Chapter 2

Theoretical Background

In this chapter, it is briefly presented the basic concepts for helping to understand the goals of this thesis. Firstly, it is presented the agent and MAS fundamentals, where the main concepts about the openness of environments and systems are addressed. Then, it is shown some of the technologies for the development of MAS; afterward, we present the relation between middleware and the IoT focusing on the *ContextNet* middleware, which will be used to provide connectivity and communicability in this thesis.

2.1 Agents and Multi-Agent Systems

Agents are intelligent entities coming from Artificial Intelligence that is capable of sensing an environment where they are situated, and after a reasoning process, they are able of acting upon the same environment in order to achieve common or conflicting goals. Agents are autonomous entities capable of learning from their past experiences, interaction with other agents, and perceiving the environment. They are also adaptable and flexible, being capable of reacting pro-actively to changes within the environment they are situated. An agent can be part of a system where it can interact in a peer-to-peer way with other agents. Hence, a Multi-Agent System (MAS) is a system, which comprises several agents competing or collaborating for achieving individual or collective system's goals. Every agent of a MAS are responsible for acting and sensing a sphere of influence, which is a piece of the environment, and more than one agent can overlap this sphere, competing or working together based on their individual or collective goals [131].

These characteristics of agents are often related to AmI and Ubiquitous systems since they need intelligent, autonomous and pro-active entities capable of interacting and reason in a highly dynamic ambient where persons and artifacts share information and cohabit

pervasively [68]. As AmI systems and ubiquitous computing consider systems built upon real ambients using physical devices taking actions pervasively, agents must follow up these characteristics. Agents can be virtual or physical entities [75]. A virtual agent runs simulated on a computer using a virtual representation of an environment, and it does not support an interface with physical devices or hardware. Conversely, a physical agent interfaces with hardware devices such as actuators and sensors to interact with real-world environments. It is embodied in physical infrastructure, and it can run embedded into platforms, tiny computers, or any computational technology. A traditional vision of a physical agent is a robot [103].

At this point, it is essential to define that both environments and ambients represent a physical or simulated place where information and entities — artifacts, devices, persons, etc. — can exchange messages, be perceived, handled and modified by entities present in the local at a specific time. In AmI, Ubiquitous Computing, and Ambient Assisted Living (AAL), for example, ambient is a physical representation of someplace such as a room or house with expected behavior and equipped with pervasive technologies. It is not limited to physical places, and simulated ambients are often used in many solutions [30, 67, 1, 72]. However, it is expected to materialize ambients and devices when adopting AmI or Ubiquitous systems. When considering agents, the environment is a place containing artifacts that can be perceived and used by agents or groups of agents in a MAS for helping in accomplishing their goals. Depending on the cognitive model adopted by agents, the artifacts' data assume different names such as beliefs and percepts. Many domains use agents as a programming abstraction, and simulated environments are not necessarily used as a trustworthy representation of reality. In some cases, agents can interact directly with the real world [75, 112, 66].

The open nature of a real ambient allows the entry and exit of entities at any time. It is natural to think in a room where it is possible to enter or leave freely or to add new features such as a television or a sound system. However, in these systems, environments can be open or closed depending on their configuration and purpose [23]. An open environment works as a real ambient allowing new artifacts and agents to enter or leave it in any time. Conversely, a closed environment allows artifacts and agents that exist from the beginning. This latter implies that the closed environment must be simulated since it does not make sense that exists a real environment that is closed for new entities or, at less, it is not useful. In this thesis, we consider an agent's environment as an open and physical environment that represents a real spot or place where exists agents that can perceive and act upon this same environment controlling objects.

The fact of existing open environments is related to the notion of MAS. If an open environment exists where agents can enter and interact with other agents, sharing information, and collaborating or competing, it is possible to consider that these agents are part of a dynamic MAS. In this thesis, we employ MAS on top of objects, and only objects can enter or leave the environment, which avoids the existence of standalone agents running freely in the environment. Hence, it is necessary to expand the discussion of openness to the MAS level.

Similarly to environments, MAS can be closed or open, depending on how they treat the mobility of their agents. The conventional approaches consider that agents established in a MAS cannot move to another system, and it does not allow existing agents elsewhere to enter in its system. Then, in a closed MAS, there is an architectural limitation that avoids the mobility of agents. When considering the communication of these agents, commonly the MAS also limits this interaction to agents that are part of the same system. However, agents able to communicate with agents from different systems does not change the openness of the system. If a MAS has open communicability, it does not mean that it is prepared for the entry of new agents in the system. Instead, an open MAS allows agents to enter or leave the system at runtime, and there are no limitations in communication since they can perform direct communication in the destiny system, for example. Hence, the openness of MAS is defined by mobile agents, which are capable of moving from one system to another and stay as long it is interesting for them or until they accomplish their expected goals [62].

