Universitatea "Politehnica" București

Facultatea de Automatică și Calculatoare,
Catedra de Calculatoare

LUCRARE DE DISERTAȚIE

Guvernare agila intr-un mediu de inteligenta ambientala

Conducător Ştiinţific:
Olivier Boissier, Gauthier Picard, Andrea Santi

Autor:
Alexandru Sorici

București, 2011

University "Politehnica" of Bucharest

Automatic Control and Computers Faculty,
Computer Science and Engineering Department

MASTER THESIS

Agile Governance in an Ambient Intelligence environment

Scientific Adviser:
Olivier Boissier, Gauthier Picard, Andrea Santi

Author:
Alexandru Sorici

Bucharest, 2011

A very special thanks goes to my supervisors Olivier Boissier and Gauthier Picard at ENS Mines de Saint Etienne whom I am grateful for all their guidance and insights offered to the scientific topics that were the basis to this thesis.
I would also like to address a big thank you to Andrea Santi from Alma Mater Studiorum Universita di Bologna for all his contributions to the definition of the application scenario presented in this work and for very valuable validation and suggestions related to the design and implementation aspects of the application.

Abstract

Ambient Intelligence scenarios are starting to get an increasing amount of attention due to the development of technologies that lie in support of such systems. A key property of AmI applications is that of adaptability. In order to provide this requirement in a robust and efficient way, we look at the use of multi-agent system (MAS) organizations for the task of agile governance of AmI environments. Specifically, we look at possibilities of achieving reorganization in an endogeneous way, by combining bottom-up (agent built) and top-down (organization imposed) directives. We introduce an analysis grid that makes the mapping between scenario characteristics and the appropriate MAS adaptation technique. Further, we detail a model of adaptation for organization-centric applications and provide an example implementation within the scenario of a smart office/school environment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context: Ambient Intelligence and Multi Agent Systems

In 1998 the board of management of Philips commissioned a series of presentations that were supposed to investigate scenarios that would transform the high-volume consumer electronic industry from a "fragmented with features" world to a 2020 world of user-friendly devices supporting ubiquitous information, communication and entertainment (M. Weiser describes this in [41]). This is how the notion of *Ambient Intelligence* (E. Zelkha, B. Epstein et al [14]; E. Aarts et al [3]) came to be. In such an ambient intelligence world, devices work in concert to support people's everyday life activities in an easy and natural way. All the information and intelligence is then hidden in the network connecting these devices. The paradigm is characterized by systems and technologies that are [3]:

- *embedded*: many networked devices are integrated into the environment
- *context aware*: these devices can recognize you and your situational context
- *personalized*: they can be tailored to your needs
- ***adaptive***: they can change in response to you
- *anticipatory*: they can anticipate your desires without conscious mediation.

Note must be taken that we have underlined *adaptive* as an important attribute. This is because the focus of this work will be on providing adaptation capabilities to ambient intelligent environments. Indeed, it is easy to envision that scenarios which feature the interaction of multiple users within a complex and unpredictable environment (e.g. a Crisis Management System) will put an explicit focus on the adaptability component.

To provide for adaptation, a *governance* system must be installed that can state precisely and efficiently *when* and *how* a change needs to occur. The usage of the term *governance* (as opposed to regulation or control) comes from the fact that in the context of multi-agent systems the usual "metaphor" is that of human organizations. Given the concepts lying at the foundation of such an approach, the word "governance" is more suited than "regulation" or "control" which are more related to automatics.
We consider the field of *multi-agent systems* (MAS) to be very well suited for this task. MAS have themselves evolved into a new programming paradigm. This status brings with itself many advantages related to the development of tools, standards and engineering methodologies that allow designers to specify in detail *how* the multi-agent system must work and *what* it has to solve.

MAS have been studied extensively and the key attributes that make them so attractive for the role of AmI environment governance are ***autonomy***, ***decentralization*** and ***scaling capabilities***. In order for Ambient Intelligence to become a reality a number of key technologies are required, among which *Dynamic and massively distributed device networks*, *Human-centric computer interfaces (intelligent agents, multimodal interaction, context awareness etc.)*, *Dependable and secure systems and devices (self-testing and self repairing software, privacy ensuring technology etc.)*.

We can see that the above mentioned strengths of multi-agent systems make them ideal candidates for the management of such technologies. Different concepts are used to program MAS. *Organization* is one of the concepts used to express coordination/cooperation strategies in such systems. It is a good candidate to express the governance strategy in agent-based approaches for AmI in the sense that it is possible to govern/control the system while allowing the agents to change the strategy.
This is why in this thesis we are going to investigate the use of *Multi-Agent System Organizations* for the management of AmI related technologies and environments, showing how these can very well contribute to the requirement of adaptability.

1.2 Motivation: The room booking application

We have previously introduced our key concerns and shown that we want to look at how MAS organizations can provide for agile and adaptive governance in Ambient Intelligence environments. We are going to showcase these aspects in the context of the *room booking application*. The scenario we propose falls in the line of *smart working environment governance*, being concerned with the automatic management of event rooms in an office or school building. What distinguishes our scenario model from other similar ones is the fact that we add a requirement for adaptation, drawn from the changes that can occur in the environment, under the form of variations in user requests or user stated preferences.

In the room booking application, rooms are divided into 3 categories which can be often encountered in real life: *teaching rooms, meeting rooms and brainstorm session rooms*. To be included into one of these categories a room has to satisfy requirements regarding:

- format - amphitheater, office, laboratory, conference
- number of seats
- equipment - whiteboards, projectors, TVs, microphones, video-conference utilities etc

Users (professors, engineers, businessmen) can then demand the scheduling of teaching, meeting or brainstorm events in the building. According to the details of the request (made in term of room type, format, number of seats and required equipment), the event will be allotted to the appropriate room. Users also have the possibility to cancel or try and reschedule an event. They can also simply ask for information about the events already scheduled.

As mentioned earlier, in an ambient intelligence scenario which envisions the automatic handling of event scheduling, one can often encounter unforseen changes in the requests made by users (e.g. a sudden increase of requests for meetings during a buisness/conference session or an increase of brainstorm events during a workshop) as well as strong constraints in the user preferences contained in the requests (e.g. favorit room and time slot). These conditions motivate the study of adaptation prospects in this scenario and in similar ambient intelligence applications in general. More specifically, in the context of the room booking application, we have to determine a mechanism that allows for a dynamic and efficient reallocation of rooms in the building to the hosting of the type of event that is currently in peak demand.
In section 3.2 we will present a scenatio governance model that meets of the above described requirements of the room booking application.

1.3 Objective: Governance of an AmI application

Having presented the main concerns of our work and the scenario that will exemplify them we can state that our objective sums up to finding a solution for the problem of *Agile Governance in AmI applications*.

Ambient Intelligence applications are user-centered and as such the response time when interacting with them has to be very short. Speed is also required for the adaptation to the changes in the environment. These arguments justify the use of the word *agile*. Indeed, the management of the AmI environment has to be done in a flexible but robust way, that will allow for seamless and efficient interaction with the user. Like we mentioned earlier, we will look at multi-agent systems and the

notion of organization within MAS as the ideal candidate paradigms for this task. More specifically our objective is to show that adaptation becomes equal to the *change in the agent organization* and that this can be achieved in an *endogenous way* (with no external influence) following a combination of *bottom-up (agent built)* and *top-down (imposed by organization)* directives.

As we will see, such an envisioned combination has a different interpretation given the application scenario it tries to solve and the MAS model used to handle it. Therefore, a first contribution of this work will be the definition of an application analysis grid that proposes different "methodological" and "domain" criteria which will create a mapping between scenario characteristics and the most appropriate MAS adaptation techniques. Using the defined grid we will show that our proposed scenario falls into the category of *"Organization-centric systems"* and we will give a suitable model of reorganization for this case. We will demonstrate the functioning and effectiveness of the model by including it in the implementation of the multi-agent system that governs the room booking application.

1.4 Thesis Outline

The rest of this work is structured as follows.

Chapter 2 presents a state of the art overview of the notion of *organization* in multi-agent systems viewed from three different angles, those of *Self-Organizing Systems*, *Coalition-based systems* and *Organization-centric systems*. The way adaptation is achieved in each of these MAS models will be studied by means of a detailed review of established applications that have made use of multi-agent system architectures and techniques. Following the differences that will be uncovered, the chapter will end by giving the definition of the mentioned *application adaptation analysis grid* which will be used to provide the mapping between application-level characteristics and MAS adaptation and modeling specifics.

Chapter 3 will first provide an interpretation and comparison of adaptation in multi-agent systems along the lines of the grid introduced in chapter 2. It will then continue by mapping the room booking scenario to the analysis grid and identifying the appropriate model for the application itself and for the way that adaptation is to be provided, through reorganization, in the proposed scenario as well as similar ones.

Chapter 4 describes the realization of the models proposed in chapter 3 by detailing the system architecture and the frameworks and tools that were used. Special focus will be placed on the way in which reorganization capabilities are added to the JaCaMo platform [21] and on the development of the JaCa-Arduino project [2].

Chapter 5 will present the workings of the simulation platform that was developed to test the system. It will continue by providing an analysis of the obtained results and highlighting the required and observed behavior.

Chaper 6 ends this thesis by presenting conclusions of the work, pointing out the contributions that were brought, and by stating the perspectives for future development.

Chapter 2

State of the art

This chapter portraits a review of the state of the art concerning the notion of organization in multi-agent systems (MAS) as a scheme for cooperation and colaboration. The different levels of abstraction at which this notion is incorporated into an agent's knowledge representation and reasoning model will be pointed out.
The focus will lie on the way in which adaptation is achieved in several reviewed multi-agent system applications, based on the organization related concepts the agents work with. Bearing on the above mentioned study, the chapter will end by proposing a possible analysis grid that aims at giving a mapping of an application's input, output and processing dimensions to the most appropriate MAS adaptation approach.

2.1 Organizations in Multi-Agent Systems

Multi-agent systems have attracted much attention in the recent years as more and more complex applications are in need of the autonomy and scaling capabilities offered by MAS. With the advent of Ambient Intelligence [40], which is envisioned to have its significant boom in this period (2010-2020), the interest in agent driven applications will increase even more. To deal with the awaited pressure of having relevant results, researchers have started building tools (agent programming languages and frameworks), standards (FIPA ACL [25], KQML [16]) and methodologies for the development of multi-agent systems. But, since the field has such a wide range of applicable scenarios, different types of MAS engineering approaches have been defined, which take inspiration from domains ranging from the behavior of ant societies to the workings of full-blown human corporate organizations. This diversity of inspiration sources has led to the specialization of interests among researchers dealing with MAS engineering and simulation resulting in the apparition of (i) *self-organizing, emergent MAS*, (ii) *coalition-based multi-agent systems* and (iii) *organization-centered system* design. For each of these major areas of research, design frameworks and engineering methodologies have seen several years of development.

From Agent Centred View to Organization Centred View of Organizations in MAS

Each of the previously mentioned MAS approaches are concerned by the notion of *organization*, i.e. a cooperation scheme to achieve a global purpose. As we will see, this notion is, however, considered at different levels. A first distinction is made in [30] between what can be considered *agent-centered (ACV)* and *organization-centered (OCV)* views of the notion of organization. In ACV, organization is considered from the point of view of emergent phenomena in complex systems. Herein, designers focus on the components of the system-to-be, namely the agents themselves, by trying to construct local behaviors and interactions, the result of which will be the global (emergent) functioning of the system. OCV is the opposite perspective, where the attention lies on engineering of effective cooperation or regulation mechanisms that allow for the design of complex multi-agent systems. Figure 2.1 shows how the 3 mentioned categories of MAS models fall into one of the two opposite views of organization. The paragraphs that follow do a brief introduction of each model, further explaining the reasons for the classification observed in the figure.

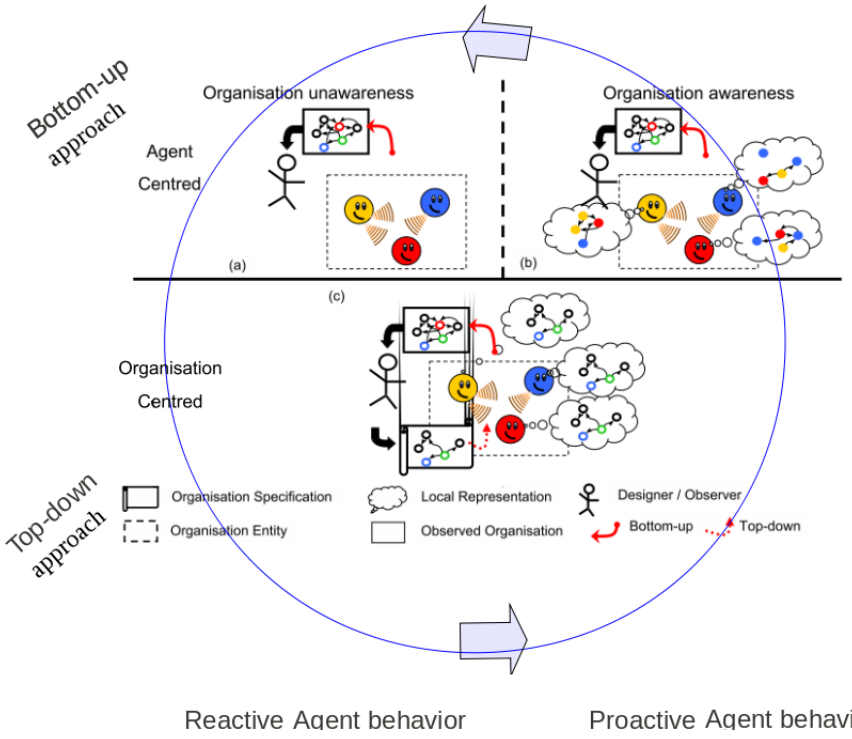


Figure 2.1: From Agent-centered view to the Organization-centered view. The blue arrows suggest the continuous combination of bottom-up and top-down directives as well as reactive and proactive agent behaviors during the adaptation phases. Adapted from Picard et al [30]

Self-organizing systems function without central control. They are based on key local behaviors and peer-to-peer interactions that can often also lead to the building of a global cooperation pattern, i.e. an organization, of which the individual agents are not aware of. An important particularity of self-organized systems is their ability to quickly respond to environmental changes and produce a new state of the organization. Serugendo and Karageorgos [12] aim at defining the concepts of *self-organization* and *emergence* and provide a state-of-the-art survey of the different classes of self-organization mechanisms applied in the MAS domain. They show how the property of adaptability can be implemented in such systems by means of direct interaction, stigmergy, reinforcement, cooperation frameworks or generic architecture designs. The now already intense study of self-organizing systems has also lead to the development of methodologies for the construction of such systems, e. g. ADELFE [31].

