# USE OF GOALS FOR CREATING AND ENACTING DYNAMIC CONTRACTS IN AMBIENT INTELLIGENCE

by

Ayça Işıksal

B.S., in Computer Engineering, Boğaziçi University, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2011

# ACKNOWLEDGEMENTS

First, I am very grateful to my thesis supervisor Assoc. Prof. Pınar Yolum for her guidance and support from the preliminary to the concluding level of this research. I would also like to thank my committee members Dr. Suzan Üsküdarlı and Assoc. Prof. Şule Gündüz for their worthful comments.

I am indebted to MAS Research Group members: Özgür Kafalı, Akın Günay, Reyhan Aydoğan and especially Fatma Başak Aydemir who supported me to deal with the challenges where I got stuck. I would like to thank them for their patience and interest in giving their valuable ideas.

I would also like to thank many friends who were always there when I needed them. I have special thanks to Başak Tunçal, Serdar Ayun, Elif Akan, Emir Baş, Betül and Mehmet Usta.

This thesis could not have been accomplished without my husband, Soner Işıksal who always believed in me no matter what I decided to do. Feeling his love and support has always made me stronger when things got even worse than I thought.

I owe my deepest gratitudes to my parents, Mesut and Sevgi Kundak and my brothers Fatih and Hakan Kundak for their continuous and unconditional support that I have always felt.

This work is accepted to be presented in the International Conference on Ambient Systems, Networks and Technologies (ANT'11).

This thesis has been supported by Bogazici University Research Fund under grant BAP5694 and by the Scientific and Technological Research Council of Turkey (TUBITAK) 2210 National Graduate Scholarship Program.

# ABSTRACT


## USE OF GOALS FOR CREATING AND ENACTING DYNAMIC CONTRACTS IN AMBIENT INTELLIGENCE

Ambient Intelligence (AmI) systems support everyday lives of humans by sensing and reacting to the environment when necessary. Devices of the AmI world cooperate with each other to achieve a common task by using the information they capture from the environment and through their internal knowledge. They make their decisions based on their reasoning processes and strategies. To increase the satisfaction of the user, several types of applications cooperate with each other such as audio, visual and health care related applications. This cooperation is hidden from the user.

This thesis proposes to view AmI devices as autonomous agents with different capabilities and goals and the entire AmI system as a distributed multiagent system. Goals refer to the states that the agent desires to achieve. The proactive behavior of the AmI devices are modeled with the goals of the agents. Each agent within the system is motivated to satisfy its goals. The agent selects actions that will have positive consequences on satisfying its goals. The agent decides and initiates relevant actions to satisfy its goals one by one. If the agent has the capability to achieve its goal, it just executes it. If the agent's capabilities are inadequate for satisfying its goal, then it asks for help from other agents. Agents use contracts to interact with each other, which are regulated with a commitment-based methodology. We provide algorithms and reasoning rules to help agents create and respond to the commitments in different situations. We apply our approach on an intelligent kitchen domain consisting of smart kitchen devices such as a coffee maker, a refrigerator and a user agent representing the resident of the kitchen. We demonstrate how our approach can accommodate several realistic cooperation scenarios.

# ÖZET

## ÇEVRESEL ZEKA ORTAMLARINDA DİNAMİK SÖZLEŞMELERİN YARATILMASI VE UYGULANMASI İÇİN AMAÇLARIN KULLANILMASI

Çevresel zeka sistemleri, gerekli olduğunda bulundukları ortamları duyumsayarak ve bu ortamlara tepki vererek insanların günlük hayatlarına destek olurlar. Çevresel zeka sistemlerindeki araçlar, ortamdan aldıkları bilgiler ve içsel bilgileriyle, ortak bir görevi yerine getirmek için birbirleriyle iş birliği yaparlar. Bu araçlar, kararlarını kendi değerlendirme süreçlerine ve stratejilerine dayanarak verirler. İşitsel, görsel, sağlıkla ilgili ve çeşitli uygulamalar birbirleriyle etkileşip, kullanıcının memnuniyetini arttırmak için birlikte çalışırlar. Araçlar arasındaki bu ortaklaşa çalışma, kullanıcıya açık değildir.

Bu tez, çevresel zeka sistemlerindeki araçları, her birinin kendilerine ait yetenekleri ve amaçları olan özerk etmenlerle temsil eder. Bu araçların oluşturduğu sistem de dağıtık çok etmenli sistem olarak ele alınır. Amaçlar, etmenlerin ulaşmayı arzuladıkları durumları gösterir. Çevresel zeka ortamlarındaki araçların proaktif davranışları, amaçlar ile modellenir. Her etmen amaçlarına ulaşmaya motive olmuştur ve kendisini amaçlarına yaklaştıracak eylemleri gerçekleştirir. Etmen, eğer kendisi amacını gerçekleştirecek yeteneğe sahipse bu yeteneğini kullanır; aksi halde de diğer etmenlerden yardım ister. Bu iletişim, etmenlerin birbirleriyle sözleşme yapmasıyla oluşturulur. Bu sözleşmeler taahhüt tabanlı yöntem ile düzene sokulur. Biz, etmenlerin gereken durumlarda diğer etmenlerle taahhüt oluşturmaları ve kendilerine yapılan taahhütlere anlamlı cevaplar verebilmeleri için algoritmalar ve kurallar geliştirdik. Geliştirdiğimiz sistemi, kahve makinesi, buzdolabı gibi araçları ve kullanıcıları temsil eden etmenlerden oluşan akıllı mutfak alanında uyguladık. Son olarak da bu yaklaşımımızı, var olan etmenler arasında olabilecek gerçekçi senaryolar üzerinden doğruladık.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONYMS/ABBREVIATIONS

| | |
|---|---|
| ACC | Agent Communication Channel |
| ACL | Agent Communication Language |
| AmI | Ambient Intelligence |
| AMS | Agent Management System |
| BDI | Belief–Desire–Intention |
| CM | Coffee Machine agent |
| DF | Directory Facilitator |
| F | Fridge agent |
| FIPA | The Foundation for Intelligent Physical Agents |
| GPS | Global Positioning System |
| JACK | Java Agent Component Framework |
| JADE | Java Agent Development Framework |
| KAOS | Knowledge Acquisition in Automated Specification |
| MAS | Multiagent System |
| OSGi | Open Services Gateway initiative |
| PDA | Personal Digital Assistant |
| PRS | Procedural Reasoning System |
| RFID | Radio Frequency Identification |
| UA | User Agent |
| 3APL | Artificial Autonomous Agents Programming Language |

# 1. INTRODUCTION

Increasingly, everyday devices operate proactively by sensing and reacting to their environment to support our everyday life, giving rise to the realization of Ambient Intelligence (AmI) [3]. Proactive behaviour is defined as self-initiated and change-oriented behaviour. In this case, devices are not simply passive components who wait for things happen to execute an action. The devices themselves, act in anticipation of their future needs or changes. These devices in an AmI context operate collectively using the information and intelligence embedded in the network of the devices. Applications such as audio, visual and health care related applications cooperate with each other to increase the satisfaction of the user. This cooperation is transparent to the user [3].

Ambient intelligence applications must sense their environment, reason with the information gathered and act as a result of this reasoning. Therefore, Ambient intelligence research comprises sensors and sensor networks, ubiquitous computing, and artificial intelligence [4]. Ambient intelligent environments have following features [5]:

- *embedded*: Several connected devices are integrated into the environment.
- *context-aware*: Devices can identify the user and adapt their behaviour accordingly.
- *personalized*: Devices can adjust themselves by providing different services and interfaces to different users.
- *adaptive*: Devices can learn to change their behavior in response to the user through learning.
- *anticipatory*: Devices can anticipate a user's desires through reasoning and without direct interaction.

Modern devices offer a large number of functionalities to their users. The designers of these devices generally expect users to combine and manage these functionalities as appropriate. Interestingly, most users are overwhelmed by the number and variety of the functionalities and rarely ever use them. We can say that technological devel-

opment in recent years have focused on the function-oriented context rather than the goal-oriented one. However, ambient intelligence aims to revert this trend to the goal-oriented progression in the sense that technology is for user's well-being. Well-being of the user refers to the satisfaction of the user in terms of health, happiness, and welfare. Besides, it aims to make the interaction between the user and the environment bidirectional, so that each device within the environment and the user can interact with each other [1].

We can easily conclude that ambient intelligence aims to improve lives by providing desirable behaviour. To accomplish this, ambient intelligence does not hold its users responsible for coordinating devices and managing the environment. Instead, interconnected and personalized devices behave intelligently to increase the satisfaction of the user [1].

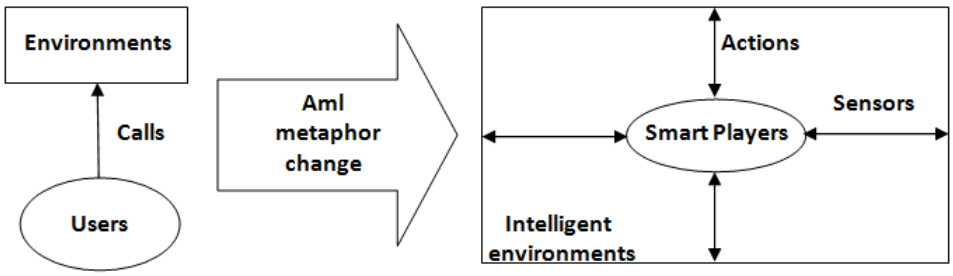


Figure 1.1. Change in the Context of the Technological Developments [1].

Figure 1.1 shows the change in trend thanks to the ambient intelligence technologies. On the left side of the figure, we see that in the function-oriented systems, environments are passive components. Users are the only active players and they use the environment in a relevant way to gather the functionalities they need. In an AmI system, all of the components are intelligent players who behave proactively with the use of sensors to sense the environment and actions they can perform certain actions.

AmI requires the following procedures to provide an intelligent platform for improving user's well-being in terms of health, happiness and welfare:

- The devices in an environment must be aware of the user's state and his/her

interactions with the environment.

- The devices must consider their state and react in a manner that serves the user's goal.
- The environment must translate the user goals into strategies that can be satisfied by the devices in the environment.

The cycle of that procedure is called *Principle Workflow Cycle of Ambient Intelligence*. Each step of the cycle leads to different types of challenges to be solved [1] such as context awareness, activity recognition, reasoning techniques and interaction patterns.

AmI environments can be represented with multiagent systems that will be described in next chapter. Multiagent systems are composed of several agents. An agent is a software process that perceives its environment, reasons on perceptions and acts to the environment based on its reasoning [6]. The agent can interact with its environment. Besides, it can also interact with other devices within the environment. In a multiagent system, there are several interaction patterns based on the context. Agents can communicate with each other to cooperate in order to achieve a common goal or to compete for individual goals.

Multiagent systems can be divided into two categories: closed or open. In a closed system, agents cannot leave or join the system. Contrarily, in an open multiagent system, agents within the environment are dynamic in a sense that agents can both join or leave the system.

Agents in a multiagent systems have two important properties [7]:

- *Autonomous* - independently motivated: Agents have a right to choose with whom, how and when to interact with.
- *Heterogeneous* - independently designed : Future behaviours of the agents cannot be predicted.

*Belief-Desire-Intention(BDI)* model is widely used for describing the agent's mental state. An agent is motivated to satisfy its goal. The agent can accomplish its goal by itself or by asking from other agents within the system. Therefore, goals of the agent are its main motivation to interact with other agents. These goals can be common for all of the agents within the system in which they all cooperate to satisfy a goal or each agent has a separate set of goals which can be compatible or in conflict with other agents'goals. The agents can cooperate or compete with each other if they have individual goals. Representation, processing and deliberation of goals play a crucial role for the agents to manage their behaviours. The agents can have several types of goals [8] which will be described in Chapter 2.

In order to cooperate, agents need to talk the same language when interacting. An agent that can only send and parse primitive commands cannot cooperate with another agent that sends more complex messages. Interaction protocols regulate the behaviours of the agents and define behaviour patterns to allow agents to cooperate with each other when trying to satisfy their goals [9]. Agents enter into contracts with each other to achieve their goals. Commitment-based protocols [10, 11, 9] are widely used as successful contractual interaction protocols among autonomous agents. Commitments are used for modeling the content of the interactions among the agents [12]. They cannot force an agent about when to send messages. This aspect of commitment protocols support the autonomy and heteregonity of the agents.

Commitments are the obligations between the agents [13]. Formally, a commitment is an expression which represents a debtor committed to a creditor to bring about some condition. Agents can create commitments, cancel them, and when an interaction is succesful discharge them. A commitment has a state and its state is changed based on the actions performed by the parties involved. Thus, the commitments are dynamic in terms of their states.

Most existing models and applications of AmI consider that devices have predefined behaviors in how they realize certain tasks. The devices have predefined interaction patterns; for example, a toaster in a kitchen can sense that the user has inserted

bread, which it communicates with a coffee maker that then makes coffee. While such cooperation patterns are definitely useful, more interesting applications of AmI will take place when the devices dynamically cooperate. The dynamism is crucial for various reasons. First, an AmI environment is an open environment, where new devices will enter and existing devices may break down. The devices themselves should be able to accommodate the entrance and departure of others into the system. Second, no matter how carefully designed, a cooperation may not produce expected results because of a broken device, a cheating agent, and so on. Assume that the sensor in the toaster sometimes mistakenly notifies the coffee maker about the toasting of a bread. If the coffee maker can detect this mistake, there is no reason for it to accept orders from the toaster the next time. Third, the environment can learn how to help a user in the system. For example, if the user never actually takes coffee, the devices should figure out that they should not be brewing coffee just because the user is toasting bread, even though that particular interaction is enforced on the devices.

