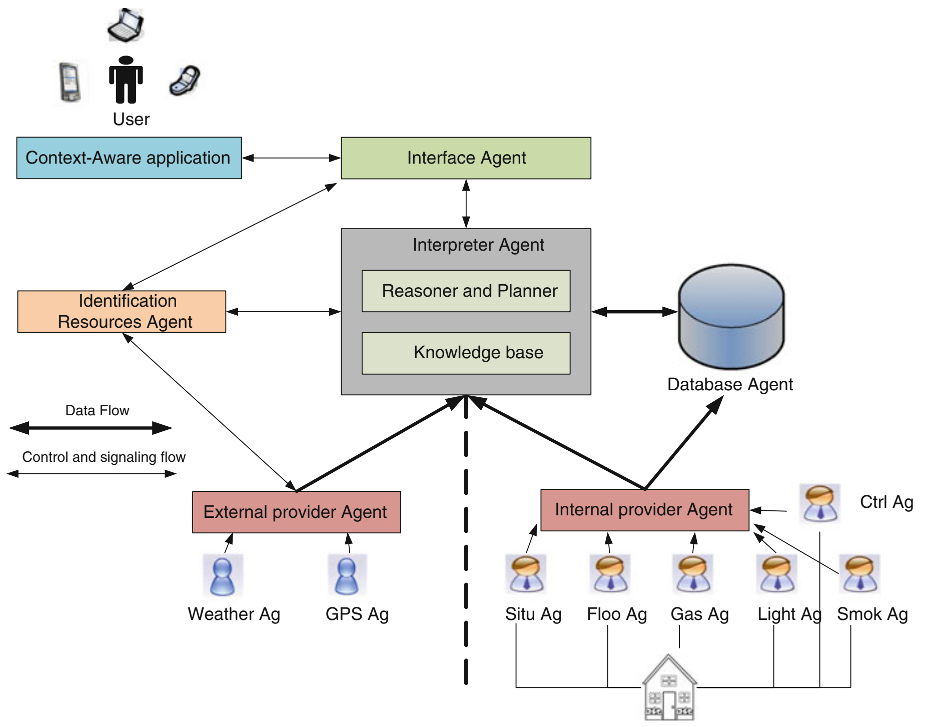


Fig. 1 Overview of the multigent system HoCCAC

data. The types of provider agents that can connect to the system are: (i) situation agents, which maintain a constant fix on the location of a patient in the home, (ii) flood detector agent, (iii) gas detector agent, (iv) smoke detector agent, (v) lighting detector agent, (vi) thermostat agent, (vii) control agent, which establishes and controls the daily parameters desired by the patient, (viii) weather agent, which registers the exterior temperature in order to respond accordingly to the internal variables of the home, (ix) GPS agent, which maintains the location of the patient outside the home. The system is dynamic and capable of incorporating an information provider agent at any moment by adding the corresponding sensor or gathering the necessary information from an external server or provider.

– Database agent stores the context-related data. This agent is responsible for managing the devices, the location of devices, and detailed information on the context in which it is located. It also manages old information related to the users that no longer interact with the environment and deactivated devices.
– The interpreter agent provides logical reasoning services to process the context information. The interpreter agent is based on the concept of case-based reasoning [12] to improve its autonomy and increase its ability to problem-solve. The sensory information is merged and translated to obtain the context information and updates the state of the patient. The interpreter agent uses the information available in the system, the signals received from the interface agent, and the knowledge base and beliefs related to the context in order to reason out the actions that it will execute. This creates a reasoning process that determines the optimal course of action or plans for the user to reach his or her objective. To this end, the interpreter agent uses the concept of CPM [3] to generate

plans as solutions. The interpreter agent provides the ability to reach complex, high-level objectives, and avoid errors that can lead to inefficiency. It also allows for greater flexibility when faced with new objectives.
- Resource identification agent (LcA). This agent is in charge of maintaining a record of the active provider agents in the system and of allowing or denying the inclusion of new provider agents. The LcA agent informs the interface and interpreter agents of any changes so that they can take into account any new context attributes that are provided, such as the inclusion of a new provider agent.
- Interface agent interacts with the interpreter and the LcA agent without the need for explicit user instructions. The interface agent reads the entries provided by the user through the context-aware applications and sends any modifications in behavior to the interpreter or LcA agent. The interface agent also notifies users with context-aware applications. The interface agent can, for example, receive many entries from context-aware applications over an extended period of time prior to deciding on any single action to take, or it can receive a single entry from a single context-aware application, which can initiate a series of actions by the agent.

HoCCAC allows the integration of context-aware applications. The context-aware applications represent all of the programs that can be used to take advantage of the system functionalities. HoCCAC makes it possible to easily use and share context-aware applications in a variety of physical spaces. The context-aware applications integrated in the HoCCAC architecture, also shown in Fig. 1, receive the information available in the system from the interface agent. They also use different levels of context information and adapt their behavior according to the active context. Context-aware applications are dynamic and adaptable to the context, continually reacting to changes in the state of the context. Context-aware applications can be executed locally or remotely, even from mobile devices with limited processing capability. One way to develop context-aware applications is to specify the actions that respond to context information under specific rules and conditions.

Figure 2 provides a general description of the HoCCAC distributed system infrastructure. The image shows how laptops and mobile devices connect to the system via Internet. Additionally, each of the devices is interconnected through wireless communication networks, mobile networks, or RFID technology. The HoCCAC system facilitates the integration and management of agents, devices, and control systems. Communication between platform agents follows the FIPA ACL standard (Foundation for Intelligent Physical Agents Agent Communication Language). The communication protocol between the agents and the services is based on the SOAP [13] standard (simple object access protocol). Agents can invoke two types of services: those that capture context information from the data obtained from autonomous components and those that respond to the automatic control systems installed within the context. All of this allows HoCCAC to be an easy system to implement within complex environments, without depending on a specific platform.

The following Sect. 3.1 provides a general description of the functions of the HoCCAC agents and their interaction with devices in their environment. Section 3.2 describes the system data model. Section 3.3 presents solutions provided by the interpreter agent. Section 3.4 describes the graphical representation of the task plans using the CPM method incorporated in the interpreter agent. Finally, Sect. 3.5 provides a detailed design of the interpreter agent.

3.1 Interaction between agents and devices in the system

The external provider agents obtain context information from external autonomous components such as, for example, a meteorological information server that provides weather infor-

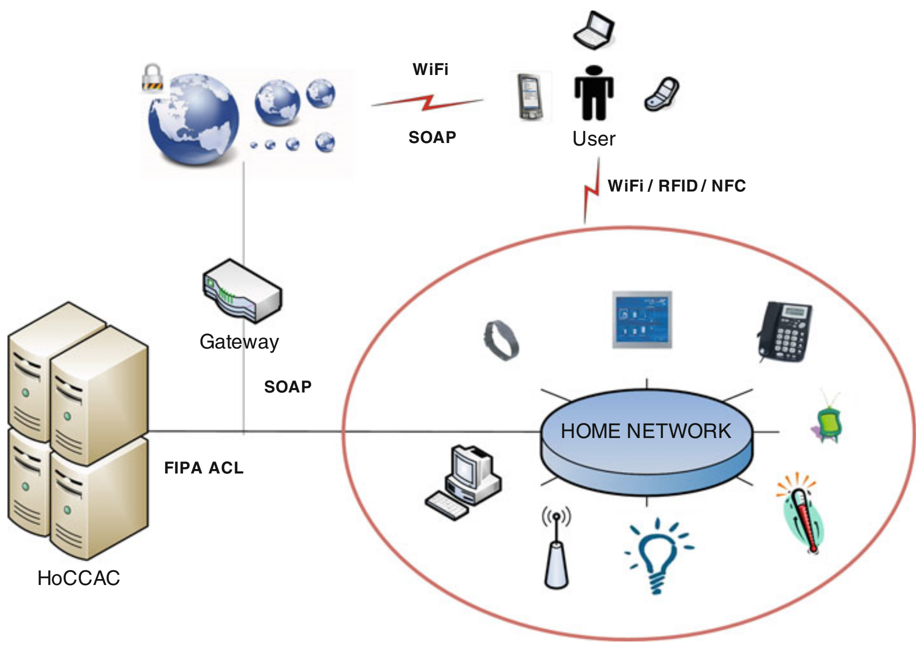


Fig. 2 Overview of the HoCCAC context-aware infrastructure

mation for a particular area. A location server can also provide information about the location of a person who is not at home. The internal provider agents gather information directly from the sensors installed in the environment, such as RFID location sensors installed in the home of a patient or light sensors. At the same time, all of the provider agents can interact with the user. The provider agents are some of the most important components for achieving appropriate interaction with users. This type of agent interprets the information sent from the agent platform and presents it to the users. The following example can be used to better understand this interaction: a location service receives constant information from RFID devices. This information is sent periodically to the agent platform. The information is interpreted and sent to a user's mobile device to find the position of the user within a specific area. The user's device does not require complex calculations to determine the position of the user since it needs only to interpret the data sent by the platform and display them to the user in a simple manner.

The functions of the interpreter agent are to process information provided by the database agent and to process the context information. The interpreter agent uses both the information it has processed and past experiences to develop plans of actions and initiate services that interact with the user and his or her context. For example, when the provider agents detect a new user in the context, they inform the other agents, then the interpreter agent processes the stored information for that user and determines the temperature and lighting desired by the user in the environment, based on past experiences and data.

The interpreter agent uses the context-aware applications to generate task plans and interacts with the sensors and automatic system controls installed in the environment. The context-aware applications use different levels of context information, as they are able to adapt their behavior to the context within which they are executed. Upon consulting the data registered in the information system by the LcA agent, these applications can locate the services from each of the providers it may be interested in. The context-aware applications can obtain context data by asking the interface agent or waiting for an event from the interface agent. The LcA allows users and agents to locate the different context applications. The primary characteristics of the LcA include scalability, adaptability, and multiple processing capabilities. The LcA controls large areas, located either in internal or external networks, where the context providers interact. Following the addition or elimination of physical sensors, or reconfigurations of the actual devices, the LcA searches for and adapts to changes in the environment. A new mechanism is also displayed to allow the context providers to inform the system of their functionality.

Figure 3 provides an example of monitoring a user in a context where the internal provider situation agent sends signals to the interpreter agent to inform on the situation of the user. The situation agent also receives signals from the interpreter agent to, for example, allow or prohibit user access to controlled areas. The LcA agent, on the other hand, records all of the services that the situation agent offers. The LcA agent continually updates the list of services provided by the situation agent which it can then transfer to the active applications in the context. Additionally, the database agent manages the context information according to the data provided by the interpreter agent.