In *systems based on coalition-formation* agents start to form inner models of interactions (e. g. degree of utility achieved by the interaction) and dependence relations (relying on a resource or task provided/solved by another agent) of their neighborhood. These models are formed according to predefined social rules and patterns (which thus help in the actual coalition formation process). In such systems, agents are often modeled as either cooperative or self-interested, but regardless of their "personality", creating or joining a coalition usually results in a significant utility gain for every member agent. *Coalition-based systems* have also been looked at from various angles. [10] and [8] investigate the use of market analogies, Sandholm and Lesser [36] look at the game-theoretic aspects of *coalition formation* while the works in [37] and [38] present studies of algorithms for the construction of such systems.

Picard, Boissier et al [30] note an important aspect that can be observed in both of the above mentioned approaches; it is the fact that the agents are "the engine" of the system, a characteristic pertaining to ACV. This means that any organizational notions are either directly included at design time in the agents knowledge representation mechanisms or that they "exist as observable emergent phenomena which state a unified bottom- up and objective global view of the pattern of cooperation between

agents" [30].

Yet, some of the above mentioned methods of MAS engineering can sometimes suffer from drawbacks such as the unpredictability of interactions or of desired system behavior (due to possible emergence). This is why a more strict, normative view to multi-agent system construction has been promoted for some time now: that of *organization-centric development.* This view considers the direction in which the organization exists as an *explicit entity of the system* [30]. The specifics can be determined either at design time or dynamically, at runtime, by the agents themselves, but the important aspect is that the organization is installed in a top-down manner in order to constrain or define the behavior of the agents. The system in its entirety now has a component that can provide a guideline for the correct and efficient achievement of the specific global goal. Like in the case of self-organizing systems, the study of OCV in MAS has led to the development of frameworks such as MOISE+ [20] and OMACS [11], that allow for robust support in the design and management of *organization-centered agent societies*, as well as *modeling methodologies* like Ingenias [28] or Gormas [5] which can provide the necessary guidance needed by engineers.

One can notice that the MAS engineering field is starting to slowly become more specialized in each of the above presented directions as more and more studies are carried out. All these approaches are talking about *organization*, i. e. recurrent patterns of cooperation, that are defined by the agents during the functioning of the system or defined by the designer. All these approaches are faced to the openness and dynamism of the applications and environments in which applications are deployed. Thus the adaptation of these organizations is a very important feature.

Though we use the notion of organization (seen as embedded in ACV and explicit in OCV), our study focuses on applications in which the agents themselves are the locus and engine for both normal functioning and change of the system. They are the place where the organization is interpreted, "internalized" and adapted. Besides presenting a classification of ACV and OCV systems, Figure 2.1 also shows an important focus of this work, namely, the combination of **bottom-up** and **top-down** directives for organization manipulation. The bottom-up view specifies that organizational structures are constructed and maintained by the agents themselves (a view more consistent with ACV). The top-down approach, on the other hand, sees the organization as being imposed on the system as a defined cooperation scheme, used to divide attributions among agents and guide them towards the achievement of the global goal (view consistent with OCV). Trying to continuously combine these two approaches is a challenging task, as we will see that the very notions of *bottom-up* and *top-down* are interpreted differently depending on the abstraction level the agents in the system use to model and reason about the application scenario in which they operate.

In each of the above presented MAS design alternatives, member agents employ and reason about different notions and structures of themselves, the environment and their fellow agents. One of the aims of this analysis is to highlight which *notions and structures* are particular to which multi-agent system engineering method (self-organizing systems, coalition-centric systems or organization-centric systems). We also believe that these agent internal representations (which we are going to detail in section 3) are better-suited models for specific application scenarios. Indeed, we will try to capture *application level* characteristics and *the way* in which they are best modeled as agent knowledge. Such application characteristics also influence the appropriate functionality of the MAS, i. e. the explicit process the agents are better off using to solve the problem at hand (*how* to interact, *what* notions to exchange etc.). A very important feature to notice here is that the property of adaptability is enabled differently depending on the way in which the MAS operates on a normal basis (the above mentioned functional process). Therefore, *ways of providing adaptability* can also be seen as *specific to certain characteristics of the scenario* in which the multi-agent system is used. This particularity is best understood as a difference in the abstraction level of the notions and reasoning mechanisms employed by the agents when they engage in a modification of their organization (from simple own behavioral changes, to structural changes of the local neighborhood, to modifications in role allocation and normative specifications)

An important result of this work is therefore that of providing an *analysis grid of application and scenario characteristics* relating to the *given inputs*, the *desired outputs* and *the process of turning inputs into outputs.* The interpretation of this grid will show how these attributes promote different levels of abstraction with respect to *organizational notions* that need to be used by agents to allow for adaptivity in the context of the given scenario. These levels of abstraction will then, in turn, promote the use of one or the other MAS modeling technique.

2.2 Adaptation in Multi-Agent Systems

It is our objective to analyze the applications that are managed by means of multi-agent systems from data centric and information flow (interaction) points of view. By looking at specific attributes of the data handled by agents and characteristics of the needed agent interaction, we want to determine a possible break-up of the scenario into parts, the functioning of which can be best modeled using techniques from *agent-centered self-organizing systems*, *coalition-based MAS* or *organization-centric approaches*. That is to say that, depending on the application, both the normal functioning as well as the adaptability that is required, can be most usefully modeled using conceptual notions and abstractions proper to one of the earlier mentioned categories of MAS.

2.2.1 MAS Adaptation - General Overview

In order to perform this analysis, we plan to look at the specifics of some of the well-established applications and scenarios where multi-agent systems have been successfully employed to handle the job. As such, this section is dedicated to a more detailed analysis of *where* and *how* different multi-agent systems have been used in practice. As explained in the introduction we will consider examples pertaining to each of the 3 identified categories of multi-agent systems.

Self-organizing / emergent systems (ACMAS – agent centered multi-agent systems)

The term self-organization "essentially refers to a spontaneous, dynamically produced (re-)organization" [12]. We refer here to this notion as both the engine to achieve an organization from within the system (emergence effect) as well as the mean of providing adaptation properties to a MAS. In such systems one embodies non-complicated behavior or local structure change abilities at an agent level. These then drive the re-organization and adaptability of the system. Behavior changes are often done on a reactive basis (controlled by reinforcement from the environment) and structural changes (links, link weights, group membership) are not assigned an explicit, meaningful typology. But, we will now shortly present applications modeled as such systems that were successful in achieving their goals. A good review of self-organization based real life applications has been done by Bernon et al. [6].

The *Self organization of User Communities using Middle Agents* [39] is meant to support information exchange between user communities by providing each user with an agent representing their interests. The application also features middle agents which act as brokers for different user agent queries. The role of the middle agents is to form groups of user agents with similar interests around them in the attempt to make the system scalable and provide a timely and efficient response to user queries. From a modeling point of view the application can be seen as clustering based on similarity. The agents have means to compute and reason about a measure of similarity that they then try to improve. An important focus is set on improving characteristics of the network scalability, in the meanwhile satisfying individual component requests. One can make notice of the fact that the identified organization has an almost flat structure - user agents and middle agents – and that there is a predominant "horizontal" interaction between agents of the same type.

The *Self-organization for the School Timetabling Problem* [29] is a classical example of a constraint satisfaction problem (CSP). A timetable for a certain duration has to be found while respecting the explicit constraints (availability, specialization, equipment needed etc) of different stakeholders: teachers, student groups, classrooms. In the system representative agents (RA) hold stakeholder constraints and launch several cooperative booking agents (BA) that are able to interact and construct the schedule in an emergent manner. The constraint satisfaction component is handled by splitting the problem into individual concerns and allowing for cooperative interaction (the AMAS theory [9]). The system uses the reactive behavior of BAs to allow them to respond to perturbations - changes in stake holder's constraints – Non-cooperative situations (NCS). Again, one can notice a flat structure - RAs, BAs - and a predominant horizontal interaction between agents of the same type (BAs).

Self-organization for Land Use Allocation [13] is based on a real-world problem, and applies a self-organizing approach to simulate the assignment of land use categories in a farming territory. The

problem exhibits a function to optimize, while respecting a set of constraints, both local (compatibility of grounds and land-use categories) and global (ratio of production between land-use categories). The scenario is resumed to an optimization problem subject to given constraints. Agents form groups that have a goal to satisfy by conquering spatial zones in the environment while respecting some constraints. All agents have to search for the more attractive land zone. Self-adaptation is achieved by solving local encounter problems based on the strength of the groups the agents are members of. Again, one can see that the problem solving is done by flat agent interactions that are governed by simple rules which specify the attractiveness of a land zone and the way to solve conflicts between two agent groups that prefer the same zone. The simplicity of these interaction guidelines is what makes the system adaptive, allowing for a very dynamic functionality.

The *Localization and Tracking using Self-organization* [17] presents a localization task as finding the position of an object (or more than one), mobile or not, in a well defined referential location. The tracking problem is to provide a succession of positions that are spatially and temporally coherent. The application proposes a reactive model to tackle this issue. Agents are equivalent to weighted particles evolving in an environment of force field. Their behavioral model is inspired by a model of flocking, being expressed through a formulation taken from Newtonian physics. Thus one can observe that the influence of the environment on agents is greater than that of agents on environment. In this application one can also observe that no notions of structural interaction (links, weights on links) are explicitly defined.

All these scenarios and additional ones are analyzed in more detail in Appendix Appendix A.1.

One feature that all the above described applications have in common is that, as it is usual in ACV, the aspect of organization is embedded directly in the agent's internal workings. More over, the level of abstraction of organizational notions resumes to representations of the local neighborhood or simple group memberships (but with no consideration of an explicit group objective). The adaptation of the organization is then accomplished by using simple reactive behaviors that can modify these structural representations (links, neighborhood).

Later in this section, based on the detailed analysis of applications like the ones in Appendix Appendix A.1, we will look at characteristic properties of application scenario inputs, outputs and processing of inputs to outputs that naturally mandate a system design where the self-organization, adaptive component acts on a per agent basis.

Coalition-based systems (CMAS)

In the study of adaptivity for coalition-based systems we are interested in characteristics of applications that would make them in need of a process that requires the presence and opinion of a whole subgroup of the organization. For what concerns such systems, it is a different interpretation of (self-, re-)organization that we are interested in. We want to study the nature of applications that require a group decision when dealing with re-organization processes, or, even if the decision is taken only by the one individual, there will be an immediate impact on the members of its group. Now, since a group decision is required concerning a matter of change within the coalition, the application model naturally evolves an additional abstraction level where agents must have notions of the global coalition objective, of responsibilities within the coalition. Agents of the system must be able to model current and past interactions and dependence relations (resource or sub-task solving dependencies). The way in which the system adapts will then take these representations into account.

The Distribution of tasks [37] is a generic problem encountered in MAS. Given a set of agents and a set of tasks which they have to satisfy, this problem considers situations where each task should be attached to a group of agents that will perform the task. Task allocation to groups of agents is necessary when tasks cannot be performed by a single agent. Also, situations often arise where groups will perform a task in a more efficient way with respect to single agent performance. A main characteristic is the decomposability of tasks. Yet the different subtask sets can be disjoint which allows agents to form groups (coalitions) with the explicit goal of solving the given list of subtask,

whereby each member agent contributes with different resources and know-how that it has. In this way, dependence networks are formed on a horizontal level, often resulting in a flat structure. At most, a one-level hierarchy can be formed for task achievement control.

The *electronic marketplace* [4] application focuses on self-interested agents that trade goods and knowledge in a large-scale multi-agent systems. It uses coalitions as a means of grouping agents together to improve cooperation among them leading to individual benefits. An important idea is the extension of the concept of task oriented coalitions to long-term coalitions based on trust relationships between the agents. In the application clear (mutual) dependency networks are observable (seller-buyer relations). There is a single objective of a coalition of sellers and buyers: purchasing of goods. The coalitions and associated dependencies represent different ways of achieving this goal. It is clear from this analysis that adaptation in the system is provided by means of choosing the coalition (group) that offers the best value of the agent's price-quality utility function. Re-organization is then equivalent to changing the dependence links, the possible weight of dependence links or just changing coalition membership altogether.