*Contributions:* In this thesis, we model an ambient intelligence environment as a multiagent system in which devices are represented as autonomous and heteregenous agents. Furthermore, the user of the system is also modeled as an agent. Therefore, each agent represents a device or user of the system. Agents have individual goal sets: they are wishing to achieve certain states or they have to maintain some of their states during their lifecycle, or both. On the other hand, they have certain capabilities that refer to the services they can serve. Goals and capabilities of the agents need not to be identical. Agents are motivated to satisfy their goals. If an agent can realize its goal on its own, then it does so. However, in many realistic settings, an agent will need the help of others to realize a goal. Since the agents are autonomous, they cannot be told what to do but can only be requested to do a task. Moreover, because each agent represents a device with possibly different (or even contradictory) goals, there are no guarantees on whether they will do a task. For example, a coffee maker cannot assume that a toaster is required to notify it about its actions. The coffee maker can request the toaster to do this and only if the toaster *commits* to notifying, the interaction will take place. The problem of cooperation then can be seen as finding appropriate agents that have the necessary capabilities in realizing an agent's goals and are willing to

commit to performing these tasks.

However, current commitment-based multiagent applications provide the commitment specifications before run time to all agents [14]. While this approach is useful in many applications, we propose the dynamic creation of commitment specifications that is more realistic and flexible. The agents themselves, based on their current goal states and capabilities will decide in which commitments they can be engaged that makes the commitment based approach possible to realize open and autonomous system such as ambient intelligence. It does not mean that our approach is just for ambient intelligence, our solution is generic so it can be also applied in different domains as well. For example, in an e-commerce domain, agents that are at the beginning of their interaction will need to figure out a way to cooperate. Our approach makes it possible for the agents to decide which commitments they can honor, what other commitments they would need from other agents, and so on. Thus, agents are able to generate the interactions that are necessary for the cooperation at run time. Based on the needs of the agents, the contract needs to be generated dynamically.

The rest of this thesis is organized as follows: Chapter 2 provides current studies related to ambient intelligence and methods applied for the realization of the ambient intelligence applications. Several projects are investigated for this purpose. Another main aspect of our study is agents' goal-oriented behavior. Thus, literature for goal-based agents is provided. Some agent programming frameworks in which the agent executes depending on its plans generated to satisfy the agent's goal are discussed. Lastly, a number of AmI applications which use multiagent systems for the representation and process of the devices and components are explained. The entire related work is discussed with our approach. Chapter 3 gives some background information about the main aspects of the agents: goals, capabilities and commitments. Goal types that we choose to adopt for agents, their goal satisfaction methods and the relationship of their goals with their capabilities and commitments are described. Capabilities of the agents together with their properties are defined. Chapter 4 shows the architecture of our approach which constitutes the components that the agent contains. It also shows which types of messages agent can send and receive from each other. Besides, the rela-

tion with the message types and the commitment states is obtained. The behaviors of the agents in terms of the algorithms and rules are represented and explained. At the end, several scenarios among selected set of agents from the intelligent kitchen domain are demonstrated. Chapter 5 provides some examples in smart kitchen domain with User, Coffee Maker and Refrigerator agents. Each scenario contains these agents with different goals and capabilities and explains the behaviours of the agents. Besides, this chapter gives a technical information related to the application we have developed. Lastly, Chapter 6 concludes our thesis comparing with the existing work and provide some future directions for our approach.

# 2. RELATED WORK

## 2.1. Ambient Intelligence Applications

There is a wide variety of existing AmI applications. Those applications fall into specific categories: smart homes, health assistance, transportation, education, hospitals, emergency services and workplaces [4]. In smart home domain, devices in the house have sensors to collect information and to act depending on it. Users profit from those devices in the sense of increased safety by monitoring the activities and giving assistance in an unusual situation, comfort by automatic adjustment of temperature, and economy by automatic lightening.

Additionally, the home itself can use its intelligence to make decisions based on its state and interactions with its users. For instance, MavHome [15] project collects sensor information for a period and deduces the patterns of the users' behaviour and models those patterns as a Markov process which in turns can predict the daily interactions approximately 80 % [16]. The University of Essex's intelligent dormitory (iDorm) [17] is a real AmI test-bed that constitutes large number of embedded sensors, actuators, processors and networks in the dorm rooms. In this application fuzzy rules are learned from the activities of the residents and they are used to manage the devices in the room [18].

Health-related applications consist of home assistance to provide independent lives to the people with physical or mental challenges. Such systems include the activity recognition [19], monitoring of daily exercises, detecting anamolies and changes. Hospitals can make benefit from those services in a way that they can monitor the patients' health status and treatment progress by analyzing the activities they perform at their homes.

Smartcard technology is used in high schools and universities to authorize students for entering libraries, dining and dorm rooms, lectures and car parks. Continuous

tracking of student activities provide huge amount of data that can be used for various reasons, e.g. analysis of the student activities, taking attendance in lectures, and so on. In the scope of the emergency services, ambulances or fire cars can use the GPS based location services and traffic signals to decrease their reaction time to emergency situations.

As it is understood by the application domains, AmI system has a real environment in which the residents communicate with each other or with the environment itself. Formally, AmI can be defined as follows [2]:

$$AmI : \langle E, IC, I \rangle$$

such that

$E$ is the environment like school, home, hospital, etc.

$IC$ is the set of interaction constraints. It defines the way that the components of $E$ and $I$ can communicate with each other. $IC$ is a set of $\langle S, A, C, IR \rangle$ in which $S$ is the set of sensors, $A$ is set of actuators, $C$ is set of interaction contexts and $IR$ is the set of interaction rules. $S$ gathers the information, system reacts to the environment via $A$, $C$ specifies the conditions for the system to react to the environment and finally $IR$ combines whole components to decide how to act.

$I$ is set of interactors who can benefit from the system, e.g. people, pets, companies. They can communicate with the system based on the rules defined by $IR$.

In Figure 2.1, a generic AmI system is presented. Sensors capture the information from the environment and pass it to the system middleware. The middleware handles the communication issues and convert the sensor data to a more structured form for the reasoning process. AI Reasoning decides the action to be taken with the help of knowledge repository, learning mechanisms and decision makers. Finally, the action is

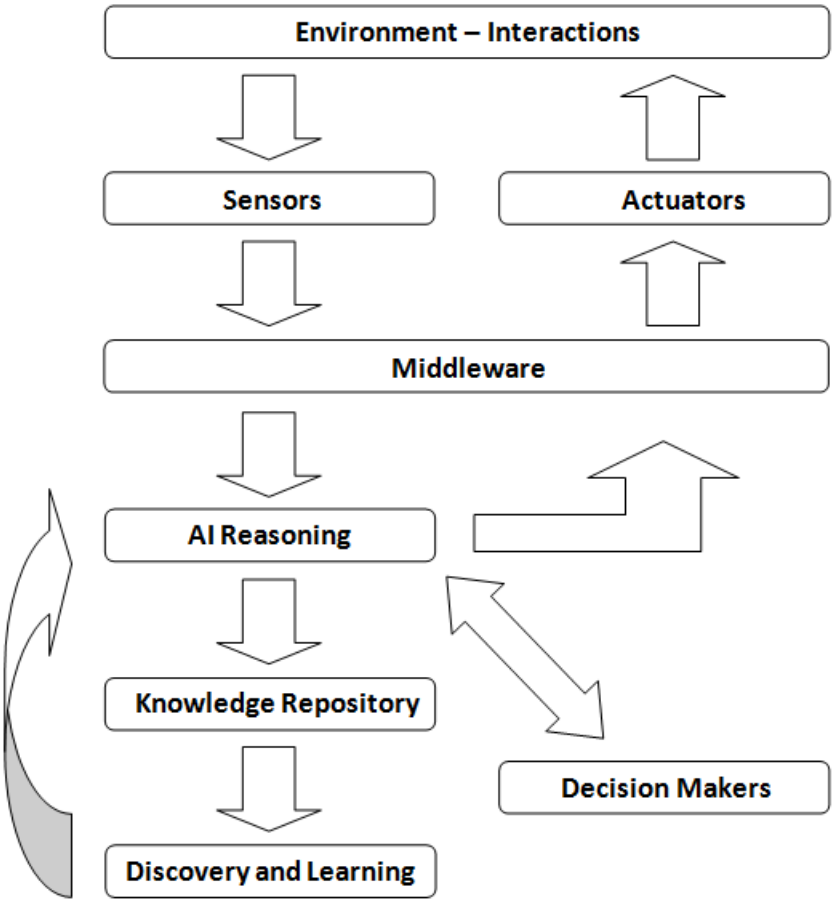


Figure 2.1. Architecture for AmI Systems [2].

executed through the actuators.

Sensors and actuators connect the physical environment and the system framework. Sensor technology has been studied a lot and various sensor technologies have been developed. Capturing, collecting, and analyzing the sensor information is somewhat challenging. Also there are several EU projects for developing AmI frameworks like Amigo and Persona. Amigo [20] project aims to develop both the framework and user applications for home safety, information, and development. Amigo adopts service-oriented architecture in which the software is delivered as services that are published and consumed on demand. Service-orientation makes the addition and removal of components easy. OSGI [21] is used for publishing and executing services. Amigo provides some parts of its architecture as open source so everyone can develop their own Amigo services. We have previously developed a smart living room with Amigo

framework. In our smart living room, we have coordinating and cooperating smart agents such as telephone, television, DVD player, and also some services other than everyday devices like the last.fm agent and google calendar agent. Each device has a set of properties that are shared among each other using an ontology. Examples for the properties of a device are: priority, having a sound, devices it needs to work properly. They all work in accordance with each other based on their key properties. For instance, if a phone starts to ring, all devices are alerted and those with lower priority that are active and have sound, mute themselves without user interrvention. When a DVD player that needs television to work is turned on, it checks whether the television is also on. If the television is off, the system turns the television on so that the DVD player can work. Cooperation aspect of the project involves two agents: last.fm and google calendar. They cooperate with each other to compose a list of songs based on the mood of the user extracted from the daily activities and the type of the listened music. From the last.fm account of the user, the types of the songs the user listened during the day are extracted. Google Calendar of the user provides daily events. By combining those information, we extract the mood of the user and select the type of the music to be played based on that mood. For example, if the user's mood is extracted as angry we play classical music; if the mood is happy we play pop music, and so on.

Artificial Intelligence can be used to make devices act autonomously and intelligently. There are various ways that the system should act intelligently [2]:

- *Activity recognition and learning:* System should group the information captured from the sensor to determine the activity that the user is performing. For instance the system should know that user is cooking from the sensors in the kitchen and on devices like the oven and refrigerator [22].
- *Context Awareness:* System should understand the context and evaluate it to operate successfully for the benefit of the users [23]. Time, location, and mood of the user are examples for contexts.
- *Reasoning:* It is necessary for the system to infer which action is took. Different views exist like rule-based, biologically inspired, etc.
- *Multiagent systems:* For the sake of the autonomy and flexibility of the system

multiagent systems play a crucial role. However, traditional agent systems are not sufficient for the dynamism AmI environments should obtain to their users. Multiagent representation of the AmI systems will be further investigated in next sections.

## 2.2. Goal-Oriented Agents

Software agents are proactive to their environments; their proactive behaviour is represented as goals [24]. Agent programming frameworks in which goals of the agents are crucial are called cognitive agent programming frameworks. In these frameworks agents are modelled with their beliefs, goals, and plans. Based on its beliefs, an agent tries to achieve its goals by executing its plans.

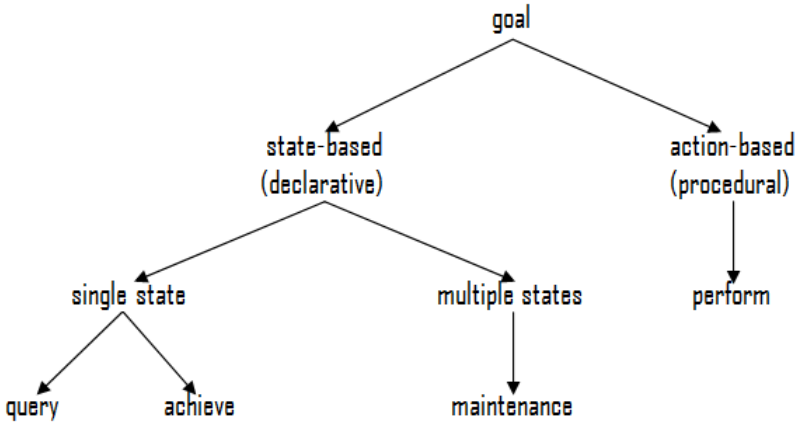


Figure 2.2. Goal Taxonomy.

There are several types of goals used in agent programming and the taxonomy of goals is shown in Figure 2.2 [25]. *Declarative goals* mean reaching a certain state whereas *procedural* goals mean performing a certain action independent of the state evolved at the end. *Achievement*, *query*, and *maintenance* goals are declarative goals. An agent must reach a desired state to satisfy an achievement goal whereas it must preserve the desired state to satisfy its maintenance goal. Brewing coffee and storing it ready to be served are achievement goals for a Coffee Maker whereas keeping its water level above 80 % is its maintenance goal. *Perform* goal is to execute actions like buying a book and *query* goal is for gathering information such as retrieving the temperature of the room. Another distinction is between *system* and *individua*l goals. System goals

are adopted by all the agents whereas individual goals are specific to the agents, and probably hidden from each other.

Another aspect regarding to the representation of the goals are relationships between them. A goal of an agent can be composed of two distinct subgoals. In such a case, the agent can decompose its goal during the achievement procedure. This decomposition can occur in two different ways: AND and OR decomposition [25]. In AND decomposition, the agent must satisfy all subgoals. However, for the OR decomposed goals, the agent must satisfy only one of the subgoals.