3.2 Description of the data model

The primary goal of the HoCCAC system is to improve the living conditions for a user in his or her context. To this end, a set of devices and context-aware applications capture context information about the user and the user's routine. This information is gathered by the agents defined in the system, and the database agent ensures that the information related to the management of the context is stored in the system data model, as shown in Fig. 4.

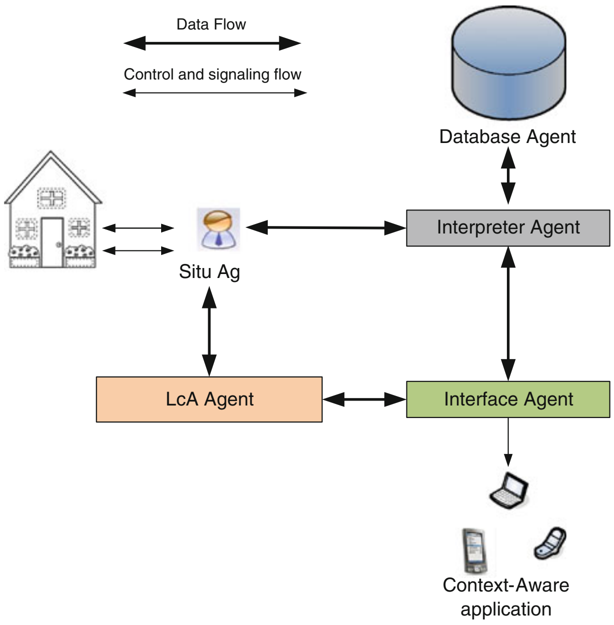


Fig. 3 Example of interaction between agents and context-aware applications

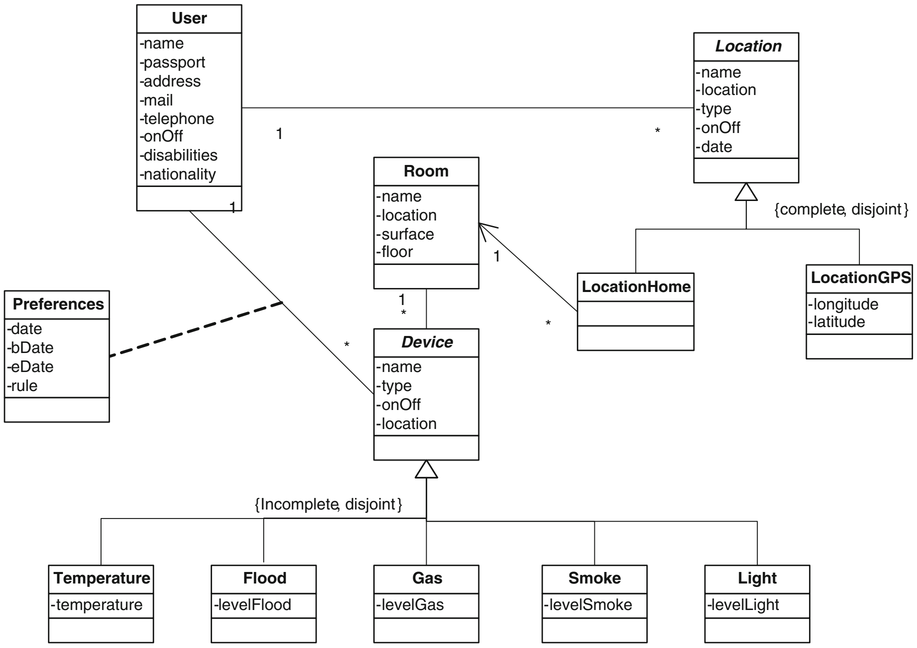


Fig. 4 Model of the HoCCAC system domain

As shown in Fig. 4, the domain model can store the preferences for various users on devices. The rules of action are specified in text format according to the attributes, methods, and operators specific to each device. The rules of interaction are interpreted using a parser, similar to how a decision tree is interpreted by data mining software such as Weka. The actions can be specified manually or generated automatically using a deliberative model. Based on the preferences and context information stored in the HoCCAC information system, the interpreter agent executes the available services to act on the context and the user. The interpreter agent stores information in its knowledge base that indicates the situation and current state of the user in the context in order to respond quickly to the user by using the devices closest to the user. In the event that the user is located outside the context-aware environment, the GPS agent will be responsible for recording the location of the user periodically using GPS coordinates. Additionally, one of the bedrooms, the kitchen, bathroom, and living room in the home will be equipped with smoke, gas, light, flood, and temperature detectors. Each of the values picked up by the sensors is gathered by the corresponding agent and sent to the interpreter agent, records stores the information.

The HoCCAC agents exchange information through ACL messages. For example, at a predetermined time, the interpreter agent receives an ACL message with the extensible markup language (XML) file that is shown in Fig. 5.

The information received by the interpreter agent in the XML file shown in Fig. 5 is stored in the system database by the database agent for its subsequent use and processed by the remaining system agents.

3.3 Solutions provided by the interpreter agent

The interpreter agent is integrated in the HoCCAC system. The goal of this agent is to provide the user with efficient solutions in execution time within the context of the user and

```xml
<room id="3">
  <temperature id="12">
    <attribs name="devicet01" temperature="22" type="dim" onOff="on" date="sysdate" location="25,46,85" />
  </temperature>
  <light id="7">
    <attribs name="devicel04" light="5" type="dim" onOff="on" date="sysdate" location="22,12,44" />
  </light>
  <smoke id="3">
    <attribs name="devices08" smoke="0" type="dim" onOff="on" date="sysdate" location="07,14,05" />
  </smoke>
  <gas id="6">
    <attribs name="deviceg02" gas="0" type="dim" onOff="on" date="sysdate" location="36,15,41" />
  </gas>
  <flood id="2">
    <attribs name="devicef06" flood="1" type="dim" onOff="on" date="sysdate" location="23,58,42" />
  </flood>
</room>
```

Fig. 5 XML file received by the interpreter agent

to improve his or her quality of life. The most important characteristics of the interpreter agent is: (i) reasoning capabilities to analyze and reason out context data collected by the system to develop task plans and provide proactive solutions, (ii) to adapt easily to the context within which it functions, and (iii) to collect sensor data and messages from other agents to improve plans of action. One example of the functionality of the interpreter agent applied to the user's preferred temperature conditions is when the system detects the presence of a user in a specific context using the RFID chip sensors that identify the user. The system has saved the user's temperature preferences within the context. It has also saved the case base for the user from other similar instances and also obtains the exterior ambient temperature. With these entries, the interpreter generates dynamic CBR-based action plans to stabilize the temperature in the environment while it detects the presence of the user. At the same time, the interpreter agent is also responsible for planning tasks for the user and, consequently, planning the activities that the user must carry out for the day. These action plans are based on the CPM method. The interpreter agent sends the action plans to the interface agent who then interacts with the temperature control device. The system thus maintains the temperature desired by the user within a specific context. In order to achieve the results desired by the user, the interpreter agent uses a CPM method to provide a CBR-based planning strategy.

CBR is a type of reasoning based on the use of past experiences to solve problems [6]. The goal of CBR systems is to solve new problems by adapting solutions that have been previously used to solve similar problems in the past. A case can be defined as a past experience and is composed of three elements: a description of the initial problem, a solution that provides a sequence of actions that must be carried out to solve the problem, and the final state that describes the end state once the solution has been applied. A CBR system generates cases, past experiences, to solve new problems. The way in which the cases are managed is known as the CBR cycle.

An example of a CBR cycle is shown in Fig. 6. The cycle includes 4 sequential phases: retrieve, reuse, revise, and retain. The retrieve phase begins when the description of a new problem is received. Similar algorithms are used in order to recover the cases with a problem description similar to the current problem. Once the majority of the similar cases have been retrieved, the reuse phase begins. In this phase, the solutions to the retrieved cases are adapted to obtain the best solution for the current case. The revise phase reviews the proposed solution. Finally, the retain phase allows the system to learn from the experiences obtained in the three previous phases and updates the data.

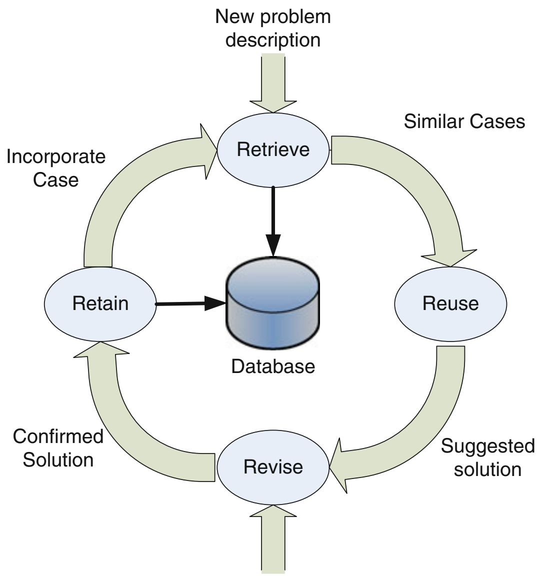


Fig. 6 Diagram including a CBR-BDI agent reasoning cycle

With regard to the CPM planning method, its primary goal is to determine the duration of a project or task plan, the latter being a sequence of interrelated activities, each of which has an estimated length of duration. The CPM method is widely applied in planning processes. CPM can control the execution of activities and determine which are the critical activities that must be carried out so that the global plan is not delayed [15,22]. CPM is applied in various fields such as manufacturing processes [17], industrial environments [15], task distribution tasks in grid computing [24], or information processing in distributed mode in hospitals [22]. With the CPM method, the duration of all activities is known, that is, there is no uncertainty. This simplifies the CPM method, making it easy to use. With the CPM method, the length of the task plan is the same as the critical path. This is the largest path for the set of project activities. The CPM method is applicable and useful in any situation in which it is necessary to carry out a set of interrelated activities to reach a specific objective. The true value of the technique in the CPM method increases when it is applied dynamically. In the event of unanticipated events or circumstances, the critical path method provides the ideal means to identify or analyze the need to reexamine or reschedule the project, reducing the adverse effects of the unanticipated events to a minimum. Similarly, when an opportunity arises to improve the project scheduling, the technique makes it possible to easily determine which activities should be accelerated in order to achieve the expected improvement. These scenarios are what make the CPM method the most adequate for the interpreter agent to implement in the HoCCAC system. The application of the CPM method by the interpreter agent allows the HoCCAC system to have the following information available: (i) which tasks should be carried out first, (ii) when to use system resources, (iii) how to schedule the use of devices, (iv) how to schedule the advancement of activities, (v) how many activities there are and which ones are carried out at any given time, (vi) the state of the current task plan in use with regard to the end date, (vii) which are the critical activities that, if delayed, will delay the length of the task plan, and (viii) which are the non-critical activities and their margin for delay; these are activities that can be delayed, without delaying the project, for a period of time referred to as a float. The CPM method essentially consists of two cycles:

- Scheduling and planning. In this cycle, the interpreter agent defines the task plan with all its activities or main components. The relationship between the activities is then established to decide which activity should begin first and which will follow. The relationship among the activities is used to design a network connecting the different activities based on their levels of priority. Subsequently, the estimated cost and time are defined for each activity, and the feasibility of the task plan is evaluated. Finally, the longest path for the project is identified, which will determine the length of the task plan, which in turn identifies the critical path.
- Execution and control. In this cycle, the interpreter agent approves the task plan, instructions are defined, advances are controlled, and finally corrective measures are implemented.