In the *Distributed Sensor Networks for real-time tracking* [38] scenario, a distributed, incremental approach to self-organization through bottom-up coalition formation is presented that is applied to the distributed sensor network (DSN) of the EW Challenge Problem. The setting consists of homogeneous sensor agents distributed throughout a region. The agents are fixed and communicate using an eight-channel RF (radio frequency) system in which each can use only one channel at a time. An organization in such a domain helps facilitate the efficient assignment of tracking tasks to particular agents and limit contention on communication channels. The application features a single, long lasting task modeled as different spatially distributed similar sub-tasks. The usefulness of coalition-based approaches then becomes observable in the fact that manager agents can negotiate (form resource dependence links) about the allotment of sensor agents to the region which they are responsible for. In this case, a one level hierarchy is observed and the power relations are formed by the agents on the top level of this hierarchy (the manager agents).

The *Vehicle routing* [36] focuses on the task of delivering orders to target destinations. Vehicles start out from a depot, may pick up orders from other depots, deliver them and return to their original starting point. The vehicles form coalitions to help them share delivery orders and minimize the total route length. The scenario has a high number of tasks of similar type (delivery orders) that have distinguishing parameters (maximum load of a vehicle and maximum route length). It also features a medium-large number of working elements (vehicles) with complementary operational capabilities (starting point, distance from given depots). Coalition formation and change approaches can then help by the building cooperative dependencies based on the complementarity of these capabilities. As the delivery orders change, so will the coalition memberships and the implied cooperation relationships.

Like stated earlier, the use of coalition-based systems sees the definition of a new level of abstraction when it comes to modeling the organization. Agents are now given means to model the history of interactions in the coalition and to assess the utility of maintaining certain relations (links) to other agents. Members of a coalition can also be aware of the global purpose of their group. Typically, agents employ an utility function based model to compute the use of joining a coalition or of interacting with a another agent. Adaptivity is then achieved by changing these structural characteristics of coalition membership, dependence, cooperation an power links so as to maximize the agent's internal utility given the current context of operation. Though from the organization point of view one can see the formation of one-level hierarchies, the majority of interactions still in such systems still only need to occur on a "flat" level. But, unlike self-organizing emergent systems, the interactions have more directionality, which is dictated by the dependence or power links that the are kept by the agents. Another important aspect that can be observed analyzing the above mentioned applications is that the agents need not reason about notions such as roles and norms inside a coalition. Rather, the agents form and maintain groups based on the utility function measurement that they are provided with at design time. Inside the coalitions, the agents play certain roles, based on the dependence relations they are currently having with other member agents (supply a resource, wait for a task execution, contractual discounts) without maintaining an explicit representation of the attributions

of that role. The complexity of the scenarios where coalitions are used does not require the inclusion of this additional level of abstraction.

Organization-centric systems (OCMAS)

With organization-centered system views, an application model reaches its highest abstraction level. In such settings agents are endowed with a declarative overview of the structure (roles, groups, explicit link types) of the organization they are part of and of the norms (explicit role obligations) that govern its functioning. More over, it is possible to show that the initial design of the MAS can be done solely on the grounds of this organizational view, in the sense that the specifications provide the main functional guidelines of the system. As a consequence, from the adaptability point of view, agents must possess these organization specific notions and the ability to reason about them because the applications in which such agents are deployed will make it very likely that the reorganization decisions that they will be faced with will concern changes related to these kind of properties. In OCMAS, the system goes a step forward from intra-group (coalition) influence. Whereas in coalition-based models it is common that two different coalitions do not interact too much with each other, OCMAS usually have a tighter influence bond between groups. Because of the global description property, the reorganization decision taken by members of the system can influence several groups (and the interaction between them) at the same time.

The *Robot Soccer Team* [20] sees the formation of an organization that manages the strategies and goal-scoring tactics of a robot soccer team. Agents play roles such as attacker, mid-field or defender and have specific missions that tell them how to score a goal. We can see that the problem at hand poses a clear separation of concerns. Each group (attackers, mid-fielders, defenders) has clear defined attributions and playing strategies which contain clear dependency mappings and flows of interactions. The complexity of the normal functioning of the system implies that adjustments of the current organization require changes at its abstract definition, not just the instantiation. It requires reasoning on the abstract components of the organization and is not limited in impact on the one group. An example would be the change in formation from 4-4-2 to a more defensive 4-5-1 which not only sees changes in the roles across groups, but also changes in the interactions (the play strategy).

The *Travel Agency* application [15] is an example in which clients try to get products from an agency considering as a brokering agent for product providers. The system identifies clear groups (client group, provider group, contract groups) and interactions within and between these groups (asking the brokers for a given service, making offers, signing a contract). The AGR (Agents/groups/roles) model is used to describe the organization in this case. One can observe a clear separation of inputs and of the actors manipulating the inputs, e. g. members of the producer group have certain responsibilities (to produce goods and respond to broker requests), while members of the client group have certain permissions (they are allowed to ask for goods) and obligations (they must channel their request through the broker). The application also features clearly defined dependency mappings and flows of interaction between system actors (e. g. client-broker-producer-broker-client; buyer-seller). A change in the products offered or in client preferences would require changes at the organization instantiation level, but still, any new agents would have to keep account of the regulations on behavior and interactions imposed by the organization specification.

The Dutch procedures for crisis management [4] are a description of the way in which crisis situations are handled in Holland. The paper explains the different organizational constructs that form in such scenarios and the way in which they change (role additions and changes in attributions) depending on the severity of the incidents. Since the scenarios involve the handling of crisis situations, a clear hierarchy of the roles is defined. There is a predominant vertical communication and chain of interaction. Abundant interactions take place within a group (e. g. ROT or CoRT), but the inter-group ones are more important even if smaller in number. The most important aspect described in the paper is the way the organization handling the situation adapts depending on its severity (from routine incidents to large scale disasters). It shows that changes in environment require not only changes in the organizational structure (new roles, new group constitutions) but, more importantly, changes in

the attributions and communication specifications given to those roles. This shows that adaptation can only be achieved if members of the organization think about changes at the abstract meta-level of the organization specification, instead of just its current instantiation.

In the *Paper review* example [15], the process of submitting a paper to a conference is simulated. The agent organization maps to groups like the Program committee or the submission group. It simulates the entire process from article submission to acceptance or refusal notification. Like in the previous applications, a clear separation of inputs and actors manipulating the inputs can be observed (e. g. program chair, program committee member, author etc). There is a clearly defined flow and orientation of interactions (author submits → submission receiver → PC chair → PC member → evaluation group → review manager → PC member → author). Like in the case of the Travel Agency application [15], adaptation would occur in this case at an instance level as well (e. g. an agent changing his role from PC member to reviewer). Again, the important thing to stress out is that while doing so, the agent needs to be aware of the meta-level organizational specifications that govern the functioning of the new role it is about to play.

In the list of OCMAS applications presented above, the *Dutch procedures for crisis management* [4] is not an actual simulation, but just a description of the real world measures taken to handle crisis situations and the actors that realize them. Yet all problem statements show why the addition of an explicit, meta-level organizational description is required for both normal functioning as well as adaptation in the face of environment changes. The complexity of the scenarios and the number of specialized subproblems that arise require the definition of clear guidelines that allow for an efficient achievement of the global goal. Adaptation requirements no longer impact only structural changes and even if they do, the impact of a useful change goes beyond the local-neighborhood or local group level (as was the case in self-organized or coalition-based systems).

2.2.2 Definition of Application-centric Adaptation Analysis Grid

We stated in the beginning of this section that our objective is to analyze the applications that are managed by means of multi-agent systems from static (data centric) and dynamic (interaction) points of view. Such application-level attribute values should then show that normal functioning and, more importantly, adaptability, can benefit greatly from using conceptual notions and abstractions proper to one of the 3 mentioned categories of MAS (ACMAS, CMAS, OCMAS). The in-depth study of the application scenarios described above (given in detail in Appendix A.1, Appendix A.2 and Appendix A.3) allowed us to introduce an application-centric adaptation grid which we present in this subsection.

In a discussion about the characteristics of self-organizing systems that they carry out, Bernon et al [6] distinguish between static (descriptive) criteria of an approach and those belonging to the dynamical aspects of the problem solving process. In the analysis we want to pursue, we will take a similar, yet distinguishable approach of investigation. Instead of pursuing characteristics of a given MAS approach, we focus on the attributes of the applications and scenarios where MAS can be used to solve the problem. Like explained earlier in the introduction, the values of these attributes promote the use of different levels of abstraction with respect to organizational notions that need to be modeled by agents to allow for adaptivity in the context of the given scenario. Since we have shown that the mentioned abstraction levels map well to a certain category of multi-agent systems, the implied consequence is that these application-level attributes will in the end promote the use of one or the other MAS adaptation technique: *self-organizing agent-centered systems (ACMAS)*, *Coalition-based systems (CMAS)* and *Organization Centric systems (OCMAS)*.

As shown in figure 2.2, our *static view points* refer to characteristic attributes of

- *the application scenario inputs*
- *the application scenario outputs*

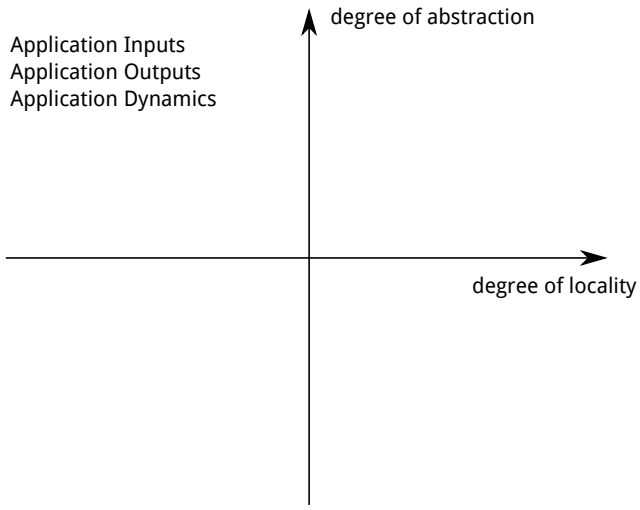


Figure 2.2: Description of the application view points and the conceptual axes that make up the 3 proposed adaptation analysis grids

The *dynamic views* are related to distinguishable attributes (*patterns of interaction, directionality of communication, area of influence* etc) of *the process* that turns the input data into the desired output objective.
For each view point, we provide attributes along two main conceptual axes (see also figure 2.2):

- *degree of abstraction* - from physical, quantifiable values to high-level, abstract concepts
- *range of influence (degree of locality)* - from local social - neighborhood to organizational inter-group influence

Degree of abstraction denotes the type of concepts agents have to deal with and model, whereas *degree of locality* relates more to properties of the functional process (turning inputs into outputs) and it does so by examining the way in which attributes from the degree of abstraction axis are exchanged between agents, by looking at the direction and spread of these interactions.

In Section 3.1, the presented multi-agent modeling possibilities will be mapped on the resulting analysis grid and we will show that the mapping forms a continuum based on the examples that we have studied. Additionally, comments will be made on properties that distinguish between different types of self-organization.

Application Inputs Analysis Grid

This view point considers the types of input data that a designer has to look at when studying the application that he has to model. From the analysis carried out we can distinguish the following attributes on the ***degree of abstraction*** axis:

1. physical continuous data – the application considers numeric values characteristic of a complex, non-linear physical or simulated processes
2. physical discrete data – is considered for high-level simulation or modeling applications where input data is quantized
3. partitionable data – the term partitionable refers to the fact that the application input can either be seen as tasks that are easily split into subtasks or that data has a sense of locality attached to it (i.e. one can assign a spatial or temporal component to it which means that agents will deal with different kind of values depending on where they work). The nature of data handled in the subtasks as well as the criteria used for the division of initial input may involve both physical and conceptual attributes (abstract properties of objects).
4. structured data – the application considers working with properties and objects that can be modeled by components and the way these are connected with each other. Data can be parti-

tioned into a small number of categories. The resulting division along distinguishable attributes can yield either many instances of few subtypes or few instances of many subtypes.

On the ***range of influence*** (degree of locality) axis we identify the following attributes:

1. environment acquisition and exchange – it explains the fact that within the application agents of the system will acquire their data only from the environment. Any data exchange will also occur only via the environment (stigmergy)
2. environment - local peers acquisition and exchange – additionally, data can be obtained from agents situated in the local-neighborhood
3. intra-group acquisition and exchange – applications in which agents usually obtain their data or tasks from another leader agent. The agents that receive work load from the same leader can be seen to form a group and any data will be then exchanged within the group.
4. inter-group acquisition and exchange – applications that establish several leader agents which feed data and workload to other member agents. The difference from the previous attribute is that the leaders themselves are strongly interacting. Thus the data exchange happens also on an inter-group basis.

Application Outputs Analysis Grid

The ***range of influence*** conceptual axis has the same attributes as for the input direction. The difference lies in the interpretation of these attributes. Whereas in the input case they represent ways in which the input data is acquired and exchanged by the agents, for this case they show the parties that are affected by, or interested in, the results of the entire (or partial) solving process.

For the output direction, the ***degree of abstraction*** axis considers the application objective on the most general level. We consider the following gradations:

1. numerical estimation values – applications in which the purpose is to provide a most accurate and dynamic prediction of an usually complex process
2. clusterization / repartition – applications where the goal is to obtain the best repartition of system entities based on some factors of similarity
3. optimization / constraint solving – the objective is to obtain the best value for a complex functions that model a specific constraint
4. short-termed strategy / decision making – the purpose is to come up with an efficient decision or short term strategy that allows the system to best achieve the task at hand. This attribute describes the system purpose at an abstract level, since it relates to the search of a strategy, which is a means of achieving a goal, rather than the goal itself.
5. long-termed strategy / decision making – the same description as above applies. The difference is that in this case, the application goal is that to *maintain* a specific state of system affairs, which requires the development of a long-term complex strategy.