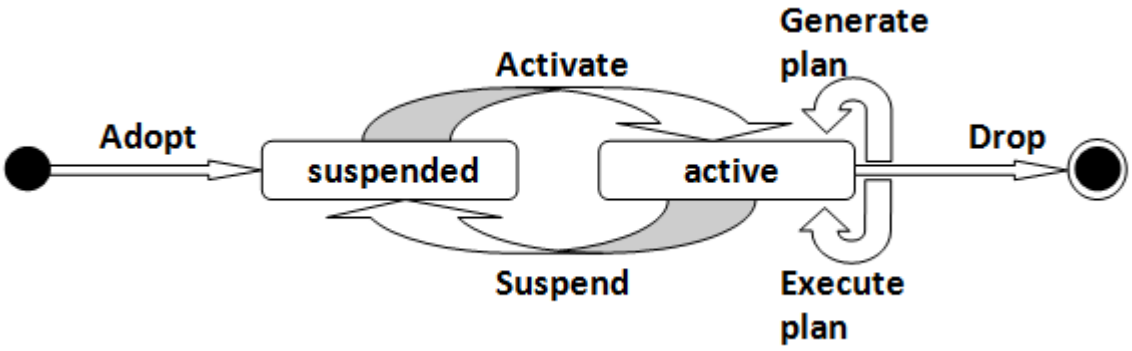


Figure 2.3. Goal Life Cycle of an Agent.

Goals of the agents can be active or suspended [25]. When an agent adopts a goal it becomes suspended until it is activated. Adversely, the agent can suspend an active goal. The agent can drop its active goal if the goal is achieved or becomes irrelevant for the agent. Figure 2.3 shows the state diagram of a goal of an agent.

There are various agent programming frameworks in which agents execute plans to achieve their goals. Some examples are: Jadex [26], JAM [27], AgentSpeak(L) [28], JACK [29], and 3APL [30]. They all have a library of plan recipes defining in which circumstances and for which goal the plan can be used. Those frameworks share the same problem: The agents have predefined behaviour patterns in order to achieve their goals. Goals that the agents can adopt and the actions they can execute are limited by the system specification. The agents should select a goal and a relevant action set (i.e. plan) from the pool. These frameworks are inadequate for fulfilling the dynamic structure of the ambient intelligence environments and they are usually for single and

primitive behaving agents like a room cleaning robot.

Table 2.1. Goal Types in Different Platforms.

| - | KAOS | Gaia | JACK | PRS | JAM | JADEX | Our system |
|---|---|---|---|---|---|---|---|
| achieve | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| maintain | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ |
| perform | - | - | - | - | ✓ | ✓ | - |
| query | - | - | - | - | ✓ | ✓ | - |

Although there exists several types of goals, existing agent development platforms do not use all of them. Table 2.1 shows some platforms and the goal types that the agents can adopt. Achievement and maintenance goals have been used in all of the platforms discussed. Currently, we also include these types of goals in our system but other types can also be added as a future work.

## 2.3. Representation of AmI Environments with MAS

The role of the multiagent systems in Ambient intelligence is briefly discussed before [31]. According to Cook, there are four distinct approaches for the usage of multiagent systems in developing AmI applications. These approaches are designing the AmI environment as a multiagent software, tracking multiple users, profiling activities and behaviours for the users, and negotiation of the agents' needs and preferences. For the first approach, there are three important aspects to be considered: the role of the agents, the organization between the agents, and the interaction and cooperation protocols among the agents.

One of the studies [32] uses wireless technology for the interaction among the agents. In this study, there is a three layer architecture for the agents: *bus layer, function layer and management layer*. Agents in bus layer connect sensors and actuators. Function layer agents contribute to the system in various ways such as identifying users and their situations. Lastly, management layer agents are responsible for the management of the whole system. This type of hierarchical definition of the agents is also used

in Mavhome [15] project. In this project, the system is composed of physical layer, communication layer, information layer, and decision layer. Rather than defining agent roles depending on the functionality, space issue can be used for specifying agents [33]. In this approach, agents are defined dependent on the regions of the environment they reside.

ALZ-MAS [34] is an Ambient Intelligence based distributed multiagent system that aims to improve the assistance and health care for Alzheimer patients in geriatric residences. Agents of the ALZ-MAS are designed as *Belief Desire Intention (BDI)* [8] agents. The motivation for this study comes from the fact that people with disabilities need more attention than healthy people in a sense that their activities, situations and their health status must be monitored and tracked continuously. An ambient intelligence system should supervise the patients during their treatments. The main contributuions of the system are developing BDI agents that have reasoning and planning capabilities and using some context-aware technologies to capture information from both the environment and its users. ALZ-MAS consists of the following types of the agents:

- *User Agent* represents the patients of the system. There is exactly one *User Agent* for each patient. This agent's role is to manage the patient's personal data and behaviour. Beliefs and goals of the agent are defined by the super-user agents. *User Agent* must ensure that all activities assigned by the super-user are performed. It sends a copy of its memory that includes goals and plans to the Admin Agent for redundancy.
- *Super-User Agent* represents the doctors of the system. There is exactly one *Super-User Agent* for each doctor. This agent can also work on PDAs. It inserts new tasks to *Admin Agent* to be reasoned and also it can assign tasks to *User Agent* and receive updates from it.
- *ScheduleUser Agent* represents the nurses of the system. There is exactly one *ScheduleUser Agent* for each nurse. It is a BDI agent with a planning ability. It schedules users' activities and provide dynamic plans based on the tasks assigned to the users, users' profiles, available resources and time.

- *Admin Agent* has two important responsibilities that are security role that monitors the users situations and environment through *Devices Agent* and management role that deals with database and task assignment issues. There is exactly one *Admin Agent* in the system.
- *Devices Agent* controls the hardware of the system. It captures information from sensors and actuators, controls the states of the devices, and report to *Admin Agent*. There is exactly one *Devices Agent* in the system.

ALZ-MAS uses RFID and wireless technologies for location awareness and communication issues. In ALZ-MAS and other works mentioned, each agent has a specific and strict role. They have predefined roles and rules for communicating. For instance, a User Agent cannot communicate with Devices Agent. Also their communication is usually in one way. They cannot enter in contracts or dialogs with each other. This structure is good for strict domains like health-related applications in which each agent has a predefined role and it cannot be changed. Also in such domains, agents do not need to negotiate with each other since each role is willing to and obligated to do their tasks. Agents of these systems cannot behave in accordance with the changes in the environment. They only follow the rules and patterns defined earlier.

Tracking multiple residents is a very challenging issue and there are few studies about it. Users can carry a device to be identified but it is very impractical [35]. However, there are some practical solutions such as identifying users with their weight profile using the pressure sensors on the floor [36]. But this approach does not work properly if two users have similar weights. Another study uses combination of vision and sound of the users to track the users [37].

Aydemir [38] also develops an AmI application using a multiagent system. She adopts a user-centered design that focuses on the demands and preferences of the user. Her motivation is consistent with the AmI systems by being human-centric. User is also represented as an agent called UserAgent (UA). The agents other than the UA cooperate with each other to satisfy the user's needs. Contracts are generated between the UA and other agents based on their internal states. It's the agents' decision to

involve in contracts or not. The agents have their own inventories and resource states but they do not have goals. An agent's responsibility is to satisy the needs of the users. Agents that represent the AmI devices provide services to the users and they can only interact with UserAgents. While this approach is well-suited for user centered environments such as AmI, other domains may not be fully represented with it such as an e-commerce application in which the system is all distributed and all agents' goals have equal importance. The sender, buyer, and deliverer agents exist in an example e-commerce system and the system cannot pay more importance to one of them.

Agent development frameworks are widely used especially for goal oriented agents. They aim to provide interoperable multiagent systems with standardizing interaction protocols, agent architecture and lifecycle, and agent execution model. JADE [39] is one of the mostly used frameworks for agent development. We have developed and executed our agents with JADE just to ignore the low level details of the agent development and messaging.

# 3. TECHNICAL FRAMEWORK

We have developed our system as a distributed multiagent system. Each agent represents a device of the ambient intelligence environment. There does not exist any central agent that has control over other agents, assigns tasks to them or manages how they can accomplish these tasks. Each agent independently makes its own reasoning and decides the actions that it will perform based on its reasoning. The reasoning of the agent is based on its own *commitments*, *capabilities*, and *goals*.

The capabilities of the agent specifies what the agent can perform by itself without interacting with other agents and getting help from them. For example, Coffee Maker is designed to make coffee and it can do it by itself assuming that the resources it needs (i.e., water, coffee beans, electiricity) are available. Capabilities of the agents can change over time. Having a particular capability does not imply that an agent will carry out the tasks to realize this capability. For example, based on its internal states and decisions (e.g., a strategy of an agent to consume the existing goods instead of making a new one), Coffee Maker can prefer to serve cappuccino instead of latte if it has already made the cappuccino.

The goals of the agent are its motivation to behave proactively in its environment. Its aim is to satisfy its goals, and to realize that, it needs to perform some actions, either by executing its own capabilities or participating in contracts with other agents that reside in the environment. The goals of an agent can change over time. The agent should rearrange its actions to achieve its new adopted goals.

The commitments of the agent represent its contracts with other agents. Since the agent's aim is to satisfy its goals, it interacts with other agents and participates in commitments to satisfy its goals that cannot be achieved by the agent alone, i.e., it does not have the relevant capability to achieve that goal. The agent stores its entire history of commitments to be used in the future decisions. Each commitment has a state, e.g., requested, active, and fullfilled. The state of the commitments helps the

agent to use the relevant commitments during its decision making processes. In the following sections, some technical details and the formalizations of the commitments, capabilities, and the goals will be given.

### 3.1. Commitments

**Definition 3.1.** *A commitment CC(x, y, q, p) where x is the debtor agent, y is the creditor agent, q is the condition of the commitment, and p is the proposition of the commitment means that the agent x is committed to the agent y to satisfy the service p if the service q holds [40]. If q holds, x must satisfy p. When q holds, commitment state is changed to be detached. If p is achieved, commitment is discharged, and ceases holds.*

The following is an example commitment between the agents Coffee Maker and User. Users of the system are represented with User Agents.

**Example 3.2.** CC(CoffeeMaker, UserAgent, water, coffee) means that the Coffee Maker commits to the User Agent that if the User Agent provides water to it, then the Coffee Maker will serve coffee to the User Agent.

There are basically two types of commitments used in the literature: base-level commitment and conditional commitment.

**Definition 3.3.** *CC(x, y, T, p) is a base-level commitment between the debtor agent x and the creditor agent y for the proposition p [12]. Condition of the commitment is T (constant for truth). When this commitment is created, the agent x becomes committed to the agent y to satisfy the proposition p. Agent y does not have to satisfy any conditions to the debtor agent x.*

**Definition 3.4.** *CC(x, y, q, p) is a conditional commitment between the debtor agent x and the creditor agent y [12].*

There are some specific actions that can be executed on commitments [10]. Those are:

- *create(x, y, q, p):* This operation is performed by the agent $x$. The commitment *CC(x, y, q, p)* is created and it starts to hold.
- *cancel(x, y, q, p):* This operation is performed by the agent $x$. The commitment *CC(x, y, q, p)* ceases to hold.
- *release(x, y, q, p):* This operation is performed by the agent $y$. $y$ releases $x$ from the commitment. The commitment *CC(x, y, q, p)* ceases to hold.
- *delegate(x, y, z, q, p):* This operation is performed by the agent $x$. The commitment *CC(z, y, q, p)* starts to hold but *CC(x, y, q, p)* ceases to hold.
- *assign(x, y, z, q, p):* This operation is performed by the agent $y$. The commitment *CC(x, z, r, u)* starts to hold but *CC(x, y, q, p)* ceases to hold.

Each commitment has a state [14]. Those states are:

- *requested:* the commitment is requested by one of the parties and the other party has not answered yet.
- *active:* requested commitment is accepted or confirmed by the other party.
- *fulfilled:* the commitment is discharged since the proposition of the commitment is satisfied.
- *rejected*: the requested commitment is rejected and it no longer exists.
- *canceled:* one of the parties cancels the commitment. Debtor agent cannot cancel an active commitment since it already becomes committed to satisfy the proposition.

Following Example 3.2, when the Coffee Maker creates the commitment *CC(CoffeeMaker, UserAgent, water, coffee)* and sends it to the *User Agent*, the commitment state becomes *requested*. If the *User Agent* accepts the commitment request and performs the condition, i.e., if the Coffee Maker obtains *water*, the commitment state is changed to *active* and it turns to be a base level commitment: *CC(CoffeeMaker, UserAgent, T, Coffee)*. From now on, *Coffee Maker* becomes committed to serve *coffee* to the *User Agent*. Whenever the *Coffee Maker* brews *coffee* and serves it to the *User Agent*, commitment is finally *fulfilled*. When the commitment is requested to the *User Agent*, if it rejects the request, the commitment's state is changed to be *rejected*.

*Coffee Maker* can cancel the commitment before the commitment is *active*. Besides, *User Agent* can cancel the commitment at any time. Then, the commitment's state is switched to be *canceled*. For instance, if the *Coffee Maker* requests a commitment including *water* from two different agents, when one of the commitments is accepted, it may cancel the other one. However, if both of the commitments are accepted and the creditors of the commitments accomplish their tasks to provide *water*, *Coffee Maker* can no longer cancel any of the commitments because it becomes committed to satisfy the proposition (i.e., to make coffee).

**Definition 3.5.** *Given that, $\mathcal{A}$ is a set of agents in the system, $\mathcal{T}_a$ is the set of commitments that $a \in \mathcal{A}$ is either the debtor agent or the creditor agent for every commitment $t \in \mathcal{T}_a$.*

## 3.2. Capabilities

An agent has a set of targeted capabilities such that each capability has source and destination agents. If an agent is capable to serve a service, it does not necessarily mean that all agents in the system can benefit from it. For instance, the Coffee Maker may be willing to serve coffee to the User Agent whereas it does not want to serve it to the Refrigerator. Also there can be discriminations over the User Agents. Coffee Maker has an opportunity to serve its coffee to one of the users that requests coffee from it, depending on its previous interactions or internal preferences, especially if the Coffee Maker has a limited coffee to be served. Another possibility is that, in health-related applications, Coffee Maker may have knowledge that a specific user is forbidden to drink coffee. When that user asks for coffee to the Coffee Maker, even it has coffee, or it can brew immediately, it rejects the user's request. This reasoning can be carried out with targeted capabilities. If the destination agent is not specified by the service provider agent, all agents can benefit from that service.