The following section describes the graphical representation of the task plans that the interpreter agent carries out with the CPM method.

3.4 Task planning with the CPM method

In order to develop the most optimal task plans, the interpreter agent receives a series of information from the HoCCAC system and follows a set of standards for the CPM method. A task plan is composed of a set of activities that must be carried out in a specific order to achieve an objective. A graphical representation of a task plan using the CPM method is referred to as a network. Each activity can include one or more tasks. The activities include events that are instances of the activity and act as control points. The events describe the moment that the activity begins or ends. The activities are represented by arrows whose length is not indicative in any way of the duration of the activity. The events are represented by circles. When building a network, it is necessary to keep in mind that each activity is represented by one and only one arrow and each activity should be identified by two nodes, as a single common node can have no more than two different activities.

Each arrow on the network represents an activity. Each activity is a task required by the project. A node represents an event, which is defined as the moment when the activities that arrive at that node finish. The points of the arrow initiate the sequence in which the events must take place. An event should precede the start of the activities that stem from that node. The node toward which all the activities lead is the event that corresponds to the conclusion of the task plan. The network can represent a task plan from its start or, if the task plan has already been initiated, it represents the plan for its completion. Thus, each node in the network represents the event that maintains the activity in progress or the event that initiates a new activity that can begin at any moment. Each arrow plays a dual role: represent an activity and assist in representing the relationship between the different priority activities. On occasion, an arrow may be necessary to define the priority relationships, even though there is no real activity to represent. In this case, a fictitious activity is represented by a dotted line, indicating the priority relationship. The fictitious or phantom activity does not consume time and achieves two objectives: (i) to save a priority relationship and (ii) to individualize each activity so that each one is identified by a unique pair of nodes. In order to create a network task plan, it is necessary to answer three basic questions about each arrow or specific activity:

- Which activities should be carried out immediately prior to executing a specific activity?
- Which activities should be carried out immediately after executing a specific activity?
- Which activities can be carried out while simultaneously executing a specific activity?

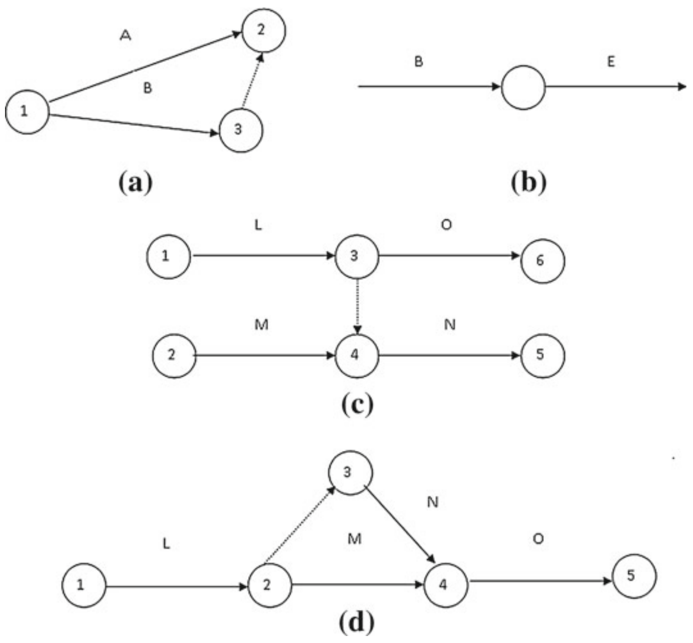


Fig. 7 **a** Concurrent and divergent activities, **b** connected activities, **c** representation of activities, and **d** fictitious activities on a network

A common rule for building these types of network task plans is that no two nodes can be directly connected by more than two arrows. The fictitious activities can also be used to apply this rule when they have two or more concurrent activities. Additionally, the node numbering allows for identifying the different activities using events with start "i" and finish "j." It is also possible to randomly number the nodes, as there is no reason that this cannot or should not be done. Nevertheless, experience has demonstrated that a systematic numbering of nodes simplify the arithmetic process. It is good practice to number the nodes in such a way that the number of the initial node for any arrow is always less than the number indicated in the node toward which the arrow is pointing. In other words, "i" should be less than "j." Nor should two different nodes be assigned the same number.

As for the activities, two activities are said to be simultaneous when than can be completely or partially carried out within the same interval of time without slowing each other down. Two activities are also said to be connected when the beginning of one of them is contingent on another the completion of another activity or activities, as seen in Fig. 7b with activities B and E. Additionally, concurrent activities are those ending in the same event, while divergent activities are those that begin in the same event, as seen in Fig. 7a with activities A and B. Another representative example can be seen in Fig. 7c whereby in order to carry out activity N, it is first necessary to finish activities L and M; and to initiate activity O, it is only necessary to finish activity L. Figure 7d also illustrates a network that includes fictitious activities, since in order to carry out activities M and N, it is necessary for activity L to have finished, and to carry out activity O, it is necessary to have completed activities M and N.

To date, the only existing restriction for correctly developing the network is to establish a logical sequence of activities. To perform a sequence of activities, the interpreter agent takes into account the information contained in the HoCCAC system and its reasoning. Once the network has carried out the activities, a corresponding duration should be assigned to each one. This way, the interpreter agent can calculate the total duration of the project and determine the early completion dates for each activity. To calculate these times, the interpreter agent must take the following into account:

- When task plan begins.
- No activity is to be initiated without having previously completed all tasks whose execution is contingent on that activity.
- Each activity should be carried out as soon as possible.
- Once initiated, each activity is executed without interruption until its completion.

Just as it is possible to calculate early start dates or times for initiating and completing an activity, the interpreter agent can also calculate late start dates or times for each activity for the total duration of the project. Calculating these times is quite simple: the earliest an activity can begin is the closest time or date that all of its preceding activities can finish. The earliest an activity can finish is the earliest required start date added to the time required for its completion.

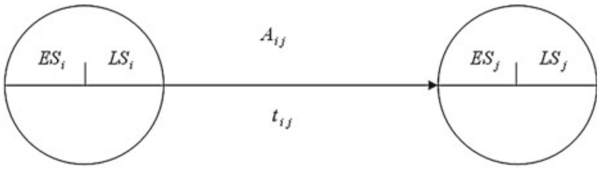
The first calculation made by the interpreter agent is of the early start times for each activity, as graphically represented in Fig. 8. The procedure is as follows:

1. Assign a start date and time ES_i to the first event "i" of the first network activity.
2. Take the sum of the duration of each of the activities that precede event "j" and indicate the total as ES_j. ES_j is also the earliest start time for event "j" and can be defined as:

$$ES_j = \max_i \left\{ES_i + t_{ij}\right\} \quad with \quad i < j \tag{1}$$

3. If two or more activities have the same final event, the maximum value obtained from the calculations performed in step 2 should be considered.
4. Steps 2 and 3 are repeated until the earliest time for carrying out ES_i is determined for all activities.
5. The final sum of the earliest start times constitutes the time required to carry out the project.

The second calculation made by the interpreter agent is for the late end times, as illustrated in Fig. 8. The procedure is as follows:



A_{ij}: Activity that begins with event "i" and ends with event "j"
ES_i: Earliest start time for event "i"
LS_i: Latest start time for event "i"
t_{ij}: Duration of activity (i,j) where i<j
ES_j: Earliest start time for event "j"
LS_j: Latest start time for event "j"

Fig. 8 Representation of the nodes for an activity

1. The late end time for the last event "j" is equal to its earliest start date ES_j. The total duration of the project is used as the initial data, which is noted to the far right of the final event as LS_j.
2. The duration of each of the activities that finish in event "j" is subtracted from LS_j. The subtracted values are the late end times LS_i and can be defined as:

$$LS_i = \min_j \left\{ LS_j - t_{ij} \right\} \quad with\ i < j \tag{2}$$

3. When two or more activities have the same starting event, the minimum value obtained from the calculations in step 2 should be considered. This minimum value is the late end time LS_i for the previous activities.

The interpreter agent also calculates the margin for delay TF (total float) for an event such as the difference between the latest time and the earliest time to begin or end an activity. This is defined as:

$$TF_{ij} = LS_j - ES_i - t_{ij} = LS_i - ES_i \quad \forall_{ij} \tag{3}$$

The margin for delay of an event indicates the maximum delay allowed to complete the event without delaying the completion of the task plan. The margin for delay of an activity indicates the same with respect to a delay in the completion of that activity. The activities with a zero margin for delay are critical since any delay will postpone completion of the task plan. If an activity A is critical, it is designated as:

$$\underset{CritialActivity}{{}^{*}A_{ij}} \rightarrow \underset{Criticaltime}{{}^{*}t_{ij}} \tag{4}$$

The critical path for a task plan is a path crossing the network in such a way that all of its activities have a zero margin for delay. It is the longest path in the network and corresponds to the minimum time for finalizing the task plan. The total duration of the task plan (D) is the sum of the critical times of the critical activities and is represented as:

$$D = \sum {}^{*}t_{ij} \tag{5}$$

Critical paths are also characterized by a series of properties. A task plan always has a critical path, often more than one. All of the activities that have a zero margin for delay should be on a critical path, while no activity with a margin for delay greater than zero can be on a critical path. All of the events with a zero margin for delay should be on a critical path, while no event with a margin for delay greater than zero can be on a critical path. A path on a network in which the initial and final events have a zero margin for delay is not necessarily critical, because it may be that one or more activities along that path can have a margin for delay greater than zero.