Application Dynamics Analysis Grid

The ***degree of abstraction*** axis contains concepts that have to be interpreted as the notions that the agent is able to reason about. Furthermore, they express *the way* in which these notions are reasoned about, be it in a more *reactive self-centered way* or a *goal-sustaining, peer modeling, proactive way.*

We identify the following attributes:

1. own structure and behavior awareness / reactive control – during interactions agents employ a simple behavior brought forth by strong reactive routines that allow them to respond quickly to changes. They may have a structural model (links, list of agents) of the local neighborhood

and can form a sense of their own dependency on a service provided by another agent (to which they have a connection)

2. peer structure awareness / reactive control – scenarios in which agents are required to model not only their own relations to others, but also the structural relations of neighboring agents to their neighbors. Thus agents become aware of the structure of a group of agents that often interact with each other. The way they manipulate and use this knowledge is still done in a reactive way, leaving room for greater agility of change in the structure.
3. peer behavior awareness / proactive control – the agents are required to be able to form a model of the behavior of their peers. In such applications, agents usually have a goal that they pursue. Local peers can either hinder or help them in achieving that goal. That is why agents need to model their encounters with their peers and proactive plan formation is used.
4. group structure awareness / proactive control – the scenario requires that member agents form dependence or cooperation groups. Links between agents can be given a type and the agents can pro-actively reason on how to build up these connections so as to be able to solve their own task as well as the global goal of the group they form in the most efficient way.
5. group behavior awareness / proactive control – in applications where it is required to come up with a good strategy or a decision on a complex matter, agents are given the means to reason about functional behavior and to model how subgoals (and the required behavior to solve them) are assigned to groups of agents. Not only can they see what the interaction inside a group looks like, but they can model what relations exist between the groups themselves.

Whereas the previous axis has attributes that relate more to what the agents reason about, the *degree of locality axis* tells us *who* it is they share these notions with and *how*. It is a characterization of the *entities involved* in an interaction and of the *predominant direction* of those interactions.

On the ***degree of locality axis*** we consider the gradations:

1. self - environment interaction – applications in which the agent communicate in a stigmergetic approach (indirect interaction via the environment)
2. self - peer flat interaction – agents communicate with any of their local peers. The information sharing network the agents build has a flat structure. All agents receive the same information and they can then filter out the elements of interest.
3. self - peer interaction with increasing univocity – agents still interact with local peers but the increasing univocity means that patterns of information flow (dependency relations) form within the network of agents.
4. self - superior interaction with increasing univocity – in addition to the above description, the application requires the agents to be able to establish a hierarchy of information exchange within a group. The dependency relations can now have a master-worker dimension.
5. self - superior unilateral interaction, increasingly unilateral – applications in which the hierarchical dimension of interaction *within a group* has a greater importance than the peer-to-peer one .
6. group - group interaction with increasing univocity – this attribute is similar in description to the self - superior interaction with increasing univocity. The difference is that applications labeled with this attribute have a much more complex distribution of responsibilities meaning that agents have to model interactions on an inter-group basis.
7. group - group interaction, increasingly unilateral – the same description as above applies, except that the hierarchical, inter-group dimension has a greater importance then the one between groups of the same level within the hierarchy (typical of strict chain of command applications).

The just introduced analysis grid - the *degree of abstraction* and *degree of locality* axes defined on each of the mentioned directions (input, output, process) and the accompanying gradations - are the result of a detailed analysis of the reviewed applications (Appendices Appendix A.1, Appendix A.2 and Appendix A.3). Section 3.1 will see an in-depth comparison of the 3 categories of multi-agent systems along the lines of this grid.

Chapter 3

Theoretical Models

In this chapter we perform a detailed interpretation of the application analysis grids defined in the last chapter. We are then going to see how this interpretation fixes the room booking scenario into the category of organization-centric systems. As such we are going to identify an appropriate model for the application itself as well as for the way that adaptation is to be provided.

3.1 Analysis of Adaptation in MAS

This section is dedicated to the analysis and comparison of the different applications issued of the three families discussed in section 2.1. We aim at comparing them along the analysis grid introduced in the previous section.

3.1.1 Analysis of Adaptation with respect to Application Inputs

The input direction is a static factor and looks at the types of input data that are particular to a given application. This data typology and the means by which it is acquired in the system leads to different ways of representation inside the agent's knowledge base, ways that we consider to be somewhat particular to one of the 3 considered MAS categories. The comparison that follows is based on the attribute values along the 2 axes introduced in section Section 2.2.2.

Considering the application descriptions done in the earlier sections one can observe that on the first axis *(degree of abstraction)* the ACMAS views would map onto the first and the last gradations. Normally, the problems which deal with such types of input data have a very high change rate, or are difficult to predict. As such, it is natural that MAS set to handle these kind of inputs have to focus on properties such as *anytime ending*, *robustness* and *fast adaptability*. ACMAS is best suited for that and trying to apply apriori strict organization notations would just lead to the system being burdened with trying to keep account of these views when adapting. The same line of reasoning applies when looking at the degree of locality axis. When dealing with the type of data described above, systems usually acquire it directly from the environment and exchange it with peers, either directly or through the environment itself. It also often happens that the data and the agents that share it fluctuate strongly making them highly unpredictable. Imposing explicit, meta-level obligations on the way this data is obtained and shared is thus either not feasible or would just add load to the system's operation (the management of such meta-level data in a rapidly changing agent environment).

Moving to coalition-based systems, CMAS, we can see that they would fit gradations 2 and 3 of the *degree of abstraction* axis. CMAS are well suited for scenarios where input tasks can be split into subtasks (partitionable data), where one can assign dependencies between tasks and resources needed to solve them. The *structured data - few subtypes, many instances* tag also applies to CMAS because it can often happen that such systems deal with task settings where there are a lot of instances of similar assignments which only vary in the amount and type of resources (physical, computational

etc) they require to be solved (e.g. Vehicle routing [36] - deliveries all have the same attributes: load and length, but the values for each of these differs from target to target).

For all such cases, system agents would try to team up to share their resources and help each other out in solving the tasks at hand. By establishing partnerships they can obtain added value and reduced losses. Given the dynamics of the problems that come their way, the agents might have to continuously keep establishing new partnerships and break old ones up when they no longer can obtain any rewards by maintaining them. Thus we can see that the process of adaptation in CMAS is strongly related to structural notions like dependence links or coalition membership and agents focus on how to change these. Therefore, the organizational instrumentation of such systems must offer ways to model exactly these structural notions. Adding extra and explicit representations of the notions of *role*, *obligation* and *global goal* at the organizational meta-level is not required when no significant interaction exists between the coalitions and the only thing that can be changed are the links and coalition memberships. No change in behavior or responsibilities of an agent would occur as a result of him changing the agents he depends on for resources or the coalition he is a member of.

Lastly, all the apriori definitions of role, group and obligation become a need when we deal with information types of the last gradation of *the degree of abstraction* axis (structured data). In cases where data is of a structured multi-subtype format, one generally wants to design systems that deal with each part of the problem separately and then see how to integrate the results from each individual work-group into the final global output. This is how the Dutch crisis management [4] works, for instance, by establishing clear attributions for each team (the ROT, the CoRT) and setting persons in charge of the information flow at each team level. Thus, on *the degree of locality* axis, OCMAS systems would be placed on the last two gradations (superior - intra-group, superior - inter-group) for information acquisition and exchange. In the paper review application [15], a submitted article arrives at a reviewer via the submission receiver, which sends it to the PC chair, which asks a PC member to setup an evaluation group to review the paper. Information flows inside the evaluation group between reviewers and review manager and between evaluation group and PC group via the review manager and the PC member that created that evaluation group.

The important thing to note here is that applications dealing with such kind of specialized input task division and controlled information flow are well modeled using OCMAS tools and notions. If a change in the working environment requires a change in the organization, then agents of the system will be well off taking such organizational concepts like capabilities, roles, groups and obligations into consideration when trying to adapt because these concepts proved to be valuable in managing the system's objectives to begin with.

3.1.2 Analysis of Adaptation with respect to Application Outputs

The output direction is also a static factor that shows a typology of application objectives and the parties that are affected by, or interested in, the results of the entire (or partial) solving process. The following comparison is based on the attributes defined in section Section 2.2.2.

Looking at the conducted application investigation from the *degree of locality* view point, we see that in ACMAS the output usually affects the environment (like in the Flood Forecast application or the one for Localization and Tracking) or the environment and other peer agents (like in the Timetabling Problem, the Formation of User Communities Problem or, generally, assignment and clusterization problems modeled with MAS). For CMAS, since the input data usually creates a dependence network, the output interests coalition peers or the agent controlling the coalition (in case of a one-level hierarchy - intra-group relation). Because problems are well divided amongst agent work-groups and there exist managers handling each group, OCMAS output is usually destined to be handled using inter-group relations as it works its way up to the top of the organization.

It can be observed that the *degree of abstraction* axis does not contain an exhaustive listing of possible application outputs. Still, the given gradation is quite exemplary for the scenarios we have studied in our previous sections. One can notice that the ACMAS applications would map onto the first 3 gradations, the CMAS would fit marks 3-4 and OCMAS would be assignable to marks 4 and 5. When looking from the re-organization viewpoint the kind of considerations that applied for the input data, apply also for the output. For ACMAS we saw that the input values showed great variance, complexity

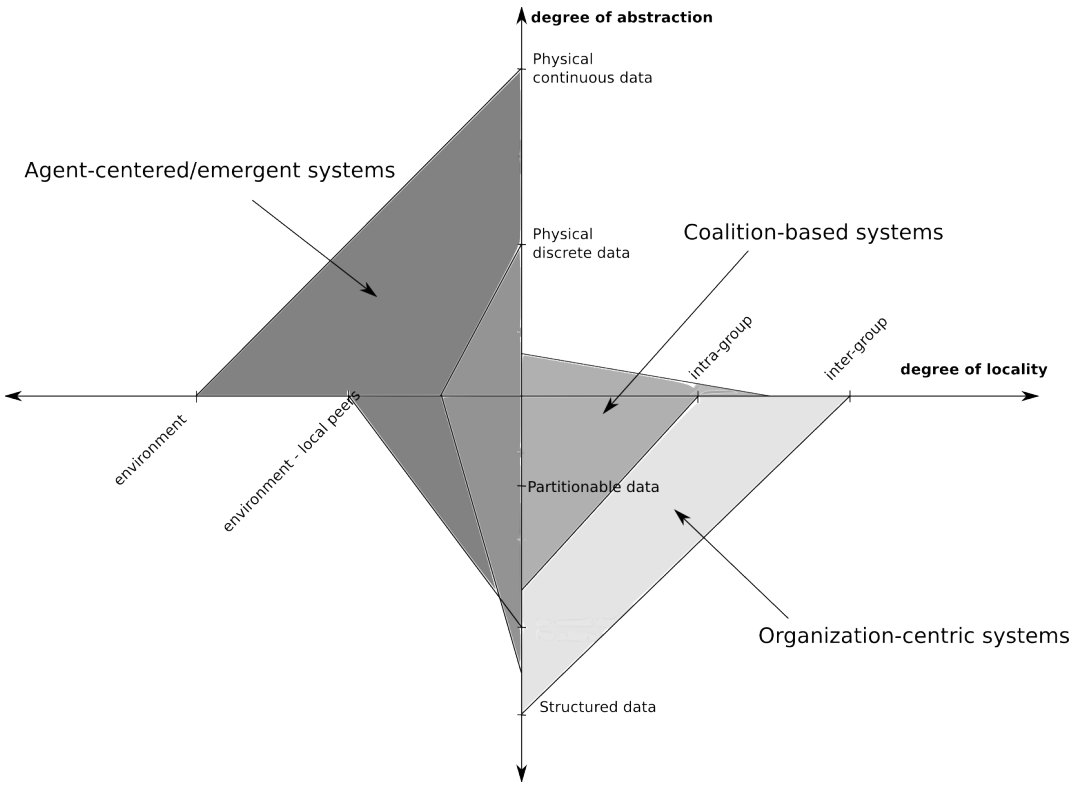


Figure 3.1: Application input analysis grid

and spatial distribution. Such properties make for fluctuating output as well, and therefore a fast, self-adapting MAS is required to best keep up with the expected output (estimation or clustering). Providing such an output with a suitable level of performance requires meta-level organizational representations to be kept at a minimum, to allow for a much simpler, faster manipulation of data at the agent level.

CMAS revolve around forming groups to alleviate resource contention for the MAS tasks (DSN Tracking [38], Task Distribution [37]) or to come up with a temporary strategy / decision that benefits the whole coalition (Electronic Marketplace [10]). When the constrained tasks dynamics or the intended strategy changes the CMAS need only adjust its dependence and membership network to conform to the new objectives. As such, the agents need only reason about these notions. Applications that look to establish long-termed strategies or decisions are generally modeled as complex organizations. When the objective decisions or strategies change, it usually implies multiple factors (structured data - many subtypes). As a consequence, any re-organization procedure should take into consideration issues regarding the change of the specific agent groups (together with individual intra-group roles) that are in charge of the factors concerned.

3.1.3 Analysis of Adaptation with respect to Application Dynamics

This direction is quite useful at making distinctions between possible MAS modeling approaches because it models a key part of application scenarios, namely the dynamics of the application (what interactions, dependencies, authority relations exist between the agents in the system), generally how the MAS sees to it that the target output is achieved given the input data.

The comparison is made based on the attributes identified in section Section 2.2.2.