Capabilities of the agents are dynamic so they can change over time. A capability can be introduced or removed, or its destination agents can be changed. We do not consider scarce resources. We assume that the agents have available resources to

execute their capabilities. For example, Coffee Maker always has water, coffee beans, and energy resources to brew coffee.

Agents can share their capabilities to enable the cooperation. The agents register themselves to a publisher agent of the system (i.e., agent providing yellow pages service) by specifiying their capabilities. This specification is common for all of the agents in order to understand the other agents' capabilities. However, it is agent's own preference to hide its capabilities. If it does not want to receive any requests for the service, it does not share that service. Additionally, the agents can unregister themselves from the publisher agent or they can change their current capabilities whenever they wish to do.

Another issue is that an agent can search for other agents in the system that have the capability the agent needs to achieve its goal. Publisher agent provides a search service in which the agent specifies its desired service and the publisher agent provides the list of the agents for the searched service. The agent uses these results to select an agent to create a commitment and request from. The selection among the agents can be random or can depend on its own reasoning. Therefore capabilities are one of the key points for the agent cooperation in our system.

**Definition 3.6.** *Action e is in the form of e(from, to, content) where, from field specifies the source agent of the action that is the agent performing the action, to field specifies the destination agent of the action. An agent performs an action if one or more of the agents in the system can benefit from it, including itself. All of the agents are included as a destination agent if this field is empty, content refers to the meaning of the action done. It can be a single proposition or a commitment. An agent can serve a service or it can manipulate a commitment by executing actions. E is the set of actions in the system.*

**Definition 3.7.** *$\mathcal{C}_a$ is the set of capabilities of an agent $a \in \mathcal{A}$. Each $c \in \mathcal{C}_a$ is an action that is formed by three fields in which the to field is empty or a set of agents and from field is the agent a itself.*

### 3.3. Goals

As mentioned earlier, there are several types of goals used in existing goal-oriented multiagent systems. We have simplified the taxonomy and selected a subset of the goal types. Each agent in our system has some individual achievement and maintenance goals. Currently, we do not have any system goals related to the environment. Since our approach involves distributed and independent agents, they do not share common goals.

An agent can achive its goal in three ways:

- If it has the relevant capabilities to satisfy the goal itself, it can do so without interacting with anyone else. It just executes its capability and satisfies its goal.
- If its capabilities are inadequate to satisfy its goal, it asks for help from another agent that can satisfy the goal. The agent generates a relevant commitment and requests it from the other party. If the request is acceptable by the other agent, it satisfies what it is committed to, i.e., the goal of the first agent.
- Following the previous case, even though the agent makes a commitment request, the other party doesn't immediately accept the request but asks for something in return. If this is acceptable for the first agent as well, the commitment is created. In this case, the first agent must satisfy the condition of the commitment that is desired by the second party. After the condition is satisfied, the second party becomes committed for the goal of the first agent.

**Definition 3.8.** *$\mathcal{G}_a$ is the set of unaccomplished goals of an agent $a \in \mathcal{A}$. Each $g\prime \in \mathcal{G}$ is the non-empty set of propositions or commitments such that $g\prime$ does not hold.*

**Definition 3.9.** *$\mathcal{S}_a$ is a set of satisfied goals of an agent $a \in \mathcal{A}$. Each $g\prime \in \mathcal{S}$ is the non-empty set of propositions or commitments such that if $g\prime$ is a commitment, then it has been fulfilled and if $g\prime$ is a proposition, then it holds in the world.*

**Definition 3.10.** *$\mathcal{M}_a$ is a set of maintenance goals of an agent $a \in \mathcal{A}$. Each $g \in \mathcal{G}$ can be a proposition. Instead of negating the propositions, we adopt the view that each $g \in M_a$ represents the states that the agent tries to avoid instead of maintaining.*

# 4. GENERATING AND ENACTING COMMITMENTS

As we have described in the previous sections, an agent first chooses to execute its capabilities to satisfy its goals. When the agent's capabilities become inadequate to achieve its goals, it interacts with other agents in the system and tries to convince them to serve its goal. Interaction among agents are handled with contracts among the agents and we have used commitments to represent these contracts. The agent can generate a base level and conditional commitment as a creditor or debtor. It requests a commitment from the other party. If the request is accepted, the relevant actions must be executed by both of the parties to fulfill the commitment. If the commitment is rejected, the agent can choose another agent to cooperate with or it can drop the goal and adopt a new one.

## 4.1. Architecture

Figure 4.1 shows the architecture of the system and the messages allowed between two agents.

In this figure, there are two agents that have separate goals and capabilities. Also they have separate sets of contracts that store the agent's history of interactions with other agents. Each agent has its own reasoner, service manager, contract manager, and goal manager. Agents register their services (i.e., capabilities) to the publisher agent.

*Service Manager:* Capabilities of an agent can be seen as the services the agent can serve to other agents in its environment. Each agent has a distinct set of capabilities, i.e., services. Service manager of the agent controls the agent's services, and prevents the possible conflicts when the capabilities are updated. It also registers the agent to the Publisher Agent and also unregisters when the agent decides to do.

*Goal Manager:* The agent has both achievement and maintenance goals that are controlled by its Goal Manager. Goal Manager selects a goal to be adopted from

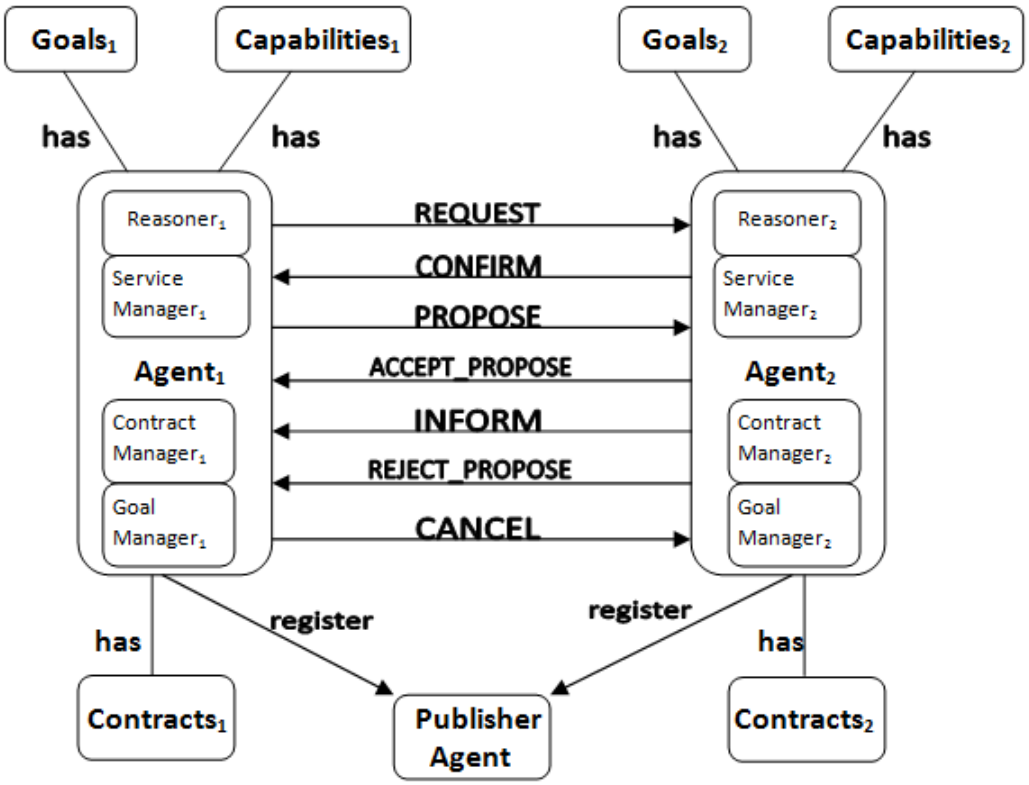


Figure 4.1. Architecture of the System.

the agent's unsatisfied goal list and reports it to the Reasoner of the agent. This selection can be realized randomly or depending on a strategy, e.g., precedence through priority. It also updates the unsatisfied and satisfied goal lists when an agent achieves a goal. Goal Manager is also responsible for the prevention of conflicts between the achievement and maintenance goals. It should not allow addition of a new goal to the agent's achievement or maintenance goal list, which conflicts with the existing goals of the agent. Otherwise, new goals are added to the relavant list by the Goal Manager.

*Contract Manager:* Contract manager has a list of contracts (i.e., commitments) that have been created during the lifecycle of the agent. The states of the commitments are modified by the Commitment Manager. Commitment Manager is also responsible for returning the queries initiated by the Reasoner of the agent related to the existing commitments.

*Reasoner:* Each agent has a Reasoner component, which performs the main

functionalities of the agents. With communicating with the agent's Service Manager, Goal Manager and Commitment Manager, Reasoner makes the reasoning and finds a way for the achievement of agent's goals. Only one of the achievement goals of the agent can be active at a time and the agent executes actions in accordance with its active goal. It also takes the agent's maintenance goals into consideration to avoid the possible conflicts. Goal Manager leads the Reasoner to select a goal from the agent's unsatisfied goal list. The Reasoner interacts with the Service Manager of the agent to ensure if the agent can achieve its goal alone without interacting with other agents. If the agent is not capable for that service, cooperation with other agents becomes requisite. Reasoner checks its commitment history via the Contract Manager before initiating a commitment with any agent. It explores the agents using Publisher Agent's service and decides with whom to cooperate for the specific goal.

*Publisher Agent:* This agent provides a yellow pages service. Each agent registers its services to the Publisher Agent if they want to share these services. When an agent needs another agent for a service, it uses the search service provided by the Publisher Agent to find the concerned agent.

### 4.2. Messages and Commitments

Commitments are created by an agent and evolve based on the messages sent and received between the agents. It is useful to discuss the possible evolutions of a commitment. Figure 4.2 shows the relation between the message types and commitment states.

Assume that the first agent has to cooperate with the second agent (so that it can satisfy its goal). It communicates to the second agent by a REQUEST message. The content of the message is a commitment. Within the commitment, the agent can both request a service or another commitment from the debtor agent. Example for the former case is that User Agent wants to drink coffee and since it does not have the capability to brew coffee, it should request it from another agent such as the Coffee Maker. Coffee Maker can accept or reject the request depending on its own decision

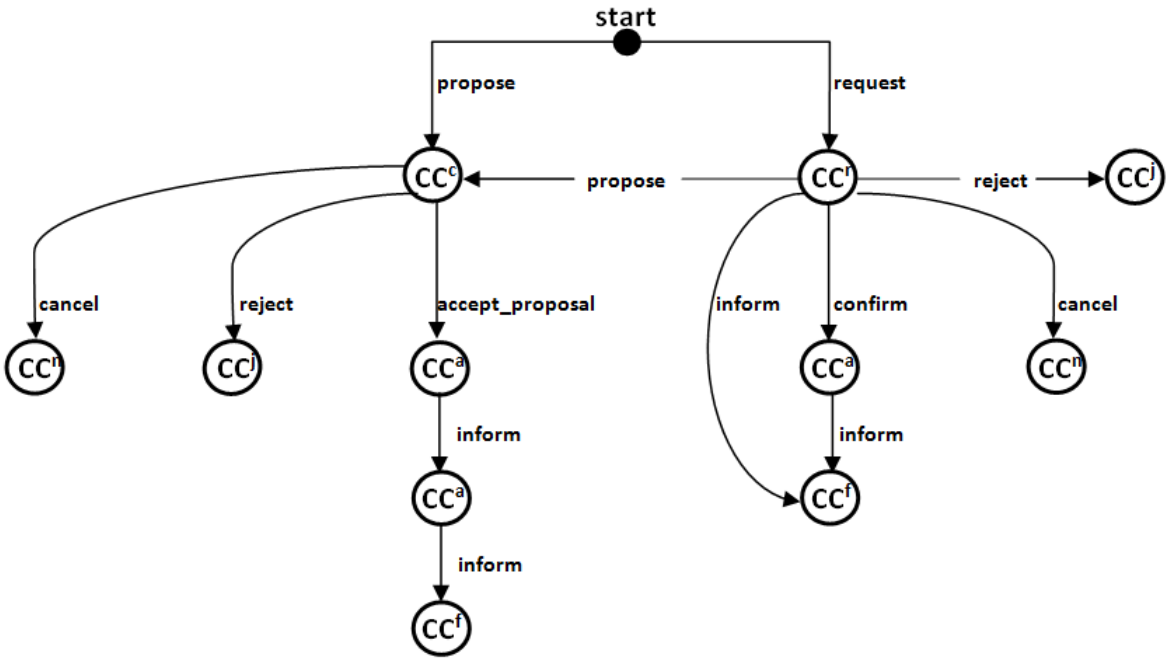


Figure 4.2. Relation Between Message Types and Commitment States.

mechanisms. If the Coffee Maker's goal is to make coffee, it can do it alone without help from any other agents. In a more intelligent environment, what an agent performs an action, others should benefit from it. When the Coffee Maker's goal is to make coffee, it can do it whenever it wants. However, it is more interesting if Coffee Maker needs someone to consume that coffee. Cooperation among the agents are mandatory for this case. The goals of the agents become more complex and agents cooperate with each other to increase their individual or overall system satisfaction. Therefore, Coffee Maker's goal is changed as to serve coffee to someone in the system. To achieve its goal, Coffee Maker should say "Would you request a cup of coffee from me?" to another agent in the system. These two cases can be formalized as follows:

*User to Coffee Maker:* Commitment from *User Agent* to *Coffee Maker* that is asking for coffee: *CC(CoffeeMaker, User, T, coffee)*. If the *Coffee Maker* also confirms this commitment, it becomes committed to the *User Agent* to serve coffee.