All of the rules and procedures previously described are contained in the interpreter agent, specifically in a module referred to as CPM context-aware plans, as explained in the next point and as shown in Fig. 9.

3.5 Interpreter agent design

As previously described in Sect. 3.3, CBR is a paradigm based on the idea that similar problems have similar solutions. The agents that are designed and implemented using CBR systems have autonomous reasoning skills and adapt to changes in the environment. As such, the interpreter agent fulfills two of its most important characteristics, as noted in the previous section: (i) reasoning capabilities and (ii) ease in re-planning to adapt to the context. The

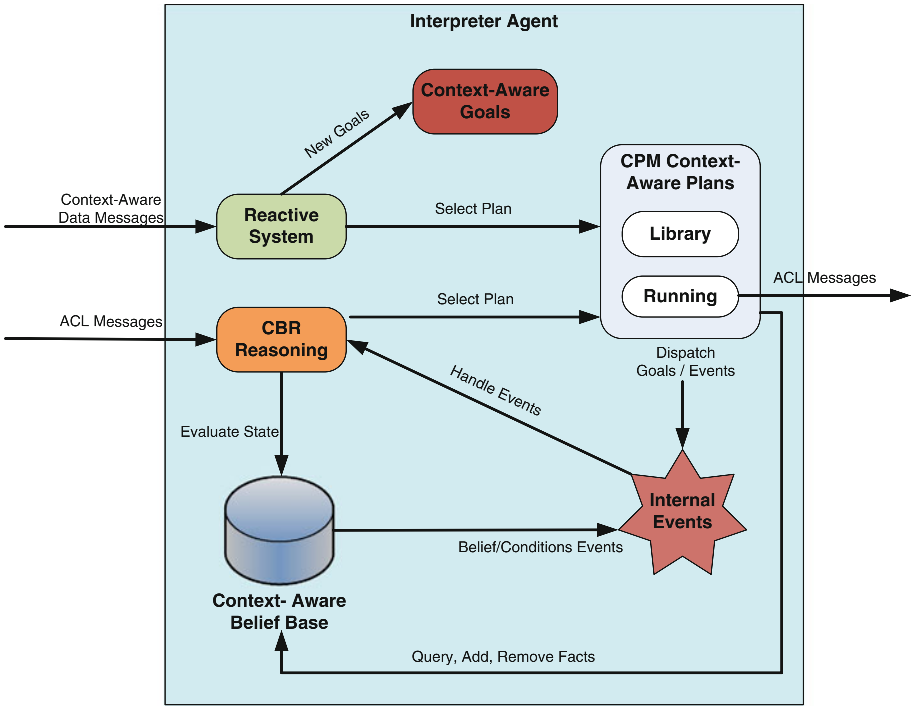


Fig. 9 overview of the interpreter agent architecture

other important characteristic is the ability to collect information from the environment using sensors and other agents. For this reason, when designing and implementing an interpreter agent, it is necessary to take into account the exchanges of information among the system agents. FIPA[1] specifications are currently considered the accepted standard for communication between agents. The interpreter agent is designed with the agent unified modeling language (AUML[2]) methodology, which provides mechanisms to obtain a design detailed enough to greatly facilitate the implementation phase.

On the other hand, the BDI [18] model is a solid base for modeling and applying the internal behavior of the agents. With the BDI model, it is possible to view an agent as an entity searching for an objective and behaving rationally. CBR systems and BDI agents can interact if the cases are implemented as beliefs, intentions, and desires that lead to the resolution of a problem. With the CBR-BDI agent, each state is equivalent to a belief. Furthermore, the objective to reach can also be a belief. The intentions will be plans that contain a specified set of actions that the agent must carry out in order to reach its objectives [6]. The agent can move from one state to another upon executing the action or task. A desire will be one of the final states reached in the past. To produce a proper transition between the design phase and the implementation phase of the CBR-BDI paradigm, the agent must be supported in the implementation phase.

The Java Agent Development Framework (JADE) platform [34] is a good option for developing agent-based applications. JADE agents follow the BDI model, and the JADE

1 www.fipa.org.

2 www.auml.org.

platform can be used to implement CBR-BDI agents. JADE also complies with the FIPA standard for operating with intelligent multiagent systems. The JADE platform focuses on applying the FIPA reference model, providing the required communication infrastructure and a services (e.g., management agent) platform, as well as a set of tools for developing and debugging CBR-BDI agents. JADE extension (Jadex) [27] is an implementation of a hybrid agent architecture (reactive and deliberative) to represent the various states of the JADE agents that are following the BDI model. Jadex is designed to be easily integrated within JADE, with the addition of a packet. The primary goal is to facilitate the use of reasoning and planning concepts during implementation.

Viewed externally, the interpreter agent is a black box that receives and sends messages. This section will now attempt to provide greater detail regarding the functionality of the interpreter via the use of a graphical representation. The interpreter agent, as previously described, will be implemented as a Jadex agent. In order to implement the interpreter agent as a Jadex agent, some variations are introduced into the Jadex architecture, as shown in Fig. 9.

Figure 9 provides a summary of the interpreter agent architecture. The incoming messages, as well as the internal events and new objectives, serve as a starting point for both the internal reactions and deliberative and reasoning mechanisms of the interpreter agent. The primary novelty in the design of the interpreter agent, as seen in Fig. 9, is that it integrates a CBR reasoning engine and a reactive system that collects data from sensors and the control systems. This provides a unique attribute in the design of the agent with regard to its conception and reasoning capabilities. Based on the results of the CBR reasoning engine, the interpreter agent develops task plans using the CPM context-aware plans, taking into account the standards and procedures described in point 3.4. These plans can be executed immediately as events, or they can be stored in the library of context-aware plans to generate new plans to be executed at a future time. Executing the plans can modify the context-aware beliefs base, send message to other agents, create new context-aware objectives, or produce future internal events.

The interpreter agent has a context-aware belief base, as shown in Fig. 9, in which it stores the beliefs that constitute its knowledge base. These beliefs are related to the context-aware environment and to the user. These beliefs include the location of the user, the exterior temperature and that of the home and its rooms, or the lighting and smoke levels in the different rooms of the home. The beliefs are structured through Java objects that represent the beliefs, as seen in Fig. 4. These objects have a name and attributes that have simple or multiple values. The knowledge base also incorporates the concept of databases oriented to objects. Object constraint language[3] (OCL) can retrieve subsets of context-aware beliefs. Another special characteristic of the context-aware belief base are the conditions, which represent an expression of a particular state, for example, or of one or various beliefs. Once the condition is satisfied, an internal event is generated. This event can activate a plan or allow the adoption of new objectives. With the interpreter agent, beliefs represent changes in the state of the sensors installed in the context-aware environment. This makes it easy to add new types of sensors that assist in the daily tasks of the user, and for the task plans to add new states for a sensor at a future time. The interpreter agent also includes specific plans for collecting data from the environment and the control systems. Furthermore, the task plans use OCL to inquire on the beliefs that meet certain conditions. All of the task and action plans specific to the interpreter agent contribute toward reaching the final objective.

The interpreter agent can define three types of objectives: reaching an objective, maintaining an objective, and carrying out an objective. To reach an objective, it is necessary to

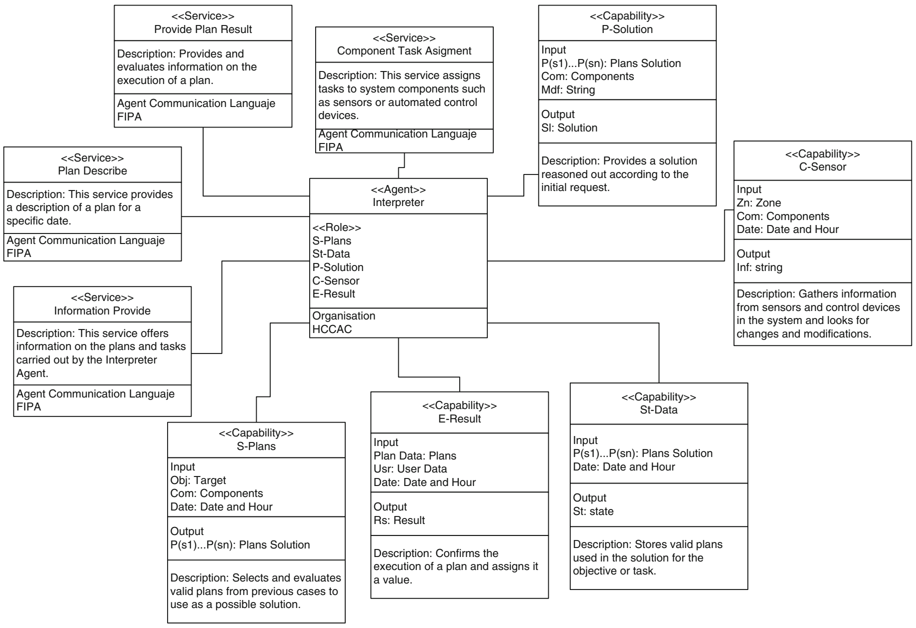
[3] http://www.omg.org/technology/documents/formal/ocl.htm.

have previously defined the state to be reached, but not the specific way to reach it. In this case, the interpreter agent has several alternatives for reaching the objective. To maintain an objective, the interpreter agent must monitor the state and execute the plans to re-establish the state when necessary. To carry out an objective, the interpreter agent specifies the actions to be executed. The objectives of the interpreter agent can also be represented by objects with various attributes. The plans for achieving the objective are explicitly stated through the conditions (e.g., using the beliefs). The name and properties of the objective facilitate the selection of a plan to initiate, and the parameters guide the tasks of the plans being executed. When an objective is carried out, the plan to execute is directly defined. The reasoning ability of the interpreter agent can be implemented in one of two ways. One is through the activation and deactivation of conditions. In this case, the interpreter agent applies rules according to the desired focus for activating or deactivating objectives that meet certain internal conditions, such as those conditions defined according to the knowledge base and user preferences. However, the objectives can also be activated and deactivated manually using task plans. The interpreter agent maintains an objective (e.g., maintaining the desired temperature in a room), carries out an objective (e.g., generating daily task plans for a specific user), and reaches an objective when, for example, it activates the gas extractor prior to activating the smoke alarm.