The *degree of abstraction* axis is interpreted as the notions that the agent is able to reason about. It also shows *the way* in which these notions are reasoned about, how the agents control their actions, being that they have a more *reactive or proactive* behavior. Here, the key point of understanding is

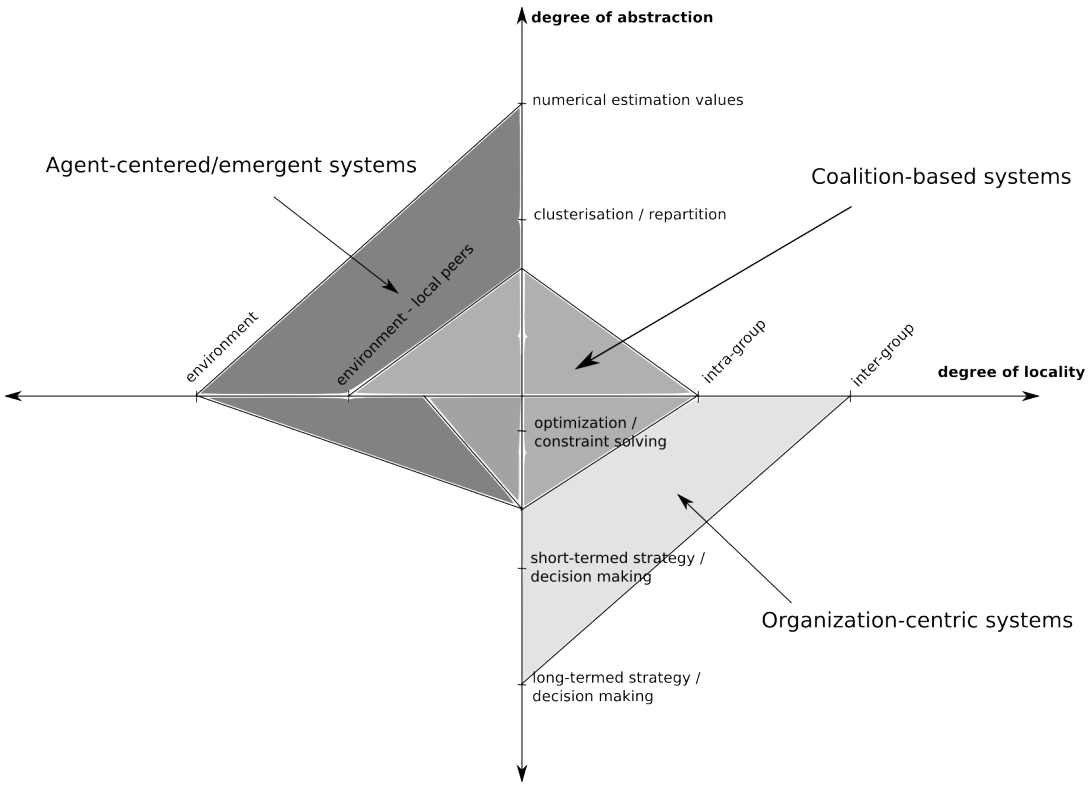


Figure 3.2: Application output analysis grid

that the applications naturally lend themselves to one or the other control type. We have seen that for applications where adaptivity and fast response to change are required (such as the "*Localization and Tracking using self-organization*" [17] or the "*Self-organization for Flood Forecasting*" [18] solutions) a simple reactive control model and the reasoning about the own actions or simple neighborhood linkage is enough to achieve good results for the problems being tackled. Re-organization for adaptation then happens from within the system as a result of using the same simple behavior but with different values for the behavior parameters or different choice of peer interaction.

Scenarios and problem statements requiring solving of numerous, resource demanding, paritionable tasks need the establishment of directed dependence (possibly even authority) relations between the group of (possibly heterogeneous) agents deployed to handle the scenario. In such cases it is reasonable to expect that agents have ways to model these peer-to-peer relations and to build a simple model (trust or cooperation) of the behavior of those they interact with. If this is the kind of model they will be using for interaction, then re-organization in CMAS must mainly be concerned about changes in these structures, without additional notions that might burden the process.

Lastly, in applications showing clear separation of task types and multi-layered hierarchical chain of dependencies (like the *Robot Soccer team [20]* or the *Dutch Crisis management [4]* problem statements) we can see from the descriptions given in the earlier sections why it is natural for agents that are members of such organizations to have ways to be aware of the structure of the group they are part of, who it is they must respond to and what the current goal of the organization is. They employ a proactive control mechanism to ensure that they can help in achieving the stated organization goal to which they have signed on to. Doing so means that trying to adapt the organization will naturally imply proactive reasoning about changes to the group roles and what they are in charge of (either on a structural or functional level).

We're moving now to the comparison along the *degree of locality* axis, which is a characterization of the *entities involved* in an interaction and of the *predominant direction* of those interactions.

Let us first better describe what the terms *"flat", "univocity"* and *"unilateral"* refer to. Flatness refers in this context to a one-layer, peer-to-peer interaction pattern. That is, any two agents can potentially interact with each other and there is no relevant vertical (hierarchical) communication.

The interpretation of the terms *"univocity"* and *"unilateral"* is taken from the work "Structural Aspects of the Evaluation of Agent Organizations" [19]. Intuitively, *"univocity"* has to do with the level of conflict and redundancies of a given structure. The higher the *univocity* criterion, the more unambiguous is the chain of commands in the organizational structure. *"Unilaterality"* has to do with the level of subordination present in a structure. The higher this attribute, the lower the amount of peer-to-peer information exchange is within the organization.

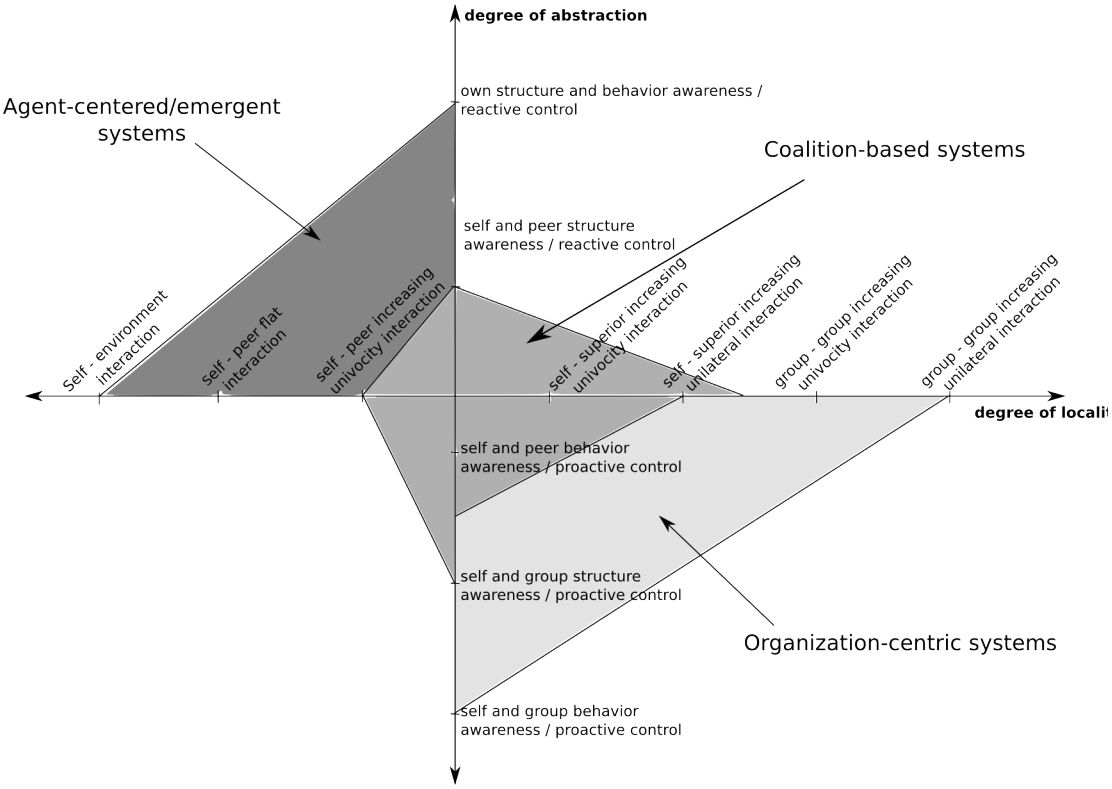


Figure 3.3: Application process analysis grid

Thus, going again through the analysis of the application scenarios that have been reviewed, on the *degree of locality axis* we can notice that problems modeled with self-organizing emergent systems tend to have a stigmergetic or direct flat communication pattern involving larger amounts of similar type agents, who differ only in the content of the information they exchange, not in the way they exchange it. Therefore, in order to provide a flexible way of change for these interactions one only needs to specify simple means for the agent to interact with the environment or with its neighbors.

Applications that were modeled as coalition-based systems tended to have small univocity self-to-peer or self-to-superior interactions. The small univocity comes from the large dependence network that is usually built in such scenarios and it actually brings the need for the formation of coalitions between agents, so that they can share resources and cooperate to solve the tasks they are assigned (Distribution of tasks, DSN for real-time tracking) or to obtain certain advantages as a group (electronic markets). When all of the system's interactions occur between peers or on a single-layer hierarchy (coalition leader - coalition member), re-organization procedures will do enough if they involve modifications of the interlocutors of said interactions (members join and leave a coalition, form one of their own, just changing who they talk to or have to obey to).

Lastly, problems that can best be handled by establishing a real organization centric view of the system, show higher univocity and unilateral communication patterns. The predominant (important) direction of interaction is on the vertical (up or down the hierarchy) as each group of agents of the organization is responsible for certain goals and has to report to managers higher up in the chain of command. Such systems make clear not only how information exchange happens within the group, but more importantly, how one group has access to required information from another group (e.g. how mayors coordinate the ROT, how defenders pass the ball to mid-fielders, how clients ask for products from producers in e-commerce applications). If the focus falls on the vertical direction (inter-group)

of communication instead of the horizontal (intra-group) it will be very likely that a re-organization attempt will have to keep track of how this information flow is changed. This means for example, what the connections of a new role introduced in the system will be (structural changes), and what kind of information should be exchanged between the newly created links (obligations in context).

We have so far made a detailed comparison of the 3 MAS adaptation categories (self-organizing systems, coalition-based systems and organization-centric systems) along the lines of the application input, output and process analysis grids. The figures at the end of each comparison section show that the mapping of scenario characteristics to appropriate MAS reorganization models forms a continuum, which was an expected result. The following sections will now take a more closer look at the room booking application. We are going to present the scenario from an organizational point of view and, using our defined grids, we will find the most suitable adaptation model. The generics of that adaptation approach and its realization in the room booking application will be detalied in Section 3.3 and Section 3.4

3.2 Room Booking Scenario Modeling

In this section we detail the workings of the room booking application and present the MAS organization that is used to manage the functionality. By doing so we will show that this particular application is best described using an organization-centric view. This in turn has implications on the way in which adaptation is to be provided and later sections will dwell on this fact, presenting a general model of adaptation in OCMAS and then tailoring it for our scenario.

The room booking application has been described briefly in Section 1.2 where we gave a presentation of the application objectives (managing the rooms and the event schedule).
Here we are going to detail the way in which the objectives are achieved and show what the different triggers for reorganization are, giving possible scenarios for each case.
We elaborate on these aspects by first presenting the agentification of the system which is used to apply the governance strategy expressed through the organizational specifications described later in this section. These specifications will be given in line with the MOISE+ [20] model, along the structural (notions like roles, groups, links), functional (goal schemes and plans) and normative dimensions (obligations and permissions that link roles to social goals).

The Multi-Agent System used to govern the room booking application performs the following:

- use *room agents* to manage the rooms they are in charge of and the events that are scheduled within them by:
  - making an inventory of the number of seats in the room and the existent equipment
  - turning on utilities (light, heating etc) when an event is about to start or when people enter the room
  - turning utilities off when if the event has ended or no users are present in the room for a long time
  - automatically registering users when they participate to an event
  - monitoring the *number of requests* they receive and the *energy* the room consumes
- use a schedule and notification service managed by a *scheduler agent* to accept and handle event schedule, deletion, modification or inquiry actions
- use a *monitor agent* to inspect the performance of the current organization of rooms in terms of succesful responses to user requests

As mentioned in the introduction, we want to exhibit the need for adaptation in the above described environment. There are 3 factors that can urge a change in the system: growing request trends, energy considerations and room inventory alterations.

Initially, all rooms are assigned one of the three categories (teaching, meeting, brainstorm) according to their characteristics. It can then happen that, for instance during a conference week, there will be daily increases in the number of requests for meeting events which may not all be scheduled due to the

lack of appropriate rooms. In this case, the system has to automatically sense this growing request trend and reorganize so as to assign additional rooms to host meeting events. It has to do this in a way that does not cancel any events of other types (teaching or brainstorm sessions).

If during some periods of time (like in the summer months) some thresholds are set for the building energy consumption and then these thresholds are surpassed, the system has to reorganize again (by switching roles for example between compatible rooms) to keep energy consumption within bounds.

Lastly, if a room is due for renovation or some equipment is missing, causing the room to not meet the requirements for the category it is assigned to, the MAS needs again to react to this by setting another location to fill in for the unavailable one (if there are many requests for events that are to be held in the room that just went offline).