However, in practice, it is hard to define a physical border for limiting agents inside a MAS and programming languages are responsible for dealing with the relation agents and system. In the same way, it is common to observe a MAS using hybrids approaches considering agents, communication, and environments. In this thesis, we treat a MAS as a closed system where particular types of agents can use abilities to communicate with other agents from other systems hosted in IoT Objects. However, the technologies employed in the development of our approach allows agents to be movable and systems to be open, and it has been explored in parallel approaches [41, 38, 39].

As stated before, agents are intelligent entities capable of reasoning about statements perceived from an environment and social relations to accomplish goals. For this, agents adopt cognitive models of reasoning that tries to model their mental behavior. A cognitive model tries to understand or scientifically explain basic cognitive processes involving the brain in how to accomplish complex tasks as perceiving, learning, and decision making [16].

There is extensive literature considering cognitive models in the agent domain, but one is highlighted at this point since it is used in this thesis. The Belief-Desire-Intention (BDI) [15] considers the cognitive process of practical reasoning based on beliefs, desire, intentions, and plans. Agents can perceive the environment as pieces of information named Beliefs, which are used for triggering desires and intentions of an agent. Plans and actions materialize Desires and Intentions considering agents' beliefs and goals, which define when the agent commits to a desire, transforming it in intentions. The BDI has been widely used in the development of MAS during the past years, including IoT and AmI.

2.2 Development of MAS

During the last decades, agents emerged as a paradigm of Artificial Intelligence for solving distributed and decentralized problems in different domains, including IoT recently. The agent approach provides abstractions and mechanisms based on cognitive models that facilitate the development of intelligent, pro-active, collaborative, and dynamic systems. There are several AOPL and frameworks used in the development of MAS in the literature using different reasoning models such as BDI or merely reactive agents, for example. The Java programming language was used in the last years for creating three of the well-known Java-based frameworks: Jade [7], Jack [19], and Jason [11].

Jade is a reactive framework where agents are developed in a Java-like style. However, there is an extension using the BDI model named Jadex [95] that runs over Jade. The frameworks Jack and Jason use the BDI architecture alongside an interpreter of the Procedural Reasoning System for providing real-time reasoning systems. Besides, Jason works together with CArtAgO [98], which provides an abstraction of artifacts placed in environments that agents can interact with and Moise [55], which presents an extension for normative and organizational models in Jason. These three technologies together are known as the JaCaMo [9] framework.

When considering applications for IoT and AmI systems using MAS, none of these frameworks are prepared for interfacing hardware and communicate to an IoT network. Hence, if one of the mentioned frameworks would be employed in the development of IoT Objects, interface mechanisms, or middleware should be necessary. The Jade itself has been used in domains such as robotics and IoT but since its reactive nature and the absence of a proper abstraction for implementing agents — agents are programmed in

Java — it is not the most appropriated framework to be employed in autonomous and intelligent IoT Objects, which is expected some cognitive behavior. Even Jadex, which implements the BDI model, still depends on Jade since this extension runs on top of Jade. The Jack framework is proprietary software, which makes it difficult to access the source code or modify the agent's reasoning cycle, for example. The frameworks Jason or JaCaMo have a BDI reasoning cycle implemented in Java, and they are open and free platforms. Besides, they have several points of extensions that can be explored for creating agents with modified behaviors, and to add external technologies. Then, it is essential to understand Jason's internal structures that will be explored in this thesis for creating Things and Smart Things.

2.3 Framework Jason

The Jason is a *framework* for developing MAS using the cognitive model BDI [15] and has an interpreter for the BDI based agent-oriented programming language *AgentSpeak* [97] in Java language. The BDI contains three basic constructions: beliefs, desires, and intentions. Beliefs are information considered to be true by the agent, which can be internal, acquired by a relationship with other agents or by the environment's information. Desires represent the agent's motivation to perform a determined goal. Intentions are actions that the agent is compromised to execute. Moreover, the Practical Reasoning System allows agents to build a reasoning system at runtime to execute complex tasks [11]. Besides, they have plans composed of actions that are activated depending on beliefs on their belief bases.

Specifically, Jason's standard agent has a reasoning cycle responsible for processing all perceptions and beliefs to generate events, which activate plans and actions. It is important to understand the reasoning cycle of a standard agent because several extensions (including the ones described in this thesis) modify some of its characteristics to enhance specific kind of agents with new customized abilities. The reasoning cycle (Figure 2.1) of a standard agent is composed of the following steps:

- **Capturing Perceptions**: The agent captures the perceptions from a simulated environment where it is situated. In this environment, the agents can interact with virtual objects that may have information represented as perceptions. In Jason, the perceptions and beliefs are literals. It is important to remark that the original distribution of Jason does not have any access to real environments using sensors

or actuators.