The interpreter agent has a library of context-aware plans based on the CPM method. With the library of plans, it can generate solutions based on action plans that have been initiated in the past. These action plans can interact with the devices installed in the system to facilitate the user's daily tasks, thus making the user's stay in the context-aware environment more comfortable. One part of the implementation process of the interpreter agent divides its functionality into separate plans composed of actions that are implemented in Java classes. We can say, then, that object orientation techniques are also used when carrying out the plans. Additionally, the functionality that is implemented in Java classes can be incorporated into other similar or hereditary systems. The following section then proposes a low level AUML design for the interpreter agent followed by the implementation using Jadex. The AUML design provides a class diagram for the interpreter agent, as shown in Fig. 10. It involves the most important agent within the HoCCAC architecture. This agent, as shown in Fig. 10, has five capabilities and four services. The capabilities are: (i) P-Solution, (ii) C-Sensor, (iii) S-Plans, (iv) St-Data y, and (v) E-Result. The services are: (i) Information Provide, (ii) Plan Describe, (iii) Result Plan Provide, and (iv) Component Task Assignment.

The execution of the interpreter agent using the Jadex model is based on events. Anything that occurs within the interpreter agent is represented as an event. The "message events" indicate when an ACL message has been received. The "objective events" announce when an objective has been reached, and the internal events inform, for example, of any changes in beliefs, timeouts, or conditions that have been met. To create and initiate an interpreter agent, the system needs to know which of the interpreter agent's properties will be instantiated. The state of the interpreter agent is determined by the beliefs, objectives, plans of execution, and library of plans. All of these concepts (beliefs, objectives, plans, conditions, filters) are defined in the agent definition file (ADF). The ADF applies a Java-based declaration of objects to define the initial beliefs and objectives. The plans are stated by specifying the way to create an instance in the Java class. The ADF file format is a file of properties with pairs of name/value that map out references to object statements. Figure 11 shows an excerpt of the definition for the interpreter agent. The next section describes the execution model that the interpreter agent follows.

The model for executing the interpreter agent is based on the four behaviors of the JADE framework, and on the context-aware data collector, a new behavior that is added to the architecture for the interpreter agent. These behaviors, which are executed within the Jadex-



<<Service>>
Provide Plan Result
Description: Provides and evaluates information on the execution of a plan.
Agent Communication Languaje FIPA
<<Service>>
Component Task Asigment
Description: This service assigns tasks to system components such as sensors or automated control devices.
Agent Communication Languaje FIPA
<<Capability>>
P-Solution
Input
P(s1)...P(sn): Plans Solution
Com: Components
Mdf: String
Output
Sl: Solution
Description: Provides a solution reasoned out according to the initial request.
<<Service>>
Plan Describe
Description: This service provides a description of a plan for a specific date.
Agent Communication Languaje FIPA
<<Agent>>
Interpreter
<<Role>>
S-Plans
St-Data
P-Solution
C-Sensor
E-Result
Organisation
HCCAC
<<Capability>>
C-Sensor
Input
Zn: Zone
Com: Components
Date: Date and Hour
Output
Inf: string
Description: Gathers information from sensors and control devices in the system and looks for changes and modifications.
<<Service>>
Information Provide
Description: This service offers information on the plans and tasks carried out by the Interpreter Agent.
Agent Communication Languaje FIPA
<<Capability>>
S-Plans
Input
Obj: Target
Com: Components
Date: Date and Hour
Output
P(s1)...P(sn): Plans Solution
Description: Selects and evaluates valid plans from previous cases to use as a possible solution.
<<Capability>>
E-Result
Input
Plan Data: Plans
Usr: User Data
Date: Date and Hour
Output
Rs: Result
Description: Confirms the execution of a plan and assigns it a value.
<<Capability>>
St-Data
Input
P(s1)...P(sn): Plans Solution
Date: Date and Hour
Output
St: state
Description: Stores valid plans used in the solution for the objective or task.

Fig. 10 Diagram of classes for the interpreter agent

Interpreter Agent

```
#start option
name                   = InterpreterAgent
class                  = jadexBDIAgent
#define plans
Instant_plans          = openS, closeS
plans                  = tskplan, tskadd, clean
#define plan mappings
openS                  = OpenSensorRoom()
closeS                 = CloseSensorRoom()
tskplan                = TaskPlan()
tskadd                 = TaskAdd()
clean                  = CleanTaskPlan()
#define activation filters for passive plans
tskplan_filter         = TaskPlan.getEventFilter()
tskadd_filter          = TaskAdd.getEventFilter()
#define initial beliefs and beliefsets
beliefsets             = tskactionsensors
tskaction              = {(idsensor, action, type)}
#define goals
goals                  = hotup, colddown, levelSmoke
#define goal mappings
hotup                  = Goal.createMantainGoal(hotup, temperature==23, hot)
colddown               = Goal.createMantainGoal(colddown, temperature==21, cold)
levelSmoke             = Goal.createPerformGoal(levelSmoke, ratioSmoke==5, low)
```

Fig. 11 Example of the definition of an interpreter agent

based interpreter agent, can be seen in Fig. 12 and are denominated as follows: the scheduler, the dispatcher, the message receiver, the context-aware data collector, and the timing. These behaviors are executed concurrently inside the structure of the agent. The proper functioning and availability of the context-aware data collector is fundamental for carrying out the tasks related to the interpreter agent's remaining behaviors. The context-aware data collector is

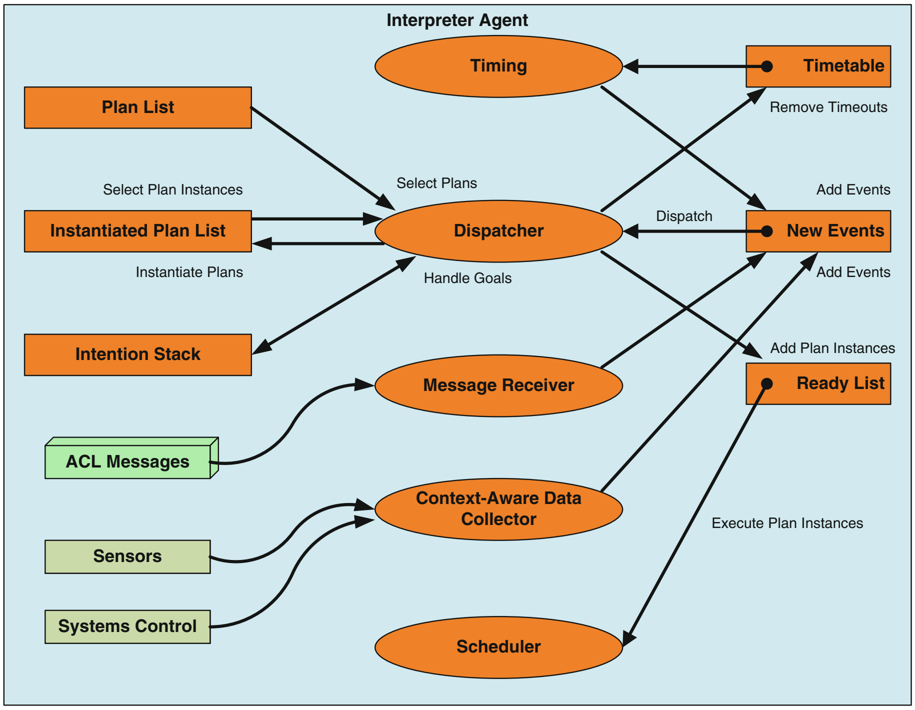


Fig. 12 definition example of interpreter agent

in charge of collecting the sensor and control system data, and generates corresponding notifications and events derived from the collected information. The behavior of the message receiver and the timing process is very simple to add new events to the event list. The message receiver listens to the ACL messages sent by other agents and generates corresponding "event messages." The timing eliminates the planned events once they have been initiated and adds new events to the list for their initiation. The dispatcher is responsible for adopting the objectives, queuing them for execution, and selecting the plans that manage the events needed to fulfill the objective. The selected plans are executed step by step by the scheduler, which also manages the supervision plan. These behaviors shut down on their own and restart themselves to keep the agent from consuming resources unnecessarily. Implementing the interpreter agent's functionalities as separate behaviors provides a clean design and allows for the flexible substitution of the behaviors with personalized implementations such as alternative planning mechanisms and CBR-BDI implementations that can be carried out using modified versions of the behaviors.

4 Applying HoCCAC to plan task the COPD patient in context-aware environment

For this case study, the HoCCAC system was used to develop a prototype for improving a patient's quality of life. The system collects information from the sensors, which capture data and interact with the patient by means of task plans developed by the interpreter agent. The primary information gathered by the installed sensors is the location-aware information for the user in the environment, using Wi-Fi, RFID, and ZigBee technologies. The system also

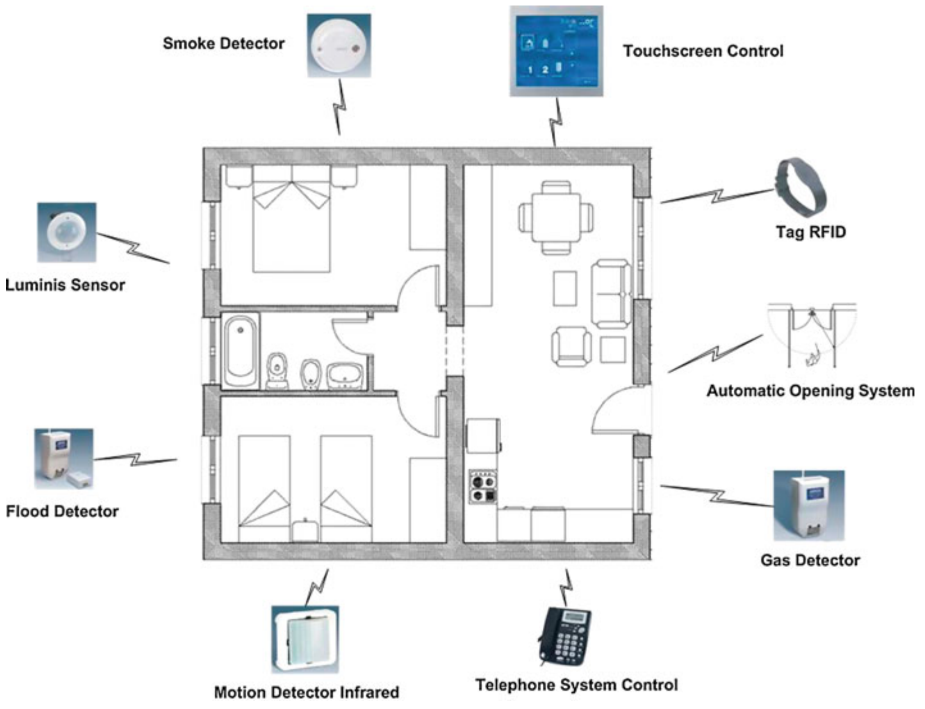


Fig. 13 Home plane

gathers information regarding the temperature throughout the patient's home and the lighting conditions in the patient's ambulatory area using ZigBee technologies, and a sensor to weigh the pill container. For this prototype, HoCCAC is based on the information collected within the environment to plan the tasks for a COPD patient at home and improve the current living conditions. The COPD patient requires artificial oxygen, which is administrated by means of a special machine shown in Fig. 14.