Having defined the functionality of our application and outlined the structure of the multi-agent system in charge of the its management, we will now make use of organizational notions to show how the agents are assigned to roles, what missions they are responsible for and what the interactions between them are.
We start by listing the roles that form the organizational structure of the building management system. These roles and their responsibilities have been determined on the basis of an iterative analysis process which considered the flow of requests and the necessary interactions on which their handling depended. We present the name of the group they are part of as well as the responsibilities and permissions that are attributed to each role:

- `building_manager` - is part of the `building_group` and is responsible for setting up the required *workspaces* as well as for the assignment of roles to agents of the system (organization instantiation)
- `scheduler` - is part of the `building_group` and is in charge of handling the event schedule, modification or deletion requests that user agents send.
- `monitor` - is part of the `building_group` and is responsible for keeping a global, wide time window view of the number of requests for each event type (teaching events, meeting events or brainstorm events) as well as of the consumption of energy for each room in the building. This information can be used for possible reorganization triggers.
- `room_manager` - is part of the `room_manager_group` which is a subgroup of the `building_group`. The role is further specialized into `teaching_room_manager`, `meeting_room_manager` and `brainstorm_room_manager`. Each specialized role is in charge of managing the appropriate rooms and the events that are scheduled in it. This entails the actions of managing the utilities (light, heating etc) at event start and end, managing the equipment in the room, managing user registration for an event.

Additionally, there are special groups created for each room, resulting into several `teaching_groups`, `meeting_groups` and `brainstorm_groups` which are all members of the `building_group`. These groups provide a context for the events taking place in each room and its participants. Each group contains the required room_manager role and the user agents that attend the event can register as either `presenter` or `participant`.

Figure 3.4 gives a MOISE [20] like visual representation of the above described structure. It shows the groups and their member roles that make up the structure of the organization for the room booking application. One important thing to notice is that the composition links for both roles and subgroups have minimum and maximum cardinalities attached to them (depicted as min .. max). Thus, for example, there can be only one agent playing the role of *scheduler* (1..1) but there must be between *NrMR* and *NrRooms* agents playing the `meeting_room_manager` role within the `room_manager_group`. We will see in later sections in this chapter that the model for adaptation in the room booking scenario involves structural changes that alter the minimum cardinality of roles and subgroups.

Having shown how the multi-agent system organization is structured, we now present the missions and goals each agent is in charge of and the interactions that take place between them to achieve them.

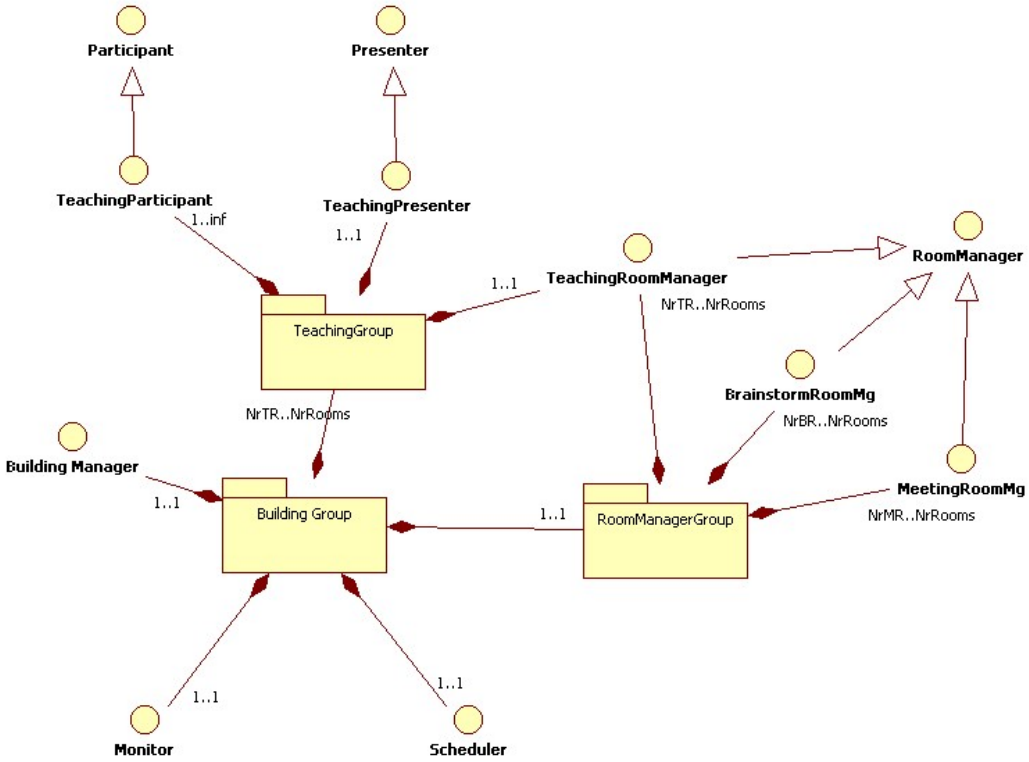


Figure 3.4: Organization structure - groups, roles and associated cardinalities

Figure 3.5 shows a dependence-graph like decomposition of the organizational goals. Each goal can be included into one or more missions. The missions themselves are then assigned to the appropriate roles. Looking at the goal names we can observe that, from a conceptual point of view, they can be regarded as a sort of *maintenance goals*, implying that their achievement is equal to the action of continuously maintaining a desired system state. Thus, the scheme in Figure 3.5 says that, in order to achieve the management of the building, agents have to manage the rooms and the schedule and monitor the building. In turn, managing the schedule means that agents have to respond to schedule, modification and deletion requests. All of these goals have to be handled simultaneously, hence the use of the *parallel* operator.

As mentioned, the organizational goals are grouped into missions which are then assigned to the corresponding roles. Table 3.1 shows how this distribution is performed in the case of our application.

Table 3.1: Assignment of missions to roles

Role	relation type	Mission = {goals}
teaching_room_manager	*obligation*	*M1 = {MgTeachingRoom}*
meeting_room_manager	*obligation*	*M2 = {MgMeetingRoom}*
brainstorm_room_manager	*obligation*	*M3 = {MgBrainstormRoom}*
scheduler	*obligation*	*M4 = {MgScheduleRequest, MgModifRequest, MgDeleteRequest}*
monitor	*obligation*	*M5 = {MonitorRequests, MonitorEnergy }*

In order to achieve the goals in their assigned missions, agents of the system also have to communicate with each other to exchange required information. In what follows, we are going to present the chain of interaction that starts from the issue of a user request to the notification of it being scheduled or not and where. When a request comes in, it is first registered in a request pool and then the scheduler is notified of the new addition. The scheduler retrieves the request and looks at the type of event requested - teaching event, meeting event or brainstorm event. It forms the list of appropriate candidate rooms and then starts to look, in order, at each room's availability for hosting the new

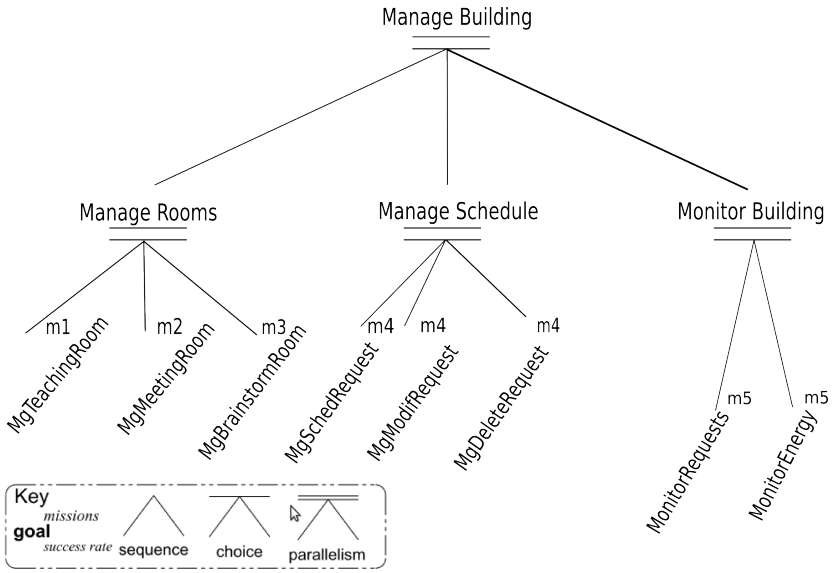


Figure 3.5: Functional specification of the organization - list of subgoals and missions

event. If a room is available, the scheduler asks the corresponding *room_manager* if he also meets all the request requirements in terms of room format, number of seats and equipment. While checking for this compliance, the room manager can potentially book missing equipment from the central equipment repository. If the room complies with all demands, the scheduler informs the user agent of the successful handling of its request, specifying the room in which the event is set to take place. After awaiting confirmation from the user agent that it has received the above notification the scheduling process is complete. If no rooms have free slots or are compliant with the request demands a failure notification will be issued by the scheduler.

While the scheduler handles new requests, room managers tend to their rooms and the events set to be held in them. They go through an operational loop that has the following steps:

- check for new event - if detected set as current event
- event starts - if the event is about to start, turn on utilities and signal energy consumption estimate for the current event
- event running - if the event is running
  - if no users in the room for a given period of time turn utilities off
  - if users enter room and utilities off, turn them back on
- event ending - if the event is about to end and no more users are in the room, set the event as finished, turn utilities off and release any booked equipment

All the room managers keep a separate schedule of events and count the requests they receive and the energy they have used during the current day. These values will help them to trigger short-term (current day) reorganization changes, as we will detail in section 3.4

With all the structural and functional aspects in place we can now turn our attention to the mapping of our application to the analysis grids defined in section 2.2.2. As by now known, we will conduct the analysis from the 3 identified view points: application input, output and processing.

From the ***application input*** point of view, the room booking scenario deals with **structured data with few subtypes, but many instances** (the degree of abstraction axis). The types of events and their requirements as well as the number of rooms and their attributes are limited in number, but many instantiations of these types exist within the one day, week or month. These kinds of data are acquired in a **self-superior inter-group fashion** (degree of locality axis). The scheduler agent gets notified of pending requests directly by the corresponding service, while room managers have their scheduled filled in by the scheduler resulting in an inter-group communication (between an agent from the *building_group* and agents from the specialized *room_manager_group* subgroup). On a

similar note, it is the room managers that monitor their individual current day schedules and send reorganization requests to the building manager.

Looking at the analysis done from the ***application output*** point of view, the room booking scenario is concerned with establishing a **long-termed strategy** (degree of abstraction axis) for scheduling and handling events and for dynamically assigning required roles to the appropriate rooms (degree of abstraction axis). The obtained strategy needs and affects the work of the entire organization, as one reorganization decision involves changes within different subgroups (**inter-group**) and is imposed by an agent from one group (building manager) on agents from a subgroup (room managers), resulting in a **self-superior** (degree of locality axis) spread of the newly considered structure.

Lastly, from the ***application process*** viewpoint, agents within our scenario exhibit an **awareness of their own structure and that of the groups that form the organization** (degree of abstraction axis), using this knowledge in a **proactive way** to control their actions. The building manager and the scheduler always know how many room managers of each type there are in the the *room_manager_group* subgroup. Likewise, room managers of a given type know how many of their colleagues play the same role and are acquainted with the structure of the main building group from which they receive the scheduled events. Both during normal functioning and even more so during reorganization, agents use this knowledge proactively to try and come up with the best possible reallocation of roles. The way in which agents of the system interact to handle a request (presented in an earlier paragraph) can be labeled as **group-to-group interaction with high univocity** (degree of locality axis). Because all agents are assigned a role, they know exactly who it is they have to interact with at any moment (both during normal run as well as during adaptation) leading to clearly defined paths of communication. This is where the increased univocity stems from.

Making the above mapping and using the results shown in figures 3.1, 3.2 and 3.3 we can observe how the room booking application clearly falls within the category of OCMAS models for adaptation. Having established this fact, in the next sections we are going to see how such a general model looks like and how it is tailored for our scenario.

3.3 Model of adaptation in Organization Centered View

We have previously identified the fact that our scenario is well suited to organization-centric techniques for both normal run as well as adaptation. Therefore, in this section we are going to look at a general process that allows us to achieve this within the OCMAS model.

Organization centric multi-agent systems feature the highest level of abstraction with regard to organizational notions. In this case, the "cooperation scheme" the agents use to guide their behavior is materialized in the form of explicit concepts, which were already introduced in previous sections, those of *role, group, communication or acquaintance links, goal schemes* and *normative specifications* that bind a given *role* in a given *group* to a given *goal*.

It is necessary to note that all the above mentioned notions play an active and important role in the reasoning process carried out by an agent in that playing a role bares with it a set of actions that the agent has to assume responsibility for in order to achieve the global objective. If at any moment, agents are capable of noticing that the current "cooperation scheme" is not suited to the reality of the environment that they operate in, they can trigger a process of adaptation in which they change the *specifications* of their current organization. These changes can occur at different levels of the specifications:

- the entity level (organizational entity - OE) – a change in the instantiations of the roles (which agent plays which role)
- the structural level (organization structure - OS) – a change occurs in the attributes associated with each structural concept (cardinality of roles, type of links etc)
- the functional level – the change occurs in the goal schemes that one or several roles are responsible for
- the normative level – the change occurs in the assignment of obligations or permissions of a given role to fulfill a given goal

Picard et al notice that there exists a generic process [30] that has to be followed by a multi-agent organization in order adapt by making changes at any of the above mentioned level. It consists of two major steps, *monitoring* and *reparation*. During *monitoring*, agents have to exactly determine the organization specific fault that prevents them from achieving their goal in the current environment. They have to make sure that the problem does not lie within their own inability to fulfil their obligations (at agent level), but within the specifications of the organization itself (the current structure, functionality or norms are not suited for the environment the agents are facing).

After identifying the fault, agents can pass to the *reparation phase* which is itself divided into *design*, *selection* and *implementation* stages. In the *design* stage agents have to analyze the fault and prescribe the most appropriate list of changes that will help the organization transition from its old and ineffective state to a new and adequate one. If there are more such proposals, the *selection* stage will choose the one that holds the best cost/effectiveness ratio. The *implementation* stage finishes the process by overseeing the correct and timely execution of the actions determined during the design phase.