*Coffee Maker to User:* Commitment from *Coffee Maker* to the *User Agent* that is asking for coffee request: *CC(User, CoffeeMaker, T, coffeerequest)*. If the *User Agent* also confirms this commitment, it becomes committed to the *Coffee Maker* for the

coffee request. In fact, coffee request is also represented as a commitment that is the same one with the previous case.

In our representation, the requested commitment is represented as $CC^r$. A REQUEST message can be answered in three ways:

- CONFIRM message indicates that the receiver of the message (i.e., debtor) accepts the request and thus creates the commitment. The commitment becomes active and it is denoted as $CC^a$. The debtor agent becomes committed to satisfy the goal of the creditor agent. Under normal circumstances, its goal will be satisfied. Currently, we do not include exception cases. If an agent becomes committed to satisfy something, it will eventually perform what it is committed to. If User request coffee from the Coffee Maker with sending the commitment *CC(CoffeeMaker, User, T, coffee)* and if the CoffeeMaker confirms the request, it will surely serve coffee to the User Agent.
- INFORM message indicates that the receiver of the message agrees on the request and it performs the request directly without confirming the request with a CONFIRM message before. If this is the case, the commitment state is changed to be fulfilled, denoted as $CC^f$. The choice of sending INFORM without CONFIRM is depends on the agent's internal strategies. The debtor agent immediately executes the proposition and the first agent's goal becomes satisfied. Following the scenario between the User Agent and Coffee Maker, when the Coffee Maker receives the request, it immediately serves the coffee to the User Agent without notifying it about acceptance before.
- PROPOSE indicates that the receiver of the message creates a new conditional commitment requesting another service beneficial for itself in return for the prerequested item. That is, the second agent is willing to perform the requested task but it is asking for something else in return. Thus, the requested commitment becomes conditional and is denoted as $CC^c$. In this case, the first agent should realize the condition of the commitment or find another agent to satisfy its goal. If the condition of the commitment is performed, the debtor agent becomes committed to satisfy the first agent's goal. Suppose Coffee Maker wants to serve

coffee and it sends a coffee request commitment to the User Agent. User Agent receives the request and concludes that it wants cake with coffee. Therefore, it creates a conditional commitment that if the Coffee Maker provides cake to it, User Agent will request coffee. Coffee Maker can accept or reject this conditional commitment proposal.

The answer of a PROPOSE message can be:

- ACCEPT_PROPOSAL which indicates that the creditor of the commitment accepts the proposal. Thus, the commitment is switched to the $CC^a$ state. If the condition is satisfied with an INFORM message, the commitment will stay in $CC^a$ state until it is fulfilled. With accepting the proposal, the agent agrees on the contract and it will realize the condition. If the Coffee Maker decides that it can serve cake to the User Agent, it accepts the proposal and provides cake to the User Agent.
- REJECT_PROPOSAL which indicates that the creditor of the commitment rejects the proposal, denoted as $CC^j$. Later, the debtor agent can request the same commitment to another agent. Coffee Maker has no capability to make cake and also it decides that it cannot find cake from anyone else, therefore it rejects the proposal.

If there is an existing commitment in the system, the debtor of the commitment can CANCEL the commitment ($CC^n$). Similarly, if there is an active commitment, $CC^a$, the debtor and creditor of the commitment can bring about the proposition and the condition by sending an INFORM message. This would fulfill the commitment ($CC^f$).

### 4.3. Algorithms and Rules

It is assumed that at a certain point, an agent $a$ only tries to realize one of its goals $g\prime$ in its unsatisfied goal set $G_a$. An individual goal itself can be a set which we call *a goal branch*. The agent always checks its maintenance goals $M_a$ and does not

behave in conflict with them. We assume that achievement and maintenance goals of an agent do not contradict with each other. The agent can generate commitments as a debtor or creditor and requests to create it from the other party. The agent commits to do something if it is capable or if it believes that it can find another agent to perform it.

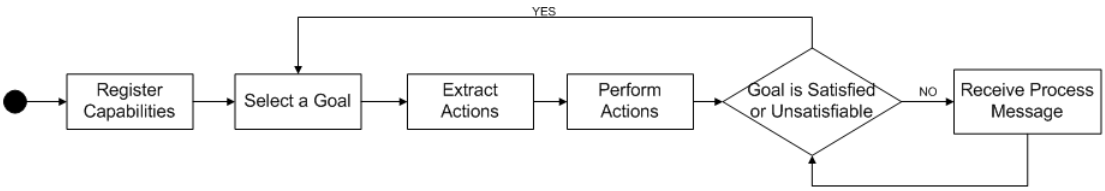


Figure 4.3. Main Lifecycle of the Agent.

The lifecyecle of the agent is shown in Figure 4.3. An agent starts its lifecycle with the *MainBehaviour* algorithm. (Figure 4.4). At the beginning, it registers itself to the publisher agent of the system with providing the services (i.e., capabilities) it can serve to the other agents. It tries to achieve its goals in the order they are adopted. Only one goal can be active at a time, which is called as a current goal of the agent. More specifically, for all of the goals of the agent, it checks its capabilities and creates the relevant requests with calling *ActionstoSatisfy* algorithm (Algorithm 4.6) *(Line 3)*. The current goal of an agent is a list, which can be composed of several subgoals. The agent may be capable for some of the subgoals and it must interact with other agents for the rest.

*ActionstoSatisfy* algorithm (Figure 4.6) takes the goal to be satisfied as an input and returns the action list which should be executed to achieve that goal as an output. The flow of extracting action lists to satisfy the goal is represented in Figure 4.5. The algorithm decomposes the goal to its subparts *(Line 1)* and for each of the subgoal it repeats the following procedure:

- The agent first checks if it is capable to achieve the concerned subgoal (*Line 2 and 3*). If the agent is capable to satisfy it, i.e., if that particular subgoal is within the agent's capability list, it adds this capability to the action list of that goal *(Line 4)*.

**Algorithm 1:** MainBehavior(): Initial Behavior of the agent a

```
1  RegisterDf(C_a)
2  foreach g in G_a do
3      E = ActionstoSatisfy(g);
4      foreach e in E do
5          if e.from == a then
6              PERFORM e;
7          else
8              SEND e;
9      while !g.isSatisfied and g.isSatisfiable do
10         ReceiveAndProcessMessage();
11 ReceiveAndProcessMessage();
```

Figure 4.4. Initial Behavior of the Agent a.

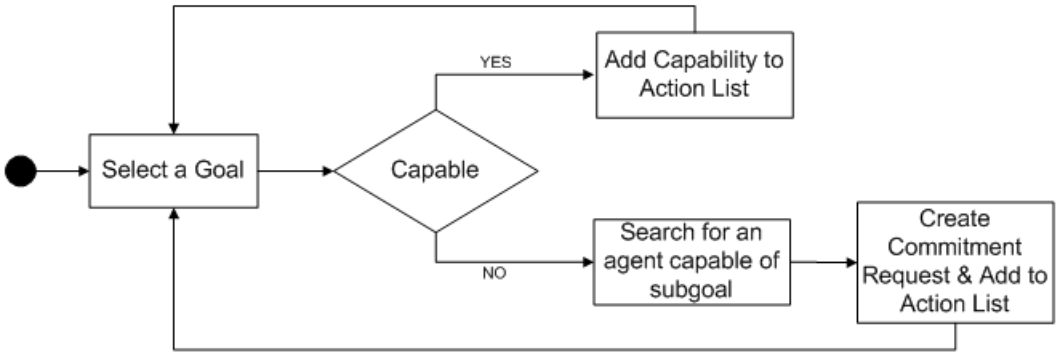


Figure 4.5. Extracting Actions to Satisfy a Goal.

- If it lacks for that capability, it searches for an agent via publisher agent which has the capability to perform the concerned subgoal, initiates a base level commitment containing the goal to that agent, and adds the commitment to the action list to be returned. (*Line 6 to 9).*

At the end, *ActionstoSatisfy* algorithm (Figure 4.6) returns the action list to be executed to the *MainBehaviour* algorithm. *MainBehaviour* algorithm retrieves this list and the agent either sends the commitment request to the debtor agent of the commitment (Line 8) or performs the action itself (Line 6).

**Algorithm 2:** E : ActionsToSatisfy(g): Finding the action list to satisfy g

**Input**: g ∈ $G_a$

**Output**: E is an action set for each part of the goal g

```
1  foreach p in g do
2      foreach e in C_a do
3          if e.content == p then
4              E = E ∪ {e};
5              isCapable=true;
6      if !isCapable then
7          a′: a′ in A and p in C_a′;
8          e = (a,a′,C^r(a′,a,true,p));
9          E = E ∪ {e};
10 return E
```

Figure 4.6. Finding the Action List to Satisfy the Goal g.

In *ReceiveProcessMessage* algorithm, the agent constantly receives and processes messages until it satisfies its current goal or decides that the goal is unsatisfiable. If all of the goals of the agent are examined, the agent proceeds its lifecycle with constantly receiving and processing messages from other agents. Currently, we do not have selfish agents that are only interested in achieving their own goals and when they succeed, they leave the system, and later come back if they have adopted new goals to be satisfied. Instead, the agents continue to be active in the system with listening and responding to the messages from other agents that are trying to satisfy their goals.

*ReceiveProcessMessage* algorithm initiates a relevant response based on the type and content of the message. We have developed rules for the agents to process the received messages. These rules consider the type and content of the message and generates a relevant response. The rules we have developed are as follows:

*When the message contains a commitment request containing a service:*

**Rule 4.1.** $$\frac{CC^r(a,b,T,p), p \in M_a, (\exists c \in C_a : c.content = p)}{CC^j(a,b,T,p)}$$

**Rule 4.2.** $$\frac{CC^r(a,b,T,p), p \notin M_a, (\nexists c \in C_a : c.content = p)}{CC^j(a,b,T,p)}$$

Agent $a$ receives a commitment that is requesting a service $p$ like coffee from agent $b$. The service is the proposition of the commitment. If that service request conflicts with the maintenance goals of the agent (Rule 4.1) i.e., if the agent needs to avoid serving that service, the commitment is rejected. In the same manner, if the agent is not capable to satisfy the service (Rule 4.2), the commitment is also rejected.

Given that, $CC^r(a, b, T, p), p \notin M_a, (\exists c \in C_a : c.content = p)$,

**Rule 4.3.** $$\frac{(\exists com \in T_a : com = CC^r(b, a, T, q))}{CC^c(a, b, q, p), CC^n(b, a, T, q)}$$

**Rule 4.4.** $$\frac{(\nexists com \in T_a : com = CC^r(b, a, T, q))}{CC^f(a, b, T, p), p}$$

If the service $p$ is not conflicting with the agent's goals and if the agent has the capability to perform it, the agent $a$ checks the previous interactions with the creditor agent. Commitment Manager of the agent $a$ provides the list of the commitments in which the agent $a$ requests a service $q$ from the agent $b$, but $b$ has not responded yet. The agent wants to gain benefit from this situation beacuse it desires to achieve its own goal and the agent $b$ that will be served with the requested service can help the agent $a$ to satisfy its goal. If such a case exists, the agent $a$ creates a new conditional commitment proposing that if the agent $b$ satisfies $q$ which the agent $a$ desires to achieve, $a$ will satisfy $p$ (Rule 4.3). Accordingly, agent $a$ forces agent $b$ to perform $q$ in order to achieve $p$. The previous commitment between $a$ and $b$ is canceled. This behaviour demonstrates the fact that agents record all of the interactions they have made and they can use this information of history during their reasoning processes. Agents do not accept all requests based on static rules concerning the goals and capabilities they possess. Conversely, they have dynamic decision making mechanisms regarding to the past and current relations with other agents. This motivation is one of the reasons of why we do not use the static contracts predefined between the agents at the beginning of the system. These static rules are not adequate for such a dynamic system. The commitments exchanged among the agents are dynamic and they cannot be estimated at the beginning. The interaction is continous and the agents can respond to the same commitment in different ways based on their current states and commitments.

Rule 4.4 handles the state that there are not any previous requested commitments from the agent $a$ to the agent $b$. In this case, the agent $b$ does not have to perform anything to $a$, and the agent $a$ is capable for achieving $p$. Thus, the agent $a$ just performs the service $p$ to behave in a friendly and helpful way. Agents are motivated to satisfy their goals, but they are also favorable to help others if certain conditions hold.

*When an agent receives a commitment in which there is another commitment request, it initiates a response based on the following rules:*

**Rule 4.5.** $$\frac{CC^r(a,b,T,CC^r(b,a,T,p)), p \in g_a, p \notin M_a}{CC^f(a,b,T,CC^r(b,a,T,p)), CC^r(b,a,T,p)}$$

**Rule 4.6.** $$\frac{CC^r(a,b,T,CC^r(b,a,T,p)), p \notin g_a, p \notin M_a}{CC^j(a,b,T,CC^r(b,a,T,p))}$$

**Rule 4.7.** $$\frac{CC^r(a,b,T,CC^r(b,a,T,p)), p \in M_a}{CC^j(a,b,T,CC^r(b,a,T,p))}$$

Rules 4.5, 4.6, and 4.7 are activated when the agent receives a commitment request that also contains a commitment request as a proposition. If the agent accepts this request, it is committed to request a commitment to the creditor agent, i.e., sender of the message. This situation shows that the agent's goal does not have to obtain a service, but also it can be a commitment to be fulfilled. As an example, Coffee Maker's goal can be both to make coffee or serve coffee. To serve coffee, it should either receive a commitment request from another agent or it should take an action to make the agents to request coffee from it. Above rules show the latter case.