- **The Belief Update Function**: This function updates the Belief Base using the captured perceptions of the environment, beliefs received from messages, and self-statements generated internally during plan execution. For each modification in the Belief Base, an event is generated. An event represents a consequence of something that an agent has to deal with to achieve its goals based on those new beliefs.
- **Checking and Selecting Messages**: The agent has a mailbox for receiving messages from other agents. It verifies at the beginning of each cycle if exists messages to be read. Then, it can select socially acceptable messages to be processed or ignore the one that is not acceptable. This step also generates events.
- **Event Selection**: In this step, an event is selected from a list of generated events of the previous steps.
- **Dealing with Plans**: when an event is selected, it retrieves all the relevant plans of the agent's plan library. After that, a verification is performed to identify which plans can be executed based on its current beliefs and perceptions, and a function selects only one plan to be executed.
- **Selecting Intentions**: a function is responsible for choosing one ready-to-use intention at a time to be executed.
- **Executing Actions**: Finally, an action of the selected plan is executed one at a time.

The Jason does not have the necessary technologies for implementing IoT Objects for working in IoT and AmI systems. It does not interface hardware, and it does not communicate with different MAS. It is only possible to create MAS that accesses simulated environments as said before, and the communicability is limited to agents inside the created MAS. Hence, it is essential to adapt Jason for creating the IoT Objects. There is an extension of Jason [90, 89] named ARGO that uses a serial hardware interface [63] for transferring sensors' values as perceptions directly for agents and receives agent's actions to activate actuators, which helps the creation of embedded MAS using microcontrollers. This extension will be used in this thesis as part of the creation of IoT Objects and will be explained in details in Chapter 5.1.

For applying IoT Objects controlled by agents in IoT, open environments are necessary and, consequently, communicability and connectivity. In this case, middleware for IoT

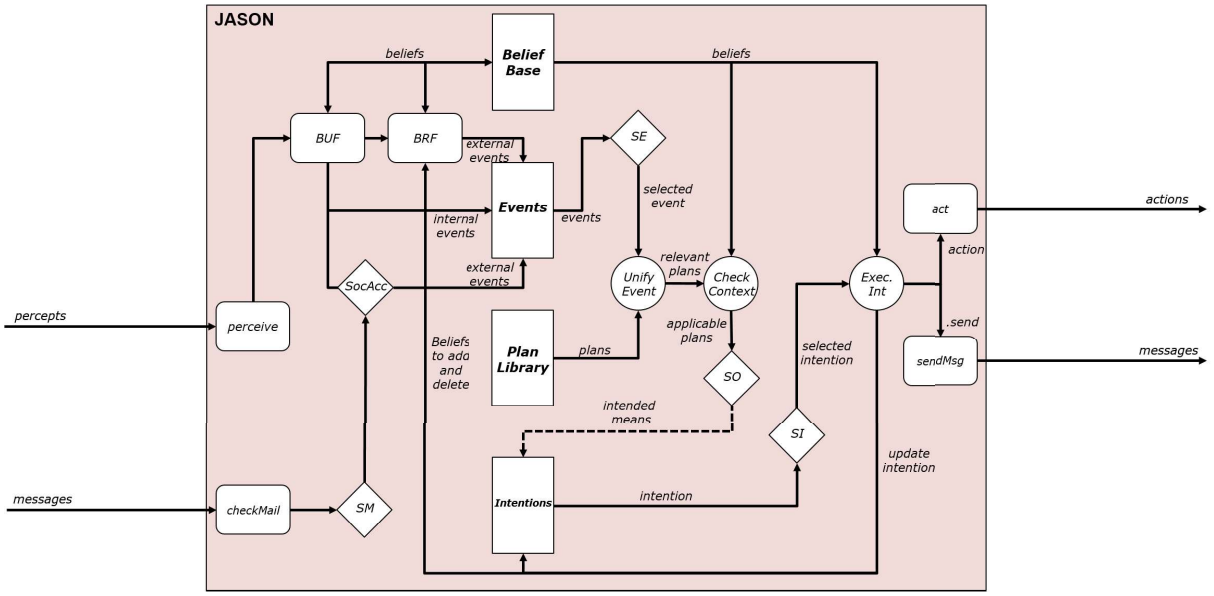

Figure 2.1: The reasoning cycle of a Jason agent [11].

should play an essential role in dealing with these issues since some of them can guarantee scalability, connectivity, communicability, data sharing, and protocols. In the next section, it is explored the chosen middleware, which is the basis for creating the proposed Internet of Smart Things (IoST) in this thesis.