All these steps can be seen as a generic process of reorganization and indeed, our analysis grids have the purpose of showing that this steps should be instantiated differently, using techniques that depend on the characteristics of the application. Like mentioned previously, in OCMAS the agents know and actively use the meta-level organizational notions that help guide their behavior. Therefore, the idea that we promote in this work is that, if the scenario requires reasoning about complex specifications like roles, groups or obligations, then *the act of changing these specifications will need guidance in itself.* The result of this trail of thought is that we envision the definition of organizational specifications with the explicit purpose of managing the process of adaptation, i. e. constructing an *organization for reorganization.*

Such a model was first proposed by Hubner et al. in [22] and we consider the approach to be very well suited for the task of managing an adaptation process in organization-centric systems. To present the specifications we are going to use graphical representations similar to the ones shown in the previous section where we described the details of the room booking application agent organization.

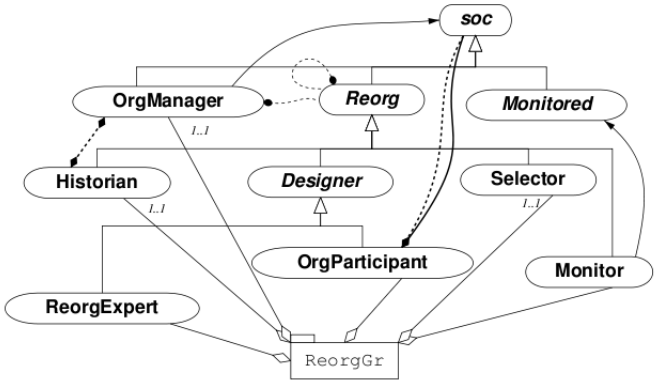
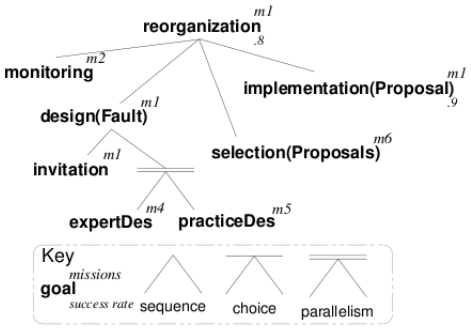


Figure 3.6: Structural specification of the reorganization group

Figure 3.6 shows the structural specification of the "organization for reorganization". We can immediately observe that it comprises one or several roles for each of the earlier mentioned stages of the reorganization process. The *OrgManager* is the role that has to instantiate the organization and supervise the whole process. This role is also compatible with the one of *Historian*, which is in charge with keeping a list of all the changes that the organization has gone through.
The entire process starts when the agent(s) playing the *monitor* signal(s) a fault with the current workings of the organization. Then, during the design stage, both *experts* and *organization participants* can come up with solutions for the identified problem. A separate role, that of the *selector* will the have to evaluate the best proposal out of those made by designers. At the end, the *OrgManager* will supervise the actual transition from the old organization to the new one.
Figure 3.7 has a representation of all the individual stages of the reorganization process and their above discussed mapping to each role.

The advantage of this organizational model for reorganization is that it makes each stage of the process



role	relation type	mission = {goals}
OrgManager	$obligation$	$m_1 = \{$*reorganisation, design, invitation, implementation*$\}$
Monitor	$obligation$	$m_2 = \{$*monitoring*$\}$
ReorgExpert	$obligation$	$m_4 = \{$*expertDes*$\}$
OrgParticipant	$permission$	$m_5 = \{$*practiceDes*$\}$
Selector	$obligation$	$m_6 = \{$*selection*$\}$

Figure 3.7: Functional and normative specification for the reorganization process

explicit and controllable. Various concrete and application specific strategies can be implemented for each step, but the core schematic remain well defined for all agents of the system, thus securing a proper guidance during adaptation, as needed in OCMAS. The way in which this generic model is instantiated in the case of the room booking application is presented in the next section.

3.4 Room Booking Scenario Adaptation Model

In the previous section we have shown that a so called "organization for reorganization" is a well suited model for governing the adaptation process within organization-centric systems. Now, we are going to see how the model is tailored to the application considered in this thesis, keeping in mind that one of the objectives is to achieve a continuous cycle of the bottom-up and top-down working perspectives, as described in section 2.1.

The bottom-up viewpoint promotes a system in which organizational changes are designed and executed by the agents within the organization, resulting in an endogeneous process. Thus, in our application, almost all agents that play a role in the room management organization described in section 3.2, also play a role in the scheme that guides the adaptation of the former. Figure 3.8 gives a visual representation of this overlap.

From the figure, one can see that all the *room managers* also act as monitoring agents. As described in section 3.2, each room agent tracks the number of events scheduled in its room and the amount of energy consumed by these during the course of one day. We will show later in this section how these values help the room managers determine faults in the organization's mode of operation and, consequently, trigger the adaptation process.

While in the building management organization the *building manager* is only used to make the initial instantiation of all the workspaces, during the reorganization phase it plays a much more active part by assuming the role of *OrgManager*. Since currently there is only one effective change that can occur in the system (a dynamic repurposing of rooms according to the most asked for event type), some small simplifications of the general model for adaptation in OCMAS have been considered for our scenario. The *OrgManager* has been made compatible with the *historian*, *designer* and *selector* roles, such that the building manager will be responsible for finding the new allocation of rooms to event types and for supervising the transition from the old organization to the new one.

To better present how the agents can change their organization in a bottom-up way, we are going to detail the algorithms used for the monitoring and design stages, at the same time highlighting another important aspect mentioned in section 2.1, the combination of reactive and proactive agent behaviors.

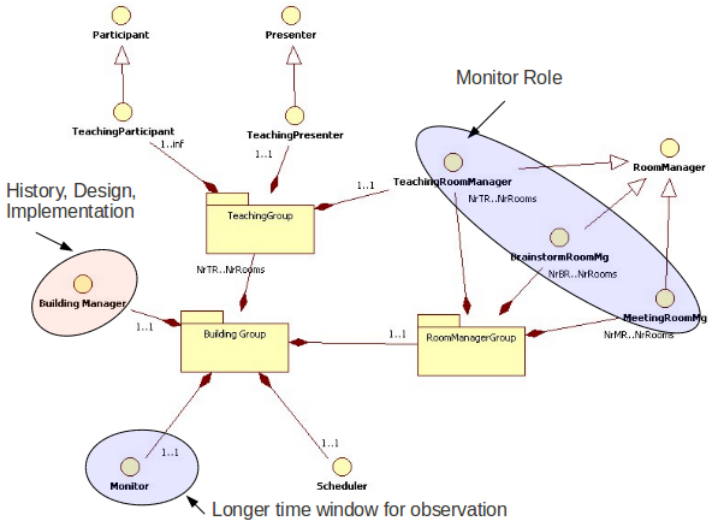


Figure 3.8: Overlap of room management and reorganization roles

In the scenario description given in the introduction (see section 1.2) we presented 3 possible triggers for adaptation, mentioning that the focus lies on the factor concerning the changing trend of event requests during one day. The monitoring activity carried out by the room agents is centered on this aspect.
Each room manager counts the number of requests he receives each hour. A simple estimation rule is then used, which says that the *number of requests expected for the next hour is equal to the amount received previously.* If this value is greater than the number of free slots left for today's schedule, a local fault occurs for that room. In the system, all the room agents that play the same role are linked to each other in a logical ring structure (receive messages from the left neighbor, send messages to the right neighbor). If a room manager notices a local fault like the one described above, it will send a token along the ring asking for the situation (ratio of expected requests to free slots) at each of the agents playing the same role (and thus hosting the same kind of events). When a room agent receives such a token, it will add information about its expected amount of requests and the number of slots it still has free for the day. When the token returns to the sender, the respective agent will check to see if the total number of expected event requests is greater than the value of free slots. If this is the case, that room manager will signal the need for reorganization, sending the values that caused this "fault" to the building manager, which plays the designer role. This entire monitorization process can be interpreted as being based on a reactive behavior of the room agents, because all the required actions (counting events, estimation, sending/receving tokens) are pre-determined, yet still lead to the desired effect, the identification of a possible request overflow.

Having received a "fault" signal, the building manager will start the *reparation phase* of the reorganization process which can be regarded as a set of proactive actions which are ment to find a redistribution of the room managing roles to room agents so as to account for the expected surplus of event types dictated by the monitors.
The building manager starts by determining how many rooms it would take to be able to host the surplus of event requests signaled in the "fault". It does this by looking at the values of total expected event type requests, total number of free slots left in the rooms handling such events and total number of hours left in the current work day. After computing the number of additional rooms required, the building manager will try to determine which agents managing event types different from the constrained one are able to change their role (e.g. from meeting room manager to brainstorm room manager). It forms a list of all the room agents which manage a room that is compatible with the general requirements of the constrained event type. It sorts the agents in this list by their number of scheduled events and then looks to see which of them have no events scheduled past the current hour, or which of them can redelegate such events to other room managers.
After finalizing this list, the building manager will construct a *reorganization plan* which consists of several commands like *leaveRole, leaveMission, changeRoleCardinality, playRole etc* representing different directives that will be sent to the agents in the list. Given the presented dynamics of our

scenario, the reorganization plan will contain the actions necessary for the "replacing" agents to leave their current managing role and adopt the constrained one, thus effectively allowing their room to host events of the constrained type.
One important feature of the reorganization commands is that they possess a *level of strictness*, which influences the way in which they are to be executed. The strictness level specifies the importance of the correct and timely execution of the associated command and can also be associated with the expected degree of cooperation of the agents within the organization. A command can therefore be labeled as:

- a regimentation - the action is to be executed immediately and is enforced at the organizational meta-level
- an obligation - the action is to be executed fast, but the targeted agent can first finish off any remaining goals from previous engagements. The changes become active at the organizational level only when the agent confirms them.
- a permission - the action may or may not be executed by the targeted agent. The new organization does not depend on the successful completion of this command.
- an interdiction - the agent must in no way execute this command during the current reorganization process.

Regimentation and *permission* can be used with more cooperative agents. In the case of regimentation the agents will have already fulfilled all their duties, so the commands can be executed immediately, while in the case of permission they are expected not to perform any negative actions. *Obligation* and *interdiction*, on the other hand, could more often be used with less cooperative agents who have to be explicitly made aware of their responsabilities during the reorganization process.

Once the design is finished, the building manager agent will assume its supervising role and oversee the implementation of the organizational transitions. For *obligation* commands, for example, he will have to wait until the targeted agent actually confirms the execution of the reorganization action.

With the transition from the old to the new, adapted organization (which can now host all future expected events), the agents start to work guided by the new organizational specifications. Consequently, we can observe that the combination of *top-down* and *bottom-up* viewpoints is materialized under the continuous cycle of *normal operation* driven by *organizational norms* and *endogeneous adaptation* which changes these norms or the structure of the organization when it no longer satisfies the required guidance for the successful achievement of the global goals.

3.5 Summary

This chapter started off with a detailed interpretation of the application analysis grids defined in section 2.2. Following an explicit, multi-agent model of the room booking scenario and the performed interpretation of the analysis grids we showed that our application falls in the category of OCMAS level adaptation requirements. We then followed with the presentation of a generic organization-centric adaptation process and its customized version for our scenario. In the latter part, we have also shown how a multi-agent system application can combine *top-down* and *bottom-up* functionality view points as well as reactive and proactive agent behaviors.
With all the models in place, the next chapter is dedicated to the overview of the technical, implementation related aspects of the room booking application.

Chapter 4

Implementation Details

This chapter brings a closer look at the technical aspects of the room booking simulation application, developed as part of this thesis (`http://jacamo.sourceforge.net/?page_id=87` is a link to our project and related ones). We will present the development platform that was employed and the way in which all the scenario and adaptation models presented in the previous chapter have been implemented on top of that platform. Also included is the description of a newly started project that has the potential to bring fast and easy development of multi-agent governance applications for ambient intelligence environments.

4.1 Development Framework

The *JaCaMo* platform, described in this section, has been devised to support and investigate in practice the idea of the integration of a rich set of abstractions concerning separate but related dimensions of multi-agent system programming: *organization*, *agent*, *interaction* and *environment*. Looking at the above definition, it immediately becomes clear why special attention has been given to this platform, given the scenario of the room booking application and its stated adaptability requirements.
A software system programmed in JaCaMo is given by a *MOISE* [23] organization of autonomous BDI agents, programmed in *Jason* [7], working in a shared, distributed and artifact-based environment programmed in *CArtAgO* [33].
Jason is a logic-based BDI-inspired language that makes it possible to program goal-oriented agents, integrating proactive, reactive and social behavior. All these features allow it to be used for the development of applications that need to exhibit some level of autonomy and flexibility, as is also the case for our scenario.
CArtAgO is a framework and infrastructure for environment programming and execution in MAS. It is based on the A & A meta-model [27] and allows for the design of environments as a dynamic set of entities called *artifacts*, collected into workspaces, which are possibly distributed among various nodes of a network. Artifacts are seen from an agent perspective as first-order resources and tools that agents can share, use and observe as they carry out their tasks. From the MAS engineering viewpoint, on the other hand, artifacts serve as the basic building blocks for the structuring and modularization of the environment.
The *MOISE* framework implements a programming model for the organizational dimension. It includes an organization modeling language, an organization management infrastructure and support for organization-based reasoning mechanisms at the agent level [18]. The framework is used to define an organization within a multi-agent system based on three independent dimensions. The *structural specification (SS)* provides the organizational structure in terms of roles, groups and relations between roles. The *functional specification (FS)* gives the organization behavior in terms of goals, missions and social schemes. Lastly, the *normative specification (NS)* defines a set of norms binding the structure (SS) to the functionality (FS). Based on modular and declarative specifications, the MOISE organization modeling language allows for the definition of dynamic, open and normative organizations.