Rule 4.7 shows the conflict case. If the agent accepts the request, it becomes committed to generate and send a commitment request to the sender of the message. The service it will get when it makes that commitment request can contradict with its maintenance goals. Therefore, the agent $a$ checks if a contradiction will appear before it accepts the request. If it decides that the service conflicts with its maintenance goals, it rejects the request at the beginning. (e.g., User Agent is asked for coffee request by the Coffee Maker but coffee is forbidden to the User Agent because of its health issues,

then coffee is in User Agent's maintenance goal list that means drinking coffee needs to be prevented. User Agent rejects the request of the Coffee Maker.)

Rule 4.5 shows that if there are no contradictions between the service it will get and the maintenance goals of the agent, and if the service it will get when it generates that commitment is in its current goal branch (e.g., User Agent's goal is to drink tea, then the commitment request which says that User Agent should ask for tea from Coffee Maker is beneficial for the User, but another commitment which requests coffee from Coffee Maker is not.), then it accepts the initial request and then generates and sends the commitment to the other party.

On the other hand, Rule 4.6 shows the case that the service it will get when it requests the commitment is not profitable for the agent's current goal (e.g., agent's goal is drinking tea but the request is related to coffee). The result is that the commitment request is rejected. The agent's main motivation for this behaviour is to satisfy its current goal. Therefore, it does not participate in contracts which are not beneficial to its current goal satisfaction process. It just focuses on its current goal and it acts in a way that these actions will get it closer to the achievement of its current goal.

*When an agent receives a conditional commitment proposal, it initiates a response based on the following rules:*

Given that, $CC^c(b,a,p,q), p \notin M_a, (com \in T_a : com = CC^r(b,a,T,q))$,

**Rule 4.8.** $$\frac{(\exists c \in C_a : c.content = p)}{p, CC^a(b,a,p,q), CC^n(b,a,q)}$$

**Rule 4.9.** $$\frac{(\nexists c \in C_a : c.content = p), (\exists com' \in T_a, d \in A : com' = CC^r(d,a,T,q))}{CC^n(b,a,p,q), CC^n(b,a,T,q)}$$

**Rule 4.10.** $$\frac{(\nexists c \in C_a : c.content = p), (\nexists com' \in T_a, d \in A : com' = CC^r(d,a,q))}{CC^n(b,a,q), CC^r(x,a,p), CC^a(b,a,p,q)}$$

An agent $a$ can receive a conditional commitment request as an answer to its previous requests or an unrelated conditional commitment request from its previous interactions.

For the former case, agent $a$ requested a base level commitment from $b$ and $b$ proposes a conditional commitment in return for the $a$'s base level commitment using Rule 4.3. $a$ is the creditor and $b$ is the debtor of this commitment. If $a$ satisfies $p$, which is the condition of the commitment, $b$ will become committed to achieve the proposition $q$. Rules 4.8, 4.9 and 4.10 illustrate this case:

- $a$ wants to achieve $q$ so it is willing to accept the commitment and perform what it is responsible to do for the fulfillment of the commitment which also leads to the satisfaction of its goal. But, $a$ must be capable to satisfy the condition. Therefore, $a$ first ensures that if it is capable to perform $p$. If it is capable, it immediately executes the condition and cancels the previous base level commitment that it requests $q$ from $b$ (Rule 4.8). That commitment becomes invalid since it is cancelled by the agent $a$.
- If the agent $a$ is not capable for serving $p$, it ensures if there are other commitments for $q$ in its repository. If there exists a requested commitment for $q$ from another agent $d$ rather than $b$, it rejects the proposal and hopes for the answer from the agent $d$ (Rule 4.9). The motivation for that behavior is that it can satisfy its goal without performing anything in return for it.
- Rule 4.10 represents the case that $q$ is only requested from $b$, and the agent $a$ is not capable to perform condition $p$. $a$ desires $q$ to be executed, but the only way to achieve it, at current circumstances, is to perform the condition of the proposed commitment. So, the agent $a$ adds the condition $p$ to a separate goal set and it participates in contracts with other agents who have the capability to satisfy it.

Given that, $CC^c(b, a, p, q), p \notin M_a, (\nexists com \in T_a : com = CC^r(b, a, T, q))$

**Rule 4.11.** $$\frac{(\exists c \in C_a : c.content = p), q \in g_a}{CC^a(b, a, p, q), p}$$

**Rule 4.12.** $$\frac{(\nexists c \in C_a : c.content = p), q \in g_a}{CC^j(b, a, p, q)}$$

**Rule 4.13.** $$\frac{(\exists c \in C_a : c.content = p), q \notin g_a}{CC^f(b, a, p, q), p}$$

Rules 4.11, 4.12 and 4.13 handle the situations in which the proposed conditional commitment is not related to any other previous commitments. The debtor agent $b$, based on its own reasoning process and strategies, requests a conditional commitment from scratch. In such a case, the agent $a$ first checks if the condition of the commitment that the agent $a$ must perform if it accepts the request is conflicting with the maintenance goal of it. If such a conflict exists, it immediately rejects the request. Otherwise, it chekcs if it is capable for $p$. If it does not have any capability to perform $p$, it rejects the request. If it has the relevant capability, it then ensures that if it wants to achieve $q$. If $q$ is in its current goal branch, it immediately accepts the commitment request, performs the condition $p$, and waits for the $q$ to be executed which is beneficial for its current goal achievement. If the agent $a$ has the capability for the condition but it does not have desire to have $q$, i.e., it is neutral for the service $q$, it accepts the request to favor other agents.

# 5. APPLICATION AND SCENARIOS

We have developed an application to realize our algorithms and rules as explained in Section 4.3. Our application domain is an intelligent kitchen. Our focus does not cover the sensor and actuator of devices. We assume that they have necessary components to sense the environment and interact with it. The low level communication of the devices is handled with the agent development framework we have used as will be described in Section 5.8. The context of the interaction between the agents is commitment-based regulations. Interaction among the agents is carried out with commitments and these commitments are regulated by the rules given in Section 4.3. The interactors of the environment are both the devices of the kitchen and the users of this domain. Therefore, our application can be expressed based on the formalization used in Section 2.1 [2]:

$$AmI : \langle E, IC, I \rangle, \text{ where}$$

- *E (environment):* kitchen,
- *IC (interaction constraints):* $\langle S, A, C, IR \rangle$ such that *S (sensors)* and *A (actuators)* are not handled, *C (interaction constraints)* are represented as commitments, and *IR (interaction rules)* are developed as given in Section 4.3,
- *I (interactors):* devices and users of the kitchen.

Consider the following scenarios that take place among the interactors (i.e., agents that represent the User, Coffee Maker and Refrigerator) of the system in the kitchen based on the interaction rules defined.

## 5.1. Scenario 1: Coffee Maker and Refrigerator are Capable to Satisfy Their Goals

The goals and capabilities of these agents for this scenario are given in Table 5.1. Since the destination agent field is not filled, Coffee Maker can serve coffee to all agents

in the system. Its goal is to make coffee. Refrigerator's capability is to serve cake to all agents in the system. Its goal is to make cake.

Table 5.1. Services and Goals of the Agents for Scenario 5.1.

| Agents | Services | Achievement Goals | Maintenance Goals |
|---|---|---|---|
| CM | (CM,-,coffee) | coffee | none |
| F | (F,-,cake) | cake | none |

In this scenario, both the Coffee Maker and Refrigerator are capable to satisfy their goals. They do not need to interact for cooperation with other agents within the system. Coffee Maker simply makes coffee with its available resources and in the same way Refrigerator obtains cake from its storage or orders from patisserie. This scenario demonstrates the environment that the goals of the agents are not interesting and meaningful in terms of intelligent domains. Making coffee for a Coffee Maker is not a profitable goal for the application, it is just useful for the Coffee Maker. If all agents in the system behave in the same manner, there is no need to have complex algorithms or rules for the achievement of agents' goals. Simple rules can work for these agents. Capabilities and goals of the agents need not be distinguished. However, someone in the system should also benefit from the action performed rather than the agent itself if we desire to have realistic applications. For instance, it is not meaningful for a chef to adopt a goal that is to cook. She desires to serve what she has cooked to the people. So, there must be at least one agent in the environment which needs the served coffee.

Presence of a User Agent which is willing to drink coffee is inadequate. User Agent that desires to drink coffee or eat cake will not be satisfied if the goals of the Coffee Maker and Refrigerator agents remain same. Our agents' motivation to take actions is their goals so goals of the agents should be changed in an appropriate way. Thus, Coffee Maker should also desire to serve coffee and Refrigerator should want to serve cake. Scenario 5.2 realizes this issue.

*Execution:* Since the capabilities of the agents are sufficient to satisfy their goals, they do not generate any commitments and they do not interact with each other.

## 5.2. Scenario 2: User Agent and Coffee Maker Cooperate to Serve Coffee to the User Agent

In this scenario, User Agent's goal is to have cake and coffee. It does not have any capability. So it needs to participate in contracts with other agents. Coffee Maker's capability is to serve coffee. Its goal is different from the previous case. It desires to serve coffee to any of the agents in the system. Formally, Coffee Maker's achievement goal is the fulfillment of a commitment in which Coffee Maker is committed to serve coffee to an agent.

Table 5.2. Services and Goals of the Agents for Scenario 5.2.

| Agents | Services | Achievement Goals | Maintenance Goals |
|---|---|---|---|
| UA | none | coffee, cake | none |
| CM | (CM,-,coffee) | $CC^f$(CM,-,T,coffee) | none |
| F | (F,-,cake) | $CC^f$(F,-,T,cake) | none |

Refrigerator's capability is to serve cake. It also should find an agent that is willing to have cake. In a same manner, Refrigerator's achievement goal is the fulfillment of a commitment in which another agent requests cake from it.

*Execution:* In this scenario, User Agent initiates the execution. Since it has two goals to be satisfied, it selects one of them randomly. Suppose it is to have coffee. User Agent tries to find an agent which is both able and willing to serve coffee for it. If it can find immediately, it requests coffee within a commitment and waits for the answer. If it could not find any agent to request coffee, it subscribes to the publisher agent for coffee, and it makes its coffee request when an agent which is capable to serve coffee registers to the system.

Table 5.3. Commitments Generated During the Execution of Scenario 5.2.

| | |
|---|---|
| **1A** | $CC^r$(CM,UA,T,coffee) |
| **1B** | $CC^f$(CM,UA,T,coffee) |

Table 5.3 shows the commitments generated during the execution of the scenario and Figure 5.2 shows the transition of those commitments.


Figure 5.1. Evolution of Commitments Generated During the Execution of Scenario 5.3.

Through the publisher agent within the system, User Agent finds out that the Coffee Maker agent has capability to brew coffee. It creates a base level commitment *1A* and sends the commitment to the Coffee Maker agent. Coffee Maker receives the message and concludes that serving coffee to the User Agent will satisfy one of its goals: *1B*. Therefore it is beneficial for Coffee Maker to accept the commitment. As a result, Coffee Maker serves the requested coffee. User Agent gets coffee and updates its goal set. Base level commitment state is changed to be fulfilled $(CC^f (CM,U,T,coffee))$.

If only static commitments exist that are created at the beginning of the system, agents can only participate in these predefined commitments. In that case, if User Agent does not have any predefined commitments including coffee, it cannot achieve its goal even an agent i.e., Coffee Maker, which is capable and willing to serve coffee exists in the system.

## 5.3. Scenario 3: User Agent Proposes Coffee Maker to Serve Cake in return for Coffee Request

Table 5.4 shows the agents and their characteristics for this scenario. User Agent's goal is to have cake and coffee. It does not have any capability. So it needs to be involved in contracts with other agents. Coffee Maker's capability is to serve coffee. Coffee Maker's achievement goal is the fulfillment of a commitment in which Coffee Maker is committed to serve coffee to another agent. Refrigerator's capability is to

serve cake. It also should find an agent that is willing to have cake. In a same manner, Refrigerator's achievement goal is the fulfillment of a commitment in which another agent requests cake from it.

Table 5.4. Services and Goals of the Agents for Scenario 5.3.

| Agents | Services | Achievement Goals | Maintenance Goals |
|---|---|---|---|
| UA | none | coffee, cake | none |
| CM | (CM,-,coffee) | $CC^f$(CM,-,T,coffee) | none |
| F | (F,-,cake) | $CC^f$(F,-,T,cake) | none |

*Execution:* Table 5.5 shows the commitments generated during the execution of the scenario and Figure 5.3 shows the transition of those commitments.

Table 5.5. Commitments Generated During the Execution of Scenario 5.3.

| | |
|---|---|
| **1A** | $CC^r$(UA,CM,T,$CC^r$(CM,U,T,coffee)) |
| **1B** | $CC^n$(UA,CM,T,$CC^r$(CM,U,T,coffee)) |
| **2A** | $CC^c$(UA,CM,cake,$CC^r$(CM,U,T,coffee)) |
| **2B** | $CC^a$(UA,CM,cake,$CC^r$(CM,U,T,coffee)) |
| **2C** | $CC^f$(UA,CM,cake,$CC^r$(CM,U,T,coffee)) |
| **3A** | $CC^r$(F,CM,T,cake) |
| **3B** | $CC^f$(F,CM,T,cake) |
| **4A** | $CC^r$(CM,UA,T,coffee) |
| **4B** | $CC^f$(CM,UA,T,coffee) |

Coffee Maker's goal is to serve cake: *$CC^f$(CM,-,true,coffee)*. It should find an agent to request a commitment to gather coffee request. Through the publisher agent, it chooses User Agent to cooperate for its goal. It creates a base level commitment request (1A) which is requesting a coffee request and sends it. User receives the message. It has already satisfied its coffee goal. Its current goal is to have cake. Therefore it concludes to propose a new commitment to the Coffee Maker. It can drink coffee with cake so it requests cake in return for coffee (2A). After the base level commitment (1A) is changed to a conditional commitment (2A), state of 1A is changed to be canceled

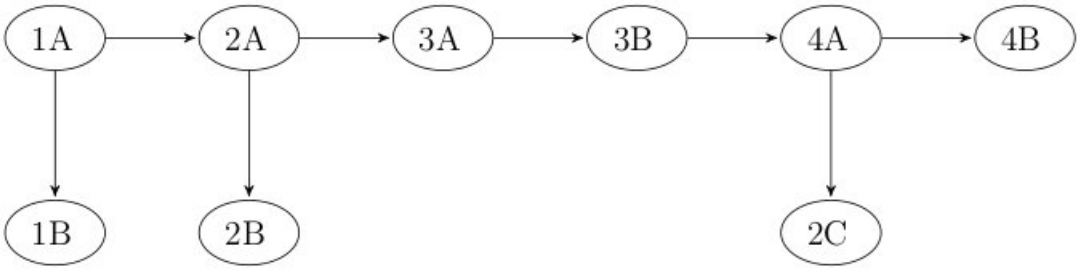


Figure 5.2. Evolution of Commitments Generated During the Execution of Scenario 5.3.