HoCCAC was employed to develop a prototype multigent system aimed at enhancing the assistance and care for low dependence patients at their homes. The house is 65 m^2 with a single dependent occupant. As shown in Fig. 13, the home is installed with (i) Simon Vit@ 81800-30 passive infrared motion detectors for roof and (ii) mechanisms for automatic door opening. The detectors' movements and mechanisms for opening doors interact with the Java Card and RFID microchip [21] and with users to offer services in run time. Each dependent user is identified by a (iii) Sokymat ID bracelet Band Unique Q5 equipped with an antenna and a RFID chip-Java-Crypto-Card with 32 K Module and Crypto-CoProzessor (1,024 bit RSA) compatible to the SUN JavaCard 2.1.1 [10]. The home also includes (iv) Simon Vit@81915-38 light sensors that are used in an effort to manage lighting levels in the home by maintaining the levels within a predetermined range of changeable values, (v) a Simon Vit@ 81221-38 TFT surface screen that displays and manages the primary system components, (vi) Simon Vit@ 81860-39 flood detectors for detecting water presence, (vii) Simon Vit@ 81861-39 gas detector designed to detect the presence of toxic gases and explosives, (viii) Simon Vit@ 81915-38 optical smoke detector for early warning of fires in progress, and a telephone control system. The sensors or actuators are placed in strategic locations throughout the home. All of the devices are controlled by agents. This sensor network is responsible for generating

Fig. 14 Machine to administrate oxygen

alarms upon comparing the user's current state with the parameters of the user's daily routine, which have been stored by the system. The system can generate alarms if it determines that the parameters fall outside the normal range. For example, if the user, on a non-working day, stands up prior to a certain hour, if the user spends more time than specified at the front door of the home without entering, if the user remains motionless in the hallway for an extended period of time, etc (Fig. 14).

In our case study, the COPD patient interacts with the system in such a way that the sensors capture the context-aware information about temperature, medication, light, and breathe. COPD is a pulmonary disease characterized by a reduction in the breathe capacity and mainly caused by the tobacco. COPD patients are chronic patients, but a personalized care plan can notably contribute to improve their quality of life. This plan must include strict medication and a fixed schedule for oxygen administration, and depends on the context conditions (temperature, humidity, etc.). The HoCCAC architecture provides context-aware technologies to obtain information about temperature, humidity, light, medication. Moreover, it provides a series of services that can be used by the patient. In this sense, HoCCAC:

- Facilitates mechanisms to access information via mobile devices (mainly for caregivers and relatives).
- Provides the medical staff with a new tool to communicate with the patients.
- Facilitates a tool to create adaptive plans based on previous experiences.
- The information is captured using different sensors.

As shown in Fig. 15, the information provider agents are directly connected to the information gathering devices. The application in this case is composed of seven modules to control: (i) the location of the patient, (ii) the interior lighting, (iii) temperature, (iv) gas leaks, (v) flood detection, (vi) smoke detection, and (vii) control screen. The resource identification agent is responsible for identifying and accepting or rejecting the data submitted by the information providers. Its task is to oversee the information provider agents that incorporate into the system. All of the data are stored in the system and interpreted by the interpreter agent.

The interface agent provides the interpreter agent with a series of basic activities that the COPD patient must perform at home. This activity list is defined by the medical personnel monitoring the patient. Table 1 shows the basic list of activities for a COPD patient.

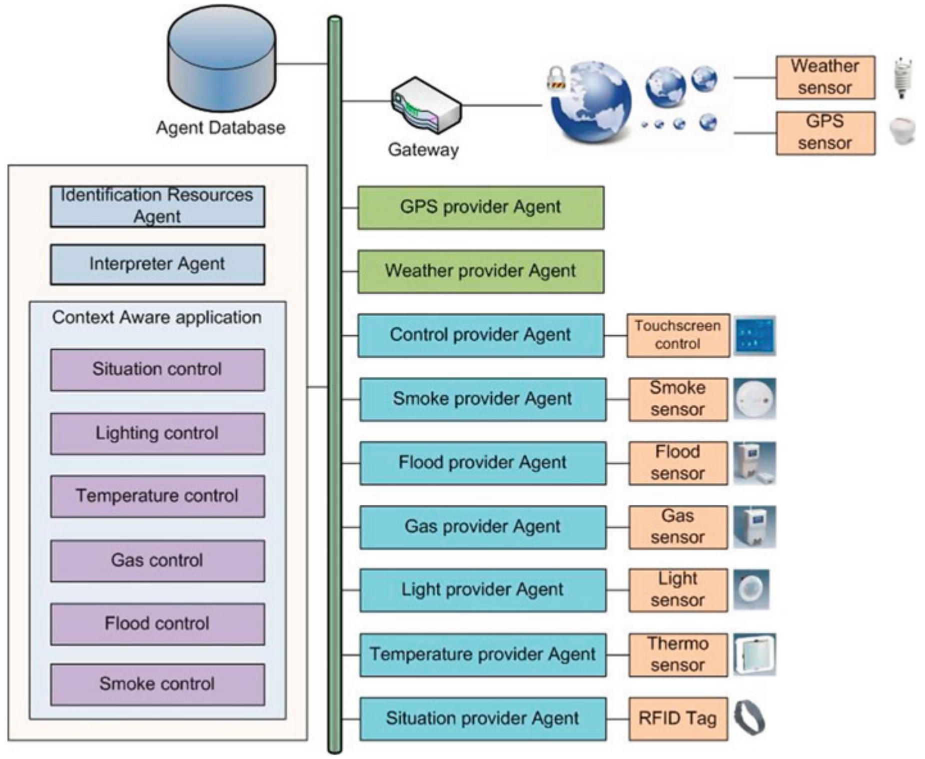


Fig. 15 Home care context-aware application

Table 1 List of activities for COPD patient

Activity	Name	Time (min.)	Task
A	Oxygen cylinder	600	1
B	Get up and do exercise	10	6
C	Breakfast	10	10
D	8:00 am pill and spray	3	3
E	Doctor visit	60	14
F	Walk	30	4
G	Mid morning snack	20	9
H	Oxygen cylinder	300	2
I	2:00 pm pill and spray	3	3
J	Lunch	40	10
K	Afternoon snack	20	11
L	Walk	30	5
M	8:00 pm pill and spray	3	3
N	Dinner	30	12

Once the activity list has been defined with the appropriate instructions as provided by medical personnel, the interpreter agent generates an order to create a task plan following

Table 2 Activities and their prerequisites

Activity	Prerequisites	Time (min.)
A	–	600
B	–	10
C	–	10
D	A	3
E	A, B, C, D	60
F	E	30
G	F	20
H	G	300
I	G	3
J	I	40
K	J	20
L	H, K	30
M	L	3
N	M	30

the CPM method. With the activity list, the prerequisites for each activity, and the time requirement list, it is possible to determine the duration for executing the task plan and developing the network for the task plan. Table 2 shows the activity list and prerequisites.

Figure 16 shows the network associated with the previous task plan, with each of the nodes numbered, and the earliest and latest possible times.

According to the network in Fig. 16, the activities can be found between two nodes where $ES_i = LS_i$ are critical tasks and determine the critical path. As such, the activities that compose the critical path in the network in Fig. 16 are A, C, E, F, G, H, L, M, and N. The remaining activities are carried out in parallel to the previous ones. For this reason, it is possible to delay the start of the non-critical activities or increase their margin for delay without increasing the overall duration of the task plan. Once the interpreter agent has defined the task plan and its duration, it sends this information to the rest of the HoCCAC system agents to execute the plan. The provider agents along with the interface agent are responsible for overseeing the completion of the task plan. If at any time the plan is interrupted, for example, because the patient is choking and has to perform the activity specified for this situation, the HoCCAC system is notified by the interface agent and the interpreter agent immediately replans the task plan. Another special case can result from a visit to the doctor, which takes place monthly. When there is no need to visit the doctor, the node representing this activity on the network is not present. Depending on the type of interruption, the provider agent may also generate a warning as, for example, in the event that an oxygen tank does not function. Figure 17 shows the Gantt diagram with the solution associated with the task plan shown in Fig. 16. It is also possible to deduce from Fig. 17 which activities are critical and which are not. The critical activities are shown in red, while those with a margin for delay are in blue. It is also possible to see which tasks have a temporal overlap. Thus, the activity of putting on the oxygen cylinder overlaps with waking up, doing exercise, and having breakfast. It is also possible to deduce that the visit to the doctor takes place after the previous activities, as well as after taking the morning medication.