2.4 Internet of Things

IoT middleware and protocols are responsible for managing the integration of the physical world and the cybernetic ones by using IoT objects and establishing an interconnected IoT network of these objects with the purpose of data collection and analysis, and reactive applications and systems [85]. The increasing number of middleware and connectivity protocols designed specifically for IoT do not consider the heterogeneity of such objects and their needs. When combining agent approach and IoT middleware for providing an IoST, the former one needs to be autonomous and pro-active and independent from the latter one. Besides, the IoT network should provide open environments where IoT Objects embedded with agents can enter and leave anytime they want.

Hence, it is necessary an open, lightweight and secure middleware capable of dealing with the heterogeneity of IoT Objects and agents technologies to be employed as the basis of the IoST without bounding the IoT Objects to the system. Besides, it is important to offer an IoT layer where different types of clients can access whenever it is necessary

without interfering in the IoT Objects functioning. In this thesis, we consider both IoT Objects and Clients as being built over agent methodologies.

A cloud-based IoT architecture should provide the necessary abstraction for creating an IoST capable of dealing with IoT Objects controlled by agents. Besides, it considers an uncoupled three-layer architecture where an IoT middleware working as a middle layer deals with connectivity and communicability of IoT Objects and clients. IoT Objects are capable of connecting and disconnecting from it as part of the device layer, and clients can interact or access these objects by accessing Application Programming Interfaces (API). Based on that, we selected the ContextNet middleware, which will provide all the necessary abstractions and constructions for creating the IoST. In the next section, the main concepts of ContextNet will be explained.

2.5 ContextNet

The *ContexNet* [47] middleware is a service for providing context data in stationary and mobile networks. It provides context services for ubiquitous and pervasive applications, and it has been employed in a wide range of solutions [36, 120, 49, 43]. It uses a Scalable Data Distribution Layer (SDDL), which employs the Data Distribution Service (DDS) [93] protocol for a real-time Publish/Subscribe mechanism for the communication within the SDDL Core. Besides, it also uses the Mobile Reliable UDP (MR-UDP) [109] for performing communication between mobile nodes and the core application running in servers. There are other services provided by the SDDL core, such as data persistence, data stream, fault tolerance, node disconnection, and group communication.

The data transferring occurs by using the MR-UDP and the Object Management Group (OMG) DDS. The MR-UDP treats messages between a client and a gateway, and the DDS is responsible for distributing data in the core of the network. The DDS is an OMG standard built upon a peer-to-peer architecture for data distribution, which guarantees Quality of Service (QoS) contracts between data users and providers. By using *ContextNet*, it is possible to enable the growth of a network, ensuring the scalability of the content distribution between millions of devices. From the point of view of the IoT developer, the ContextNet middleware allows the development of clients and core applications. Clients can be fixed or mobile devices able to connect to the server, and to communicate with other clients and to the core application. It deals with the MR-UDP connection to available gateways, and it isolates technical details from the application

layer. The core application deals with the data flow coming from clients, and it is useful for creating solutions where it is essential to collect data from several different clients and process this information somehow. In this thesis, the ContextNet middleware is used in the cloud solution as the core application dealing with all the requests from IoT Objects and other application that need to access to retrieve some action or act upon some environment. It offers all the necessary constructions to allow the development of the proposed architecture as the scalability, communicability, and connectivity.

Besides, the ContextNet also provides dynamic management for groups of clients. Devices running the client library of ContextNet can be arranged in groups, which helps to organize the collective goals and to facilitate communication since it uses broadcast and multicast messages. It is a promising characteristic to be explored when considering organizations of physical objects using MAS. However, in the scope of this thesis when considering the agent approach, ContextNet is used to create a new type of agent able to communicate with other entities, including other agents. This new agent will use a client instance of ContextNet, and it will be responsible for all inbound and outbound communications related to this MAS with other entities. As it is a ContextNet client, it will be uniquely identified in the IoT network. Hence, when this agent is embedded in Smart Things, it will identify this object uniquely in the network, providing the necessary connectivity and communicability for them to interact in open environments.

The Mobile Hub extends the capability of communication in ContextNet by running in mobile devices that adopt the Android operating system. It provides mobile communication and data processing extending the cloud SDDL middleware. It discovers nearby objects enhanced with wireless technology for short distance as Bluetooth, BLE, and NFC. As the Mobile Hub is a client from ContextNet, it is working as a bridge between these objects and the core of ContextNet, opportunistically connecting to those objects to transfer data to the cloud. Since Bluetooth has a distance range for objects to connect to the mobile device, we decided to employ wired connections to sensors and actuators, and we use embedded systems running ContextNet clients for managing these resources. In this way, it is possible to create Things and Smart Things capable of controlling resources that do not depend on distance since they are all connected in a single object. Besides, it is possible to adopt cognitive ability and reasoning in the case of Smart Things, which employs MAS.