(1B). Coffee Maker is not capable for serving cake, but it will achieve its goal (serving coffee) if it finds an agent for the User Agent which is willing to serve cake. Thus, it accepts the proposal (2B), and tries to find an agent that will serve cake for the User Agent. It searches cake service via publisher agent and concludes that Refrigerator has the capability for serving cake, and then it requests cake from Refrigerator(3A). Refrigerator receives the request, it is willing to serve cake so it accepts the request and serves cake (3B). Since the User Agent obtains cake and satisfies its goal, it becomes committed to Coffee Maker for the coffee request. Therefore, it makes the coffee request (4A) and Coffee Maker serves coffee. Both the proposed commitment (2C) and the coffee request commitment (4B) become fulfilled which leads to the satisfaction of both Coffee Maker's and User Agent's goals.

## 5.4. Scenario 4: User Agent's Request Contradicts with Coffee Maker's Maintenance Goal

User Agent's goal is to have cake and coffee. It does not have any capability. So it needs to involve in contracts with other agents. Coffee Maker's capability is to make coffee and tea. Coffee Maker's achievement goal is to serve coffee and more formally the fulfillment of a commitment in which Coffee Maker is committed to serve coffee to another agent. Despite that the Coffee Maker can serve tea, its maintenance goal is not to serve tea. There can be several reasons for that behavior such as to consume the existing coffee first instead of making tea. Refrigerator's capability is to serve cake. It

also should find an agent that is willing to have cake. In the same manner, formally, Refrigerator's achievement goal is the fulfillment of a commitment in which another agent requests cake from it. It does not have any maintenance goal.

Table 5.6. Services and Goals of the Agents for Scenario 5.4.

| Agents | Services | Achievement Goals | Maintenance Goals |
|---|---|---|---|
| UA | none | coffee, cake | none |
| CM | (CM,-,coffee), (CM,-,tea) | $CC^f$(CM,-,T,coffee) | tea |
| F | (F,-,cake) | $CC^f$(F,-,T,cake) | none |

*Execution:*

Table 5.7. Commitments Generated During the Execution of Scenario 5.4.

| | |
|---|---|
| **1A** | CC$^r$(CM,UA,T,tea |
| **1B** | CC$^j$(CM,UA,T,tea) |

Table 5.7 shows the commitments generated during the execution of the scenario and Figure 5.4 shows the transition of those commitments.


Figure 5.3. Evolution of Commitments Generated During the Execution of Scenario 5.4.

Suppose User Agent has achieved its cake and coffee goals. Currently, it is trying to satisfy its last goal that is to drink tea. It searches for the agent which is able to serve tea and it finds out that Coffee Maker can serve it. It creates a commitment request for tea (1A) and sends it. Coffee Maker receives the message and concludes that serving tea conflicts with its maintenance goal. Therefore it rejects the request of the User. The commitment state is updated (1B) as rejected.

## 5.5. Scenario 5: Coffee Maker Leaves the System During the Run Time

Initially there are three agents in the system: User Agent, Coffee Maker and Refrigerator. User Agent does not have any capability and its goal is to have coffee and cake. Coffee Maker can make coffee to all of the agents. Its goal is to serve coffee which is expressed as the fulfillment of a commitment in which Coffee Maker is committed to serve coffee to another agent. Refrigerator is capable to serve cake. Its goal is to serve cake to other agents in the system. None of the agents has a maintenance goal.

*Execution:*

Table 5.8. Commitments Generated During the Execution of Scenario 5.5.

| | |
|---|---|
| **1A** | $CC^r$(CM,UA,T,coffee |
| **1B** | $CC^n$(CM,UA,T,coffee) |
| **2A** | $CC^n$(F,UA,T,cake) |
| **2B** | $CC^n$(F,UA,T,cake) |

Table 5.8 shows the commitments generated during the execution of the scenario and Figure 5.5 shows the transition of those commitments.

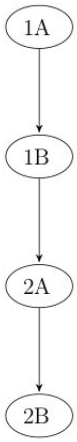


Figure 5.4. Evolution of Commitments Generated During the Execution of Scenario 5.5.

Suppose User Agent initiates the execution and tries to achieve its one of the

unsatisfied goals that is to have coffee. It searches for the agent which is able to serve coffee and it finds out that Coffee Maker can serve it. During the User Agent's reasoning, Coffee Maker decides to leave the system since it achieves its goals and it needs to rest for some time or it is unavailable due to power shortage. When it leaves, the services it provides are automatically deregistered but User Agent is not aware of that. It creates a base level commitment request for coffee (1A) and sends it. But its message cannot be delivered to the Coffee Maker. User Agent realizes the situation, it cancels its commitment (1B), and searches for another agent who can make coffee. However, there does not exist such an agent. User Agent decides that its coffee goal is unsatisfiable with current circumstances, and it changes its current goal as to have cake. It finds Refrigerator agent who can serve cake and requests cake from it (2A). Refrigerator receives the request and since its goal is to serve cake, it accepts and serves cake to the User Agent (2B). Both the User Agent's and Refrigerator's achievement goals are satisfied.

## 5.6. Scenario 6: Coffee Maker Changes its Capability During the Run Time

Initially there are three agents in the system: User Agent, Coffee Maker and Refrigerator. User Agent does not have any capability and its goal is to drink tea. Coffee Maker can make coffee and tea to all of the agents. Its goal is to serve coffee which is expressed as the fulfillment of a commitment in which Coffee Maker is committed to serve coffee to another agent. Refrigerator is capable to make cake. Its goal is to serve cake to other agents in the system. None of the agents have a maintenance goal.

Table 5.9. Services and Goals of the Agents for Scenario 5.6.

| Agents | Services | Achievement Goals | Maintenance Goals |
|---|---|---|---|
| UA | none | tea | none |
| CM | (CM,-,tea), (CM,-,coffee) | $CC^f$(CM,-,T,coffee) | none |
| F | (F,-,cake) | $CC^f$(F,-,T,cake) | none |

*Execution:*

Table 5.10. Commitments Generated During the Execution of Scenario 5.6.

| | |
|---|---|
| **1A** | $CC^r$(CM,UA,T,tea |
| **1B** | $CC^j$(CM,UA,T,tea) |

Table 5.10 shows the commitments generated during the execution of the scenario and Figure 5.6 shows the transition of those commitments.


Figure 5.5. Evolution of Commitments Generated During the Execution of Scenario 5.6.

Suppose User Agent initiates the execution and tries to achieve its goal that is to drink tea. It searches for the agent which is able to serve tea and it finds out that Coffee Maker can serve it. During the User Agent's reasoning, Coffee Maker removes its tea capability since it decides to consume existing coffee it contains. It deregisters tea service from publisher agent but User Agent is not aware of that. It creates a base level commitment request for tea (1A) and sends it. When the Coffee Maker receives that commitment request, it rejects since it does not want to serve tea (1B). User Agent searches for another agent who can serve tea but it cannot find one. Since it does not have any other goal to be achieved it starts to wait for an agent who can serve tea to it.

## 5.7. Scenario 7: Coffee Maker has a Capability with Specified Destination Agent

There are four agents in the system: User $Agent_1$, User $Agent_2$, Coffee Maker and Refrigerator. User $Agent_1$ and User $Agent_2$ do not have any capabilities and they share the same goal that is to drink tea. Coffee Maker can serve tea to only User $Agent_1$.

Its goal is to serve tea which is expressed as the fulfillment of a commitment in which Coffee Maker is committed to serve tea to another agent. Refrigerator is capable to make cake. Its goal is to serve cake to other agents in the system. None of the agents have a maintenance goal.

Table 5.11. Services and Goals of the Agents for Scenario 5.7.

| Agents | Services | Achievement Goals | Maintenance Goals |
|---|---|---|---|
| $UA_1$ | none | tea | none |
| $UA_1$ | none | tea | none |
| CM | (CM,$UA_1$,tea) | $CC^f$(CM,-,T,tea) | none |
| F | (F,-,cake) | $CC^f$(F,-,T,cake) | none |

*Execution:*

Table 5.12. Commitments Generated During the Execution of Scenario 5.7.

| | |
|---|---|
| **1A** | $CC^r$(CM,$UA_2$,T,tea |
| **1B** | $CC^j$(CM,$UA_2$,T,tea) |

Table 5.12 shows the commitments generated during the execution of the scenario and Figure 5.7 shows the transition of those commitments.


Figure 5.6. Evolution of Commitments Generated During the Execution of Scenario 5.7.

Suppose User $Agent_2$ initiates the execution and tries to achieve its goal that is to drink tea. It searches for the agent which is able to serve tea and it finds out that Coffee Maker can serve it. It creates a base level commitment request for tea (1A) and sends it. When the Coffee Maker receives that commitment request, it rejects since it

can serve tea to only one of the agents: User $\text{Agent}_1$ (1B). User $\text{Agent}_2$ searches for another agent who can serve tea but it could not find. Since it does not have any other goal to be achieved it starts to wait for an agent who can serve tea to it.

## 5.8. Implementation

### 5.8.1. JADE

The Foundation for Intelligent Physical Agents (FIPA) [41] is a nonprofit organization that aims to specify standards for agent technologies in different application domains. Figure 5.7 shows the agent framework of FIPA. In FIPA architecture, there are some specific regulative agents for the operation and management of the system. It also defines the agent management content language and ontology. There are three key agents: *(1) The Agent Management System (AMS)* is responsible for authenticating and registering the agents. *(2) The Agent Communication Channel (ACC)* provides the way for communication between the inside and outside agents. *(3) The Directory Facilitator (DF)* agent obtains a yellow pages service to the platform.

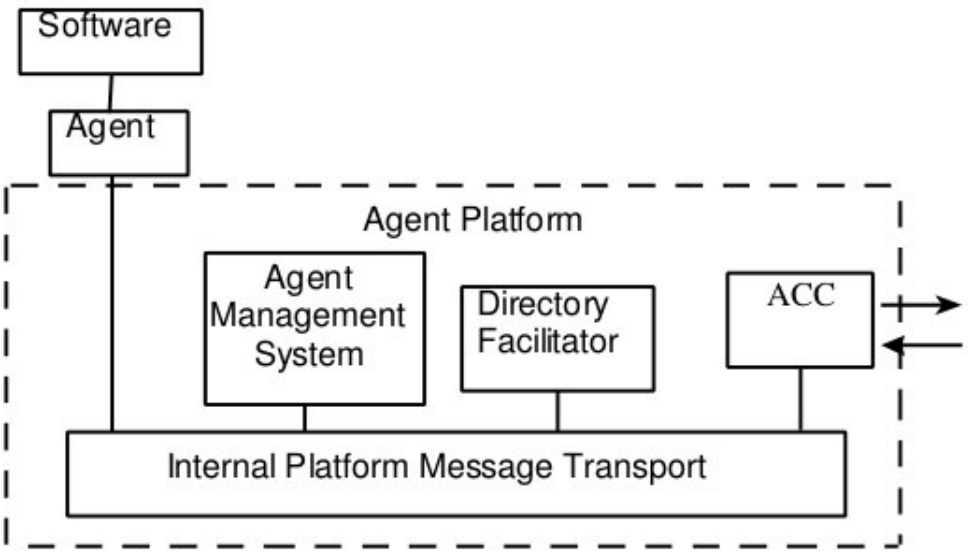


Figure 5.7. FIPA Agent Framework.

*Agent Communication Language (ACL)* is defined as the standard for messaging among agents. ACL is based on speech-act theory; messages are actions that the

actions are specified by the types of the messages provided.

JADE (Java Agent DEvelopment Framework) is a software framework which provides a framework for the development of FIPA-compliant agents. Main features of JADE are:

- FIPA-compliancy: AMS, ACC and DF agents are present and activated at the start of the execution. Agents communicate with ACL messages.
- Distributed: the platform can be split on different hosts.
- multiple DS: for implementing multi-domain applications, several DSs can be started at run time.
- Graphical User Interface: It makes the registration, termination of agents and monitoring of agent activities easy.
- automatic registry of agents to AMS agent.

### 5.8.2. System Details

We have developed our system in Java. The application platform we have used is Netbeans IDE. We have developed our agents with JADE [39]. The reasons for it are:

- JADE is a successful agent development platform that have been used by many people.
- It is a FIPA-compliant framework which also makes our application compatible with other FIPA based applications.
- It handles the messaging architecture between agents. It uses the ACL messages which is beneficial for our approach. Our agents send a commitment as a content of the message, and the type of the ACL message fits well with our commitment state transition.
- JADE provides a graphical user interface which creates and terminates the agents in a friendly way.
- JADE allows interaction of the agents in different hosts which supports our distributed architecture.

JADE is composed of containers that are the instances of JADE running environment. There is one main container that must be always active and other created containers register to that container. The Main Container also holds AMS and DF agents.

Each agent is a single-threaded Java program. Agent's actions are regulated by its set of behaviours. Behaviours are Java objects and they execute until they are finished. A behavior represents a task that the agent must achieve. The agents process their received messages by triggering the relevant behaviour based on the type and content of the message.