The case study was carried out during 3 months. During three weeks, we observed the environment and the patient behavior. The third week was used to deploy the technological

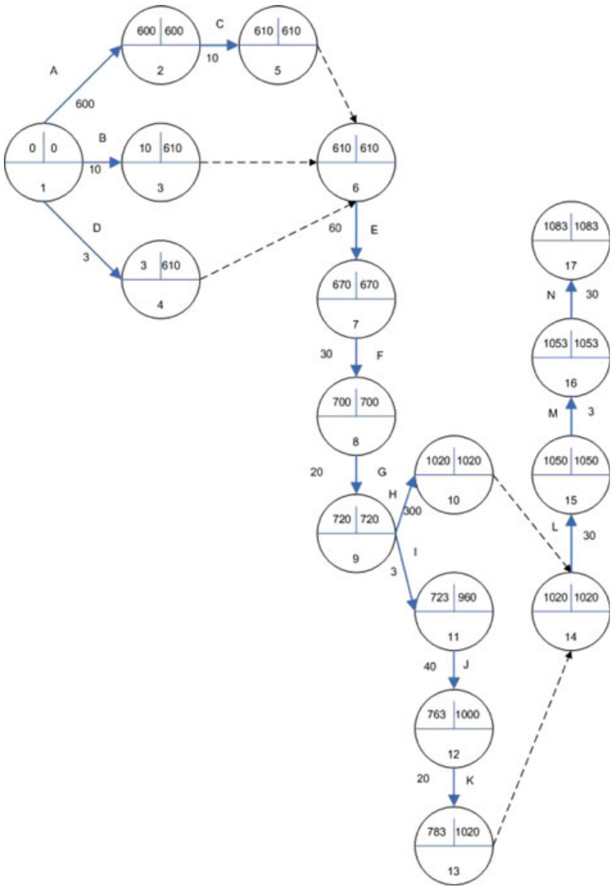


Fig. 16 Network of task plan defined for COPD patient

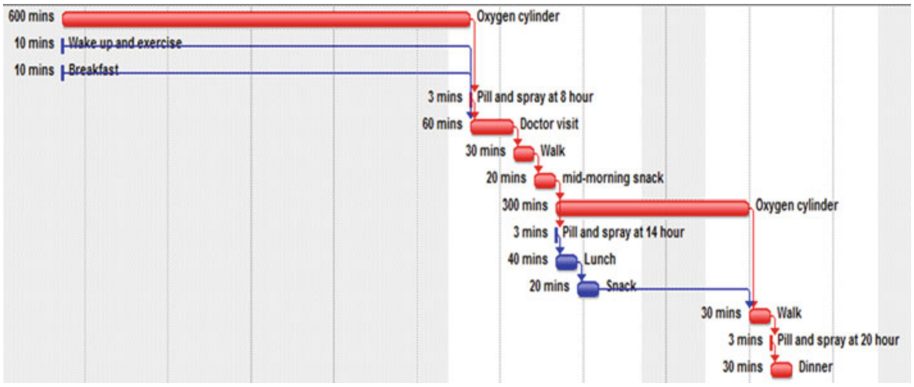


Fig. 17 Gantt diagram associated with the task plan

infrastructure and tune up the architecture. In the second month, the system obtained context-aware information about the environment and the medical staff. Finally, the system was

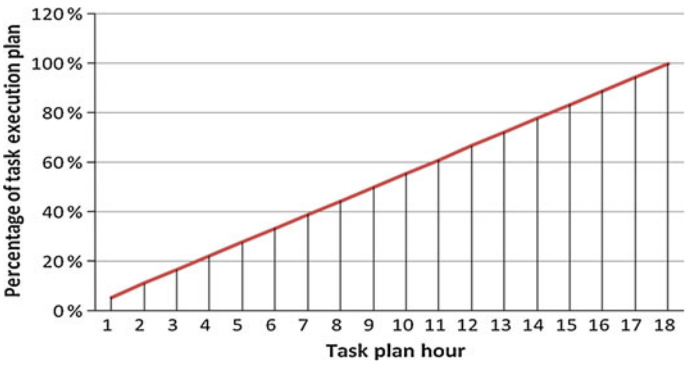


Fig. 18 Performance graph associated with the task plan presented in the case study.

tested during 1 month in the real environment, providing daily plans to the patient. During the sequence of operations carried out by the HoCCAC agents, it is necessary to consider the transformation of the information produced in the system. First of all, low level data are collected from the patient's environment. These are subsequently stored in the information system as high level data so that they are more quickly interpreted and easier to use. The information provider agents work with the interpreter agent to carry out this task. Additionally, the patient can use context-aware applications to interact with the context at all times and establish the parameters that dictate how the HoCCAC system functions.

5 Results

HoCCAC was used to develop a prototype for the home of a dependent patient. It incorporates JavaCard technology to identify and control access, which provides an added value to existing RFID technology. The integration of these technologies provides the system with the capability to perceive stimuli from within the environment automatically and in execution time. This makes it possible to personalize the system's functioning and adjust it to the characteristics and needs of the context for any specific situation.

The CPM planning mechanism is appropriate and simple to implement in a dynamic and changing environment in which information systems require the incorporation of mechanisms that can obtain high levels of learning and adaptation. The proposed planning system differs from other existing systems in that it optimizes the possibility of assigning new plans and dynamic replanning in execution time. To obtain results and data from the evaluation of the CPM method used in our case study, we created a performance graph (Fig. 18) which allows us to observe the rate of development for the activities described in the case study. At the same time, this graph enables us to know which activities are nearing completion. In the graph shown in Fig. 18, the *y*-axis lists the percentage of completion for the task plan, while the x-axis lists the number of hours used for the defined task plan. The plans evolved over the time, and we can show an example of a plan obtained during the last week. Our case study presented a task plan with a total duration of 18 h and 5 min. This duration was calculated by adding together the times associated with the tasks that compose the critical path. This graph indicates the final task plan, which is found on the 100 % efficiency path and the final time coordinate for the plan of actions.

At this point, it is now possible to calculate the advances achieved in the project every hour, as shown in Fig. 18. The advance in the task plan is the advance of the sum of the

Table 3 Advance report for task plan activities

Hour	Activity	% advance	Hour	Activity	% advance
1	A	10 %	12	G	T
1	B	T	12	H	3 %
1	C	T	12	I	T
2	A	20 %	13	H	23 %
3	A	30 %	13	J	T
4	A	40 %	14	H	43 %
5	A	50 %	15	H	63 %
6	A	60 %	15	K	T
7	A	70 %	16	H	83 %
8	A	80 %	17	H	T
9	A	90 %	17	L	5 %
10	A	T	18	L	T
11	D	T	18	M	T
11	E	95 %	18	N	98 %
12	E	T	19	N	T
12	F	T			

advances achieved for each of the activities in the plan. Table 3 shows the advance reports for each of the activities in the defined case study. Immediately upon receiving information on the actual advance of the task plan, the interpreter agent fills in the data in Table 3 in the following manner:

- Document the hour that the information is received (*Hour* column)
- Document the activity carried out during that hour in the *Activity* column
- Finally, the *% advance* column lists the advances the activities have made during that hour. The letter "T" appearing in the column means that the activity has concluded.

Given the data defined in Table 3, it would be possible at any time to confirm the advance in the task plan by adding together the partial advances achieved by the activities. This sum represents the actual advance of the task plan. It is important to note that 1-hour time intervals were used for the present case study because it is a daily task plan. However, there can be more specific and short task plans where the time intervals can be measured in minutes. The diagonal line in the graph shown in Fig. 18 is completely straight because during the execution of the activity plan for the case study, there was no delay or acceleration in any activity. When the task plan has accelerations or delays, the resulting graph is not represented with a straight line, but with various arcs as shown in Fig. 19. The graph in Fig. 19 shows that there is an acceleration delay in progress with the tasks in zone (a) and there is also a delay acceleration in progress with the tasks in zone (b) with regard to the correct execution of the task plan.

HoCCAC functions like a global system for context-aware environments. HoCCAC not only captures information from the environment and interacts with users and their requests, but also, assisted by the information system, constantly evaluates the attributes for the user context and provides proactive solutions. The solutions provided by HoCCAC are supported by a vast knowledge base that the system stores and processes. At the same time, the system provides the patient with proactive task plans that attempt to improve the patient's quality of

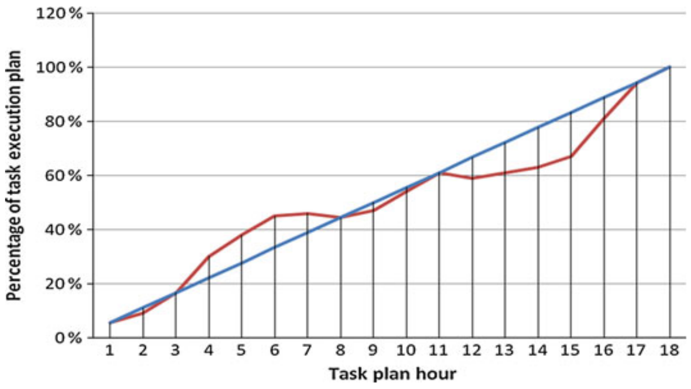


Fig. 19 Performance graph associated with the task plan experiencing accelerations (**a**) and delays (**b**) in the activities

life. The user can perform daily tasks and receive support from the intelligent context without the need for explicit interaction. As a result, the user is not required to learn to use the system. This increases the level of user satisfaction with the HoCCAC system managing the environment. Furthermore, the HoCCAC system evaluates the use of resources in the environment, sending useful reports to the medical personnel. Information on the use of resources is supplied by the HoCCAC provider agents, while the interpreter agent is responsible for evaluating the supplied information. The interpreter agent uses this evaluation to determine how the daily resources will be allocated to the activities requesting their use. The interpreter agent also examines its own evaluations and can reassign activities to available resources to shorten the duration of a task plan. This allows the HoCCAC system to progressively improve the solutions offered over time.

Furthermore, the technology used is not conditioned by the services offered. The choice of JavaCard and RFID to locate and identify objects and users is independent of the specific implementation of the technology, that is, the services are valid for any type of sensors, antennas, readers, and RFID tags. Thus, the system is valid for very different types of environments whose devices need not have a fixed technology base. The incorporation of new JavaCard and RFID sensors in the case study significantly improves a series of functions such as controlling medication and diet, and detecting anomalies in the patient's behavior, as shown in Table 4. The percentages for each task have been calculated by means of the rule of Laplace and they are shown in the Table 4:

$$\frac{Number\ of\ activities\ that\ arecompleted}{Number\ of\ activities\ of\ thattask} * 100 \tag{6}$$

Table 4 Comparison of control variables in the case study before and after implementing the HoCCAC architecture

	Control medication (%)	Control meals (%)	Anomalies in patient behavior (%)
Before HOCCAC	92	88	46
After HOCCAC	100	100	15

For example, regarding the control of the taking of medications, we use a sensor to weight the pill container and to inform to the system when the pills have been taken. If the weight does not vary when a medication should have been taken, it is considered anomaly. The anomalies are measured in terms of number of episodes of the abnormal patient behavior, since we are working with completed and incomplete tasks.

Table 4 shows the percentage of success in controlling the three variables associated with the case study. One can easily appreciate the efficiency of the HoCCAC system in controlling the intake of medication and food for each patient with a 100 % efficiency rate. The system avoids errors in the administration and control of medication and special diets for the patients. Furthermore, the HoCCAC system can detect anomalous behavior in the patients and create behavior patterns for high risk situations. The percentage of anomalous situations has decreased substantially. The main reason is the use of a CBR system in the interpreter agent that takes past experiences into account in order to organize the new task plans for the system. In this regard, the HoCCAC system takes advantage of past experiences and learns from them. This behavior provides a great adaptive ability, as shown in the results in Table 4.