ACL messages mainly contain those fields:

- type of the message: request, inform, propose, accept-proposal, reject-proposal, cancel, and so on.
- sender: the id of the sendor
- receiver: the id of the receiver
- content : in our approach, the content is the commitment.

Our agent classes extend the JADE agents. Agents communicate with ACL messages and they respond to the events with JADE behaviors.

We select a smart kitchen domain to execute and test our algorithms. There are several agents representing the devices of the kitchen and users. The environment is dynamic. New devices can enter to the kitchen. Existing ones can leave or can be broken for some reason. Therefore agents decide to create contracts during their execution depending on their goals and capabilities states.

# 6. CONCLUSION

We have proposed and realized a distributed multiagent system for the representation of autonomous and dynamic environments. Each agent has a set of capabilities and goals. They aim to satisfy their goals; they can do by themselves if they have all capabilities for the goals. Otherwise, the agents need to interact with other agents. In our system, agents decide with whom to communicate and the content of the interaction dynamically depending on their current reasoning. The content of interaction is formalized with using the commitment protocols that are some kind of obligations between two agents. They make contracts with each other and cooperate for satisfying their individual goals.

Agent based approaches for ambient intelligence applications have been applied before, especially for infomobility services (the Im@gine IT project) [42]. Examples for those services include location-based services like travel planning, trip progression monitoring and pushing information and events to the user. Additionally, there has been recent research that focuses on the elderly and disabled people. Those types of applications include both the devices related to the AmI environment such as sensors within the home and mobile services run on the PDA, cell phone etc. Thus, both the agent architecture and service oriented approach must be combined. In a travel assistance application [43], FIPA agents are developed and they work on OSGi framework together with the services. They provide a way to combine two technologies but they do not specify the content of interaction between the services and the agents. The problem is that the intelligence that should be integrated to the agents is missing.

LAICA [44] project develops agent based ambient intelligence applications. They argue that agents can be used to represent devices and users and the multiagent system approach can be adopted to implement the AmI environment. They are also interested in the mobility and modularity that the agents can provide. The project focuses on the use of traffic lights, video cameras and motion services. All of the data streams produced by the devices are stored in a centralized database. All of the distributed

sensors of LACIA can host the execution of data processing algorithms. The data sensed by the devices are pre-processed to extract relevant information. AmI agents can be of two types: sensor agents attached to devices and effector agents that uses the information. The system is developed with Java but they do not prefer to use existing frameworks. They argue that the existing frameworks are too complex for their primitive interactions. Also, they do not prefer to use agent development frameworks like Jade to handle the communication among agents. One reason is that their messages are too simple and they simply transfer sensor data streams. From the data they capture, they aim to identify the normal conditions and the warnings. They provide two use cases: Tracking a stolen car and identifying odd human behaviours. In a similar manner, this projects lacks the intelligence behind the agents and the environment. Sensors observe the environments, system identify some patterns for activities with the data, and the recognition of the unusual activities are recognized. The interaction among the agents are just transfering raw data.

Apart from the AmI applications, commitment based approach for agent communication has been previously used [14] as mentioned in Chapter 1. The problem here is that the commitments are predefined from the beginning of the system. Agents select commitments to create or request from another agent from the commitment specification given to the agents before run time.

In this thesis, we have introduced some algorithms and goals to regulate the interactions and cooperations of agents. Also, we have developed an application to realize these algorithms and rules. Results show that our approach works well among the standard agents that fully adopt them. It will be more interesting to observe the situation if we have different behaving agents in our system. Moreover, our proposed system can be applied to different domains to extract the properties of the environments it best fits. We can include exception cases during the execution of the agents. Even if an agent is committed to satisfy a proposition, because of several reasons, it may not realize it. Exception cases are not handled and it can be a interesting future work. Another important aspect is that, in a cooperative environment, reputation mechanism is crucial for agents. The agents store their history of commitments to make more

reasonable deductions. This history can also be used to evaluate the agents it interacts with. It can start to trust some of the agents that it had positive interactions with, and vice versa. Using reputations during the decision processes will make the system more intelligent and interesting to observe. On the other hand, our agents only adopt maintenance and achievement goals. Perform and query goals can be added to improve the goal scope of the agents.

# REFERENCES


1. Aarts, E. and J. Encarnao, *True Visions: The Emergence of Ambient Intelligence*, Springer Publishing Company, Incorporated, 2008.

2. Augusto, J. C., "Past, Present and Future of Ambient Intelligence and Smart Environments", J. Filipe, A. L. N. Fred and B. Sharp (Editors), *Proceedings of the International Conference on Agents and Artificial Intelligence*, pp. 11–18, 2009.

3. Ramos, C., J. C. Augusto and D. Shapiro, "Ambient Intelligence - The Next Step for Artificial Intelligence", *IEEE Intelligent Systems*, Vol. 23, pp. 15–18, March 2008.

4. Dix, A., J. E. Finlay, G. D. Abowd and R. Beale, *Human-Computer Interaction (3rd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

5. Denning, P. J. (Editor), *The Invisible Future: the Seamless Integration of Technology into Everyday Life*, McGraw-Hill, Inc., New York, NY, USA, 2002.

6. Huhns, M. N. and M. P. Singh, "Agents and Multiagent Systems: Themes, Approaches, and Challenges", M. N. Huhns and M. P. Singh (Editors), *Readings in Agents*, chap. 1, pp. 1–23, Morgan Kaufmann, San Francisco, 1997.

7. Singh, M. P. and A. K. Chopra, "Correctness Properties for Multiagent Systems", *Lecture Notes in Computer Science*, pp. 192–207, Springer, 2009.

8. Braubach, L., A. Pokahr, W. Lamersdorf and D. Moldt, "Goal Representation for BDI Agent Systems", R. H. Bordini, M. Dastani, J. Dix and A. E. Fallah-Seghrouchni (Editors), *Second International Workshop on Programming Multiagent Systems: Languages and Tools*, pp. 9–20, Springer, 2004.

9. Baldoni, M., C. Baroglio and E. Marengo, "Behavior-oriented Commitment-based

Protocols", *Proceedings of 19th European Conference on Artificial Intelligence*, pp. 137–142, IOS Press, 2010.

10. Yolum, P. and M. P. Singh, "Flexible Protocol Specification and Execution: Applying Event Calculus Planning Using Commitments", *Proceedings of 1st International Conference on Autonomous Agents and Multiagent Systems*, pp. 527–534, ACM, 2002.

11. Alberti, M., F. Chesani, M. Gavanelli, E. Lamma, P. Mello and P. Torroni, "Verifiable Agent Interaction in Abductive Logic Programming: The SCIFF framework", *ACM Transactions on Computational Logic*, Vol. 9, No. 4, pp. 1–43, 2008.

12. Yolum, P. and M. P. Singh, "Enacting Protocols by Commitment Concession", *Proceedings of 6th International Conference on Autonomous Agents and Multiagent Systems*, p. 27, 2007.

13. Singh, M. P., "An ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts", *Artificial Intelligence and Law*, Vol. 7, pp. 97–113, 1999.

14. Fornara, N. and M. Colombetti, "A Commitment-based Approach to Agent Communication", *Applied Artificial Intelligence*, Vol. 18, pp. 853–866, 2004.

15. Youngblood, G. M. and D. J. Cook, "Data Mining for Hierarchical Model Creation", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 37, No. 4, pp. 561–572, 2007.

16. Youngblood, G. M., D. J. Cook and L. B. Holder, "Managing Adaptive Versatile Environments", *Pervasive and Mobile Computing*, Vol. 1, pp. 373–403, December 2005.

17. Doctor, F., H. Hagras and V. Callaghan, "A Fuzzy Embedded Agent-based Approach for Realizing Ambient Intelligence in Intelligent Inhabited Environments",

*IEEE Transactions on Systems, Man, and Cybernetics, Part A*, Vol. 35, No. 1, pp. 55–65, 2005.

18. Doctor, F., H. Hagras and V. Callaghan, "A Type-2 Fuzzy Embedded Agent to Realise Ambient Intelligence in Ubiquitous Computing Environments", *Information Sciences—Informatics and Computer Science: An International Journal*, Vol. 171, pp. 309–334, May 2005.

19. Barger, T., D. Brown and M. Alwan, "Health Status Monitoring Through Analysis of Behavioral Patterns", *8th congress of the Italian Association for Artificial Intelligence on Ambient Intelligence*, pp. 22–27, Springer-Verlag, 2003.

20. Thomson, G., D. Sacchetti, Y.-D. Bromberg, J. Parra, N. Georgantas and V. Issarny, "Amigo Interoperability Framework: Dynamically Integrating Heterogeneous Devices and Services", *Constructing Ambient Intelligence*, pp. 421–425, 2008.

21. Matos, M. and A. Sousa, "Dependable Distributed OSGi Environment", *Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion*, Companion '08, pp. 104–106, ACM, New York, NY, USA, 2008.

22. Aztiria, A., J. Augusto, R. Basagoiti and A. Izaguirre, "Accurate Temporal Relationships in Sequences of User Behaviours in Intelligent Environments", J. Augusto, J. Corchado, P. Novais and C. Analide (Editors), *Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence*, Vol. 72 of *Advances in Intelligent and Soft Computing*, pp. 19–27, Springer Berlin / Heidelberg, 2010.

23. Dey, A. K., "Understanding and Using Context", *Personal Ubiquitous Computing*, Vol. 5, pp. 4–7, 2001.

24. van Riemsdijk, M. B., M. Dastani and M. Winikoff, "Goals in Agent Systems: a Unifying Framework", *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 713–720, International Founda-

tion for Autonomous Agents and Multiagent Systems, Richland, SC, 2008.

25. van Riemsdijk, M. B., M. Dastani and M. Winikoff, "Goals in Agent Systems: a Unifying Framework", *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, pp. 713–720, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008.

26. Pokahr, A., L. Braubach and W. Lamersdorf, "Jadex: Implementing a BDI-infrastructure for JADE Agents", *EXP - in Search of Innovation (Special Issue on JADE)*, Vol. 3, No. 3, pp. 76–85, 9 2003.

27. Huber, M. J., "JAM: A BDI-theoretic Mobile Agent Architecture", *In Proceedings of the Third International Conference on Autonomous Agents*, pp. 236–243, ACM Press, 1999.

28. Rao, A. S., "AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language", R. van Hoe (Editor), *Seventh European Workshop on Modelling Autonomous Agents in a Multiagent World*, Eindhoven, The Netherlands, 1996.

29. Howden, N., R. Rönnquist, A. Hodgson and A. Lucas, "JACK Intelligent Agents - Summary of an Agent Infrastructure", *Proceedings of the 5th International Conference on Autonomous Agents*, 2001.

30. Dastani, M., M. B. van Riemsdijk, F. Dignum, M. Birna, R. F. Dignum and J. jules Ch. Meyer, "A Programming Language for Cognitive Agents Goal Directed 3APL", pp. 111–130, Springer, 2003.

31. Cook, D. J., "Multiagent Smart Environments", *Journal of Ambient Intelligence and Smart Environments*, Vol. 1, pp. 51–55, January 2009.

32. Duan, P. and H. Li, "Zigbee Wireless Sensor Network Based Multiagent Architecture in Intelligent Inhabited Environments", *Proceedings of the International*

*Conference on Intelligent Environments*, 2008.

33. Chen, W.-H. and W.-S. Tseng, "A Novel Multiagent Framework for the Design of Home Automation", *Proceedings of the International Conference on Information Technology*, pp. 277–281, IEEE Computer Society, Washington, DC, USA, 2007.

34. Tapia, D. I. and J. M. Corchado, "An Ambient Intelligence Based Multiagent System for Alzheimer Health Care", *International Journal of Ambient Computing and Intelligence*, Vol. 1, No. 1, pp. 15–26, 2009.

35. Hightower, J. and G. Borriello, "Location Systems for Ubiquitous Computing", *IEEE Computer*, Vol. 34, No. 8, pp. 57–66, 2001.

36. Liau, W.-H., C.-L. Wu and L.-C. Fu, "Inhabitants Tracking System in a Cluttered Home Environment Via Floor Load Sensors", *IEEE Transactions on Automation Science and Engineering*, Vol. 5, No. 1, pp. 10–20, 2008.

37. Augusto, J. C. and J. O'Donoghue, "Context-aware Agents - The 6Ws Architecture", *Proceedings of the International Conference on Agents and Artificial Intelligence*, pp. 591–594, INSTICC Press, 2009.

38. Aydemir, F. B., *Contract Based Cooperation For Ambient Intelligence: Proposing, Entering and Executing Contracts Autonomously*, Master's Thesis, Bogazici University, Istanbul, June 2011.

39. Bellifemine, F., A. Poggi and G. Rimassa, "JADE: A FIPA-compliant Agent Framework", *Proceedings of the Fourth Conference on the Practical Application of Intelligent Agents and Multiagent Technology*, pp. 97–108, London, UK, 1999.

40. Chopra, A. K., F. Dalpiaz, P. Giorgini and J. Mylopoulos, "Reasoning About Agents and Protocols via Goals and Commitments", *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pp. 457–464, International Foundation for Autonomous Agents and Multiagent Systems, Rich-

land, SC, 2010.

41. Foundation for Intelligent Physical Agents, *FIPA Personal Travel Assistance Specification*, 2001, May 2011.

42. Moraitis, P., E. Petraki and N. I. Spanoudakis, "An Agent-based System for Infomobility Services", *Proceedings of the Third European Workshop on Multiagent Systems*, pp. 224–235, 2005.

43. Spanoudakis, N. I. and P. Moraitis, "An Ambient Intelligence Application Integrating Agent and Service-oriented Technologies", pp. 393–398, Springer, 2007.

44. Cabri, G., L. Ferrari, L. Leonardi and F. Zambonelli, "The LAICA Project: Supporting Ambient Intelligence via Agents and Ad-Hoc Middleware", pp. 39–46, IEEE Computer Society, 2005.