6 Conclusions

While there is yet much work to be done, the system prototype developed for this study improves home security for dependent persons through the use of vigilance and alert devices. The interpreter agent in the HoCCAC system is capable of reacting automatically when faced with dangerous or emergency situations, replanning any plans in progress and sending alert messages to the system. Thus, HoCCAC creates a context-aware environment that facilitates the development of distributed intelligent systems and provides services to dependent individuals in their home. This makes it possible to automate certain tasks used to monitor dependent persons and improve their quality of life. The use of multiagent systems, automatic control and RFID systems, JavaCard, and mobile devices provides a high level of interaction between the context, caregivers, and patients. Furthermore, the proper use of mobile devices facilitates social interactions and transfer of knowledge. Our future work will focus on obtaining a model to define the context, as well as improving the current prototype and its testing with a big number of patients, as well as with patients suffering from other diseases or deficiencies.

7 Discussion

This paper presented a prototype where the HoCCAC architecture was used to improve the quality of life of a COPD patient but, in the future, the system needs to be tested with a big number of patients to have a more objective evaluation. The system was developed to be applied in a COPD scenario, but we believe that it can be easily adapted to be applied in most of the home care scenarios, especially for elderlies. The setup and number of sensor to be used are dependent on the characteristics of the scenario. It is necessary to take into account that the planning system requires initial information to provide acceptable solutions, as it is based on a CBR system. These are our next challenges.

Acknowledgments This work was supported in part by the MEC OVAMAH 2009-13839-C03-03 Project.

References

1. Anastasopoulos M, Niebuhr D, Bartelt C, Koch J, Rausch A (2005) Towards a reference middleware architecture for ambient intelligence systems. In: ACM conference on object-oriented programming, systems, languages, and applications
2. Angulo C, Tellez R (2004) Distributed intelligence for smart home appliances. In: Tendencias de la minería de datos en España. Barcelona: Red Española de Minería de Datos. pp 1–12
3. Aquilano NJ, Smith DE (1980) A formal set of algorithms for project scheduling with critical path scheduling/material requirements planning. J Oper Manag 1(2):57–67
4. Ardissono L, Petrone G, Segnan M (2004) A conversational approach to the interaction with Web Services. Comput. Intell. 20:693–709
5. Bahadori S, Cesta A, Grisetti G, Iocchi L, Leone R, Nardi D, Oddi A, Pecora F, Rasconi R (2003) RoboCare: pervasive intelligence for the domestic care of the elderly. Artif Intell 1(1):16–21
6. Bajo J, De Paz JF, Tapia DI, Corchado JM (2007) Distributed prediction of carbon dioxide exchange using CBR-BDI agents. Int J Comput Sci (INFOCOMP) Special Edition 16–25
7. Bajo J, De Paz Y, De Paz JF, Corchado JM (2009) Integrating case planning and RPTW neuronal networks to construct an intelligent environment for health care. Expert Syst Appl 36(3):5844–5858
8. Borrajo ML, Corchado JM, Corchado E, Pellicer MA, Bajo J (2010) Multi-agent neural business control system. Inf Sci 180(6):911–927
9. Burrell J, Gay G (2002) E-graffiti evaluating real-world use of a context-aware system. Interact Comput Special Issue Univers Usab 14(4):301–312
10. Chen Z (2000) (Sun Microsystems). Java card technology for smart cards. Addison Wesley Longman. ISBN 0201703297
11. Corchado JM, Pavón J, Corchado E, Castillo LF (2005) Development of CBR-BDI agents: a tourist guide application. In: 7th European conference on case-based reasoning 2004. pp 547–559
12. Corchado JM, Laza R (2003) Constructing deliberative agents with case-based reasoning technology. Int J Intell Syst 18(12):1227–1241
13. Corchado JM, Bajo J, de Paz Y, Tapia D (2008) Intelligent environment for monitoring alzheimer patients, agent technology for health care. Decis Support Syst 34(2):382–396
14. Dey AK (1998) Context-aware computing: The CyberDesk project. In: Proceedings of the AAAI, spring symposium on intelligent environments, Menlo Park, CA
15. Dundas GR, Krentler KA (1982) Critical path method for introducing an industrial product. Ind Mark Manag 11(2):125–131
16. Gu T, Pung HK, Zhang DQ (2004) Towards an OSGi-based infrastructure for context-aware applications in smart homes. IEEE Pervasive Comput 3(4):66–74
17. Gupta T (1991) Applying the critical path method to manufacturing routing. Comput Ind Eng 21(1–4):519–523
18. Holvoet T, Valckenaers P (2006) Beliefs, desires and intentions through the environment. In: AAMAS'06, proceedings, pp 1052–1054
19. Jameson A (2001) Modeling both the context and the user. Pers Ubiquitous Comput 5(1):29–33
20. Kaluža B, Luštrek M, Dovgan E, Gams M (2012) Context-aware MAS to support elderly people (demonstration). In: Proceedings of the 11th international conference on autonomous agents and multiagent systems, vol 3, pp 1485–1486
21. Kerer C, Dustdar S, Jazayeri M, Gomes D, Szego A, Caja JAB (2004) Presence-aware infrastructure using web services and RFID technologies. In: Proceedings of the 2nd European workshop on object orientation and web services, Oslo, Norway
22. Kwak NK, Long DA, Schniederjans MJ (1985) A CPM analysis of microbiology computerization project for hospital management information processing. Inf Process Manag 21(2):149–155
23. Lim CH, Anthony P, Fan LC (2009) Applying multi-agent system in a context aware. Borneo Sci 24:53–64
24. Lina M, Lin Z (2006) A cost-effective critical path approach for service priority selections in grid computing economy. Decis Support Syst 42(3):1628–1640
25. Muñoz MA, Gonzalez VM, Rodriguez M, Favela J (2003) Supporting context-aware collaboration in a hospital: an ethnographic informed design. In: Proceedings of workshop on artificial intelligence, pp 330–334
26. Park D, Hwang S, Kim A, Chang B (2007) A context-aware smart tourist guide application for an old palace. In: Proceedings of the 3rd international conference on convergence information technology, pp 89–94
27. Pokahr A, Braubach L, Lamersdorf W (2005) Jadex: a BDI reasoning engine, chapter multi-agent programming. In: Multi-agent programming: languages, platforms and applications, vol 15. Springer, pp 149–174

28. Pokahr A, Braubach L, Lamersdorf W (2003) Jadex: implementing a BDI-Infrastructure for JADE Agents. In: EXP—in search of innovation (Special Issue on JADE) pp 76–85
29. Schilit B, Theimer M (1994) Disseminating active map information to mobile hosts. IEEE Netw 8(5):22–32
30. Skov M, Hoegh R (2006) Supporting information access in a hospital ward by a context-aware mobile electronic patient record. J Pers Ubiquitous Comput 10:205–214
31. Uhm Y, Hwang Z, Lee M, Kim Y, Kim G, Park S (2007) A context-aware multi-agent system for building intelligent services by the classification of rule and ontology in a smart home. In: 32nd IEEE conference on local computer, network, pp 203–204
32. Wang Y, Goddard S, Perez L (2007) A study on the cricket location support system communication protocols. In: 2007 IEEE international conference on electro/information technology, pp 257–262
33. Want R, Hopper A, Falcao V, Gibbons J (1992) The active badge location system. ACM Trans Inf Syst 10(1):91–102
34. Zhao Z, Belloum A, de Laat C, Adriaans P, Hertzberger B (2007) Using Jade agent framework to prototype an e-Science workflow bus. In: Seventh IEEE international symposium on cluster computing and the Grid, pp 655–660

Author Biographies

Juan A. Fraile obtained a Ph.D. at the University of Salamanca (Spain). He obtained the Information Technology degree at the Pontifical University of Salamanca (Spain) in 1996 and Engineering in Computer Sciences degree at the Pontifical University of Salamanca in 2000. He has been involved in the development of automated systems in the DMR Consulting company and the Data Processing Centre (Pontifical University of Salamanca). He has also been a co-author of papers published in recognized workshops and symposiums.

Yanira De Paz received a Ph.D. in Mathematics from the University of Salamanca (Spain) in 2008. At present, she is a Professor of the International University of the Rioja and Consultant of the University of Cataluña. She hold a scholarship provided by the Spanish Minister of Education to complete a Ph.D. program at the University of Salamanca (Spain). She obtained a Mathematics degree in 2002 and a Statistic degree in 2003 at the University of Salamanca (Spain). She was an Assistant Professor at the Faculty of Economy at the University of Salamanca and European University of Madrid; and co-author of several mathematical, statistical… books, and articles. She has also been lecturer in the Faculty of Mathematics at the Complutense University of Madrid.

Javier Bajo received a Ph.D. in Computer Science and Artificial Intelligence from the University of Salamanca in 2007. At present, he is Associate Professor at the Pontifical University of Salamanca (Spain) and researcher at the BISITE research group (http://bisite.usal.es) at the University of Salamanca (Spain). He obtained an Information Technology degree at the University of Valladolid (Spain) in 2001 and an Engineering in Computer Sciences degree at the Pontifical University of Salamanca in 2003. He has been a member of the organizing and scientific committee of several international symposiums such as CAEPIA, IDEAL, and HAIS and is co-author of more than 150 papers published in recognized journals, workshops, and symposiums.

Juan Francisco De Paz received a Ph.D. in Computer Science from the University of Salamanca (Spain) in 2010. He is Assistant Professor at the University of Salamanca and researcher at the BISITE research group (http://bisite.usal.es). He obtained a Technical Engineering in Systems Computer Sciences degree in 2003, an Engineering in Computer Sciences degree in 2005 at the University of Salamanca, and Statistic degree in 2007 in the same University. He has been co-author of published papers in several journals, workshops, and symposiums.

Belén Pérez-Lancho received a Ph.D. in Sciences from the University of Salamanca (Spain) in 1995 and she completed her postdoctoral formation at the Pierre and Marie Curie University of Paris (France) in 1996–1997. She is Professor of Computer Science and Vice-dean of Infrastructures and Innovation at the Faculty of Sciences (University of Salamanca). She teaches undergraduate courses in Computer Sciences and Physics and official graduate courses in Intelligent Systems. Her main research interests are in the area of modelling, intelligent control, and computation. She is co-author of several papers published in recognized journals, congress, and symposiums and has participated in national and international research projects.