Having briefly described each of the programming dimensions of the *JaCaMo* platform we will now present its most important strength: the synergies and conceptual mappings that have been identified

during the definition of this integrated meta-model. The connections Figure 4.1 terminating in filled or not filled squares show how the three building blocks of the *JaCaMo* framework are integrated.

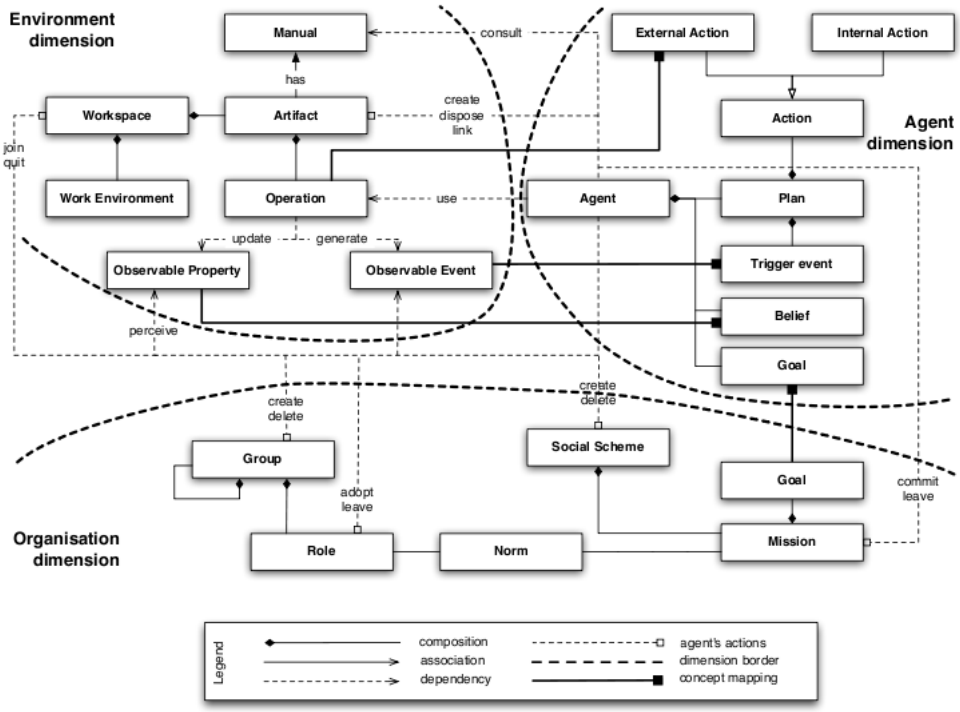


Figure 4.1: JaCaMo Programming Meta Model

The synergy between *agent* and *environment* dimensions is based on the earlier mentioned A & A (Agents and Artifacts) meta-model [27]. As detailed in [32], the interaction of agents with artifacts and workspaces are defined by rigorous semantics based on actions and percepts. Working in a workspace implies that the agent has access to the overall set of operations provided by the dynamic set of artifacts which currently exist in that workspace. On the perception side, an artifact's observable properties and the events it generates are directly mapped into agent percepts. Thus, observable properties make their way directly into the belief base of the agents observing the given artifact, while any observable event that an artifact generates is translated into a corresponding *Jason* event.

The synergy between the *organization* and *environment* dimensions is based on an organization management infrastructure for the MOISE framework which is described in detail in [21]. The basic idea consists in the uniform design of the organizational infrastructure as a part of the artifact-based agent environment. The approach works by allowing all the computational entities aimed at managing the current state of the organization (in terms of groups, social schemes and normative state) to be reified in the organization infrastructure by means of environment artifacts (e.g. every group is an instance of the *GroupBoard* artifact and every social scheme is managed by an instance of a *SchemeBoard* artifact). The agents of the system can then execute different organizational actions, such as adopting a role, committing to a mission or signaling the achievement of a social goal, by invoking the corresponding operation provided by the above mentioned artifacts. The evolution of the organizational states is made known to the agents through the events produced by the artifacts which are then mapped into agent percepts.

The *JaCaMo* platform also proposes a mapping from organizational goals to agent goals. The state of such organizational goals is computed by an organizational artifact and is then displayed as obligations for the agent. On the Jason agent side, the perception of such obligations is handled by creating a corresponding local goal.

Looking at the above descriptions, one can observe that the presented integrated meta-model is made concrete by the *JaCaMo* platform. The usefulness of the uniformity and ease of design and programming provided by this synergetic framework will become apparent in the following sections where we will use all the concepts and mechanism presented here to implement the functionality of

the room booking application and its accompanying adaptation model.

4.2 Scenario model implementation

This section is dedicated to the presentation of the way in which the *JaCaMo* platform helps implement the multi-agent model of the room booking application and the environment the agents work in. Logical work division structures, artifacts and messages used for inter-agent communication will al be detailed. We will start by showing how the work environment is modeled with the use of *workspaces* and *artifacts*, continue by detailing how organization specific entities are instantiated and end by presenting the type of messages exchanged by the agents during both normal functioning as well as when reorganizing. The following descriptions are all based on the MAS and environment models presented in section 3.2.

The environment of the room booking application comprises models of rooms in the building, handling scheduled events within them and responding to user requests. More so, all of the above actions have to be taken in a coordinated manner. Therefore, the work stage is divided into 3 types of *CArtAgO workspaces*: a *Request and Notifications workspace*, a *Room workspace for each individual room of the building* and an *Organization workspace* where all the tools for role assumption and obligation management reside (see figure 4.2).

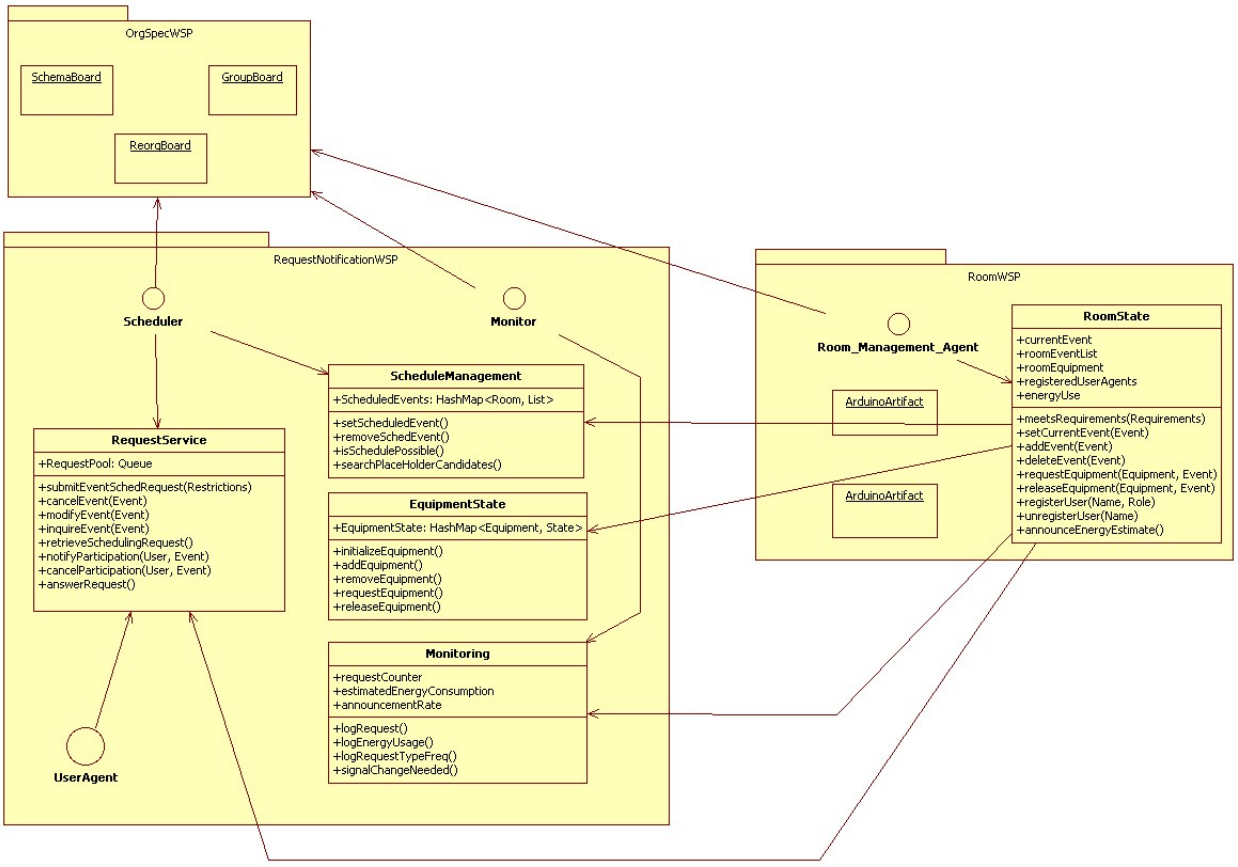


Figure 4.2: JaCaMo based system architecture of the room booking application

The *RequestNotificationWSP* workspace is where the main management agents (building manager, scheduler and monitor) carry out their work. It contains the following artifacts:

- a *RequestNotification* artifact – helps with the handling of requests. User agents can invoke operations on this artifact to schedule, modify or delete an event. They can search for already scheduled events over several criteria (title, type, start and finish hours) and signal their intent to

participate in a future happening. The artifact also allows system agents (notably the scheduler agent) to inform user agents of the success or failure of a given request.

- a *ScheduleManagement* artifact – holds the global schedule for all rooms. During normal functioning, it helps the schedule agent quickly determine if an event can be scheduled or not (*isSchedulePossible()* operation). During reorganization, it provides a method (*searchPlaceHolderCandidates()*) to find out which agents have a free enough schedule such that they can change their role to host the event types that show an increasing demand. The algorithms that use these operations have been detailed in sections 3.2 and 3.4 respectively.
- an *EquipmentState* artifact – is used as an inventory for all the equipment that can be moved from one room to another according to needs (e.g. projectors, TVs, video-conference utilities etc). For each such equipment, the artifact maintains a list of booking orders coming from the room agents that detect the need for that particular item.
- a *Monitoring* artifact which is used by system agents to store wide time window statistics about the number of requests of each type and about the energy consumption (by type and by room). These statistics can be used to try and determine medium to long range patterns of activity and change the room agent role assignments accordingly.

Apart from the *RequestNotificationWSP*, there are separate workspaces for each room agent in the building, that define the context of management actions within the room the agent is responsible for. Each *RoomWSP* contains a principal artifact called the *RoomState* artifact. This entity helps a room agent track the current event, the list of all future scheduled events, the list of all the different equipment items it senses in the premises and a list of all the user agents that have registered to the current event, as presenters or participants.
The CArtAgO framework allows artifacts to be linked to each other and the *RoomState* artifact takes advantage of this to connect different local operations to their corresponding ones offered by the other main artifacts, as shown in figure 4.2. Thus, for example, when looking to satisfy a request's requirements, a room agent might have to book additional equipment not currently present in the room (e.g. a projector). The agent then invokes an operation on its *RoomState* artifact to try and get the required technical item, which in turn re-delegates the demand to the *EquipmentState* artifact which handles such requests. At event start, the room agent executes an operation to announce the estimated amount of energy it will use for the current event. This operation also logs the value in the *Monitoring* artifact. When an event is finished or canceled, deleting it from the room agent's list also causes a corresponding removal in the *ScheduleManagement* artifact. Finally, when user agents use the *RoomState* artifact of a room to register, as presenters or participants, for the event currently taking place, a linked method is invoked that checks the user credentials kept in the *RequestNotification* artifact.
In each room worspace there may also be several *ArduinoArtifacts* (detailed in section 4.4), which help the managing room agent execute actions on the utilities (light, heating), sense (e.g. via RFID tag tracking) the equipment that is brought into or taken out of the room (movable projectors, TVs etc) and count the number of seats available.

Lastly, all the system agents join the *OrgSpecWSP* workspace, which contains the organizational artifacts that manage the current organizational state. Instances of the *GroupBoard* artifact are created for each group of roles (e.g. the *building_group* or the *room_manager_group* as described in section 3.2). These instances allow the agents to execute operations like assuming or leaving a role and generate observable properties that notify any other agents, that have a focus on that artifact instance, which agent plays which role inside the group.
For each goal scheme (an example is given in figure 3.5) there is an instance of the *SchemeBoard* artifact. A scheme is added to all the groups that contain agents participating in the goal-solving process defined by that scheme. The artifact thus allows the agents to commit to a mission or leave it when finished. Whilst in the committed state, the agents can also invoke operations that specify that a given social goal has been achieved. The observable properties displayed by the *SchemeBoard* make known the evolution of the coordinated process to satisfy the global goal.
The *OrgSpecWSP* may also contain a *ReorgBoard* artifact which is created by the building manager agent only during reorganization and its functionality will be detailed in section 4.3.

The *CArtAgO* framework offers facilities for the perception and manipulation of the environment. System agents also coordinate through the exchange of messages. Most of the interactions that exist

during the normal run of the system have already been described in section 3.2. Now we are just going to take a more technical look at the type of those messages and the way they are handled.
System agents communicate through the use of KQML performatives. This offers an advantage within the *Jason* agent language, because the inter-agent communication component is built around the KQML model. Thus, *tell, askOne or achieve* performatives appear directly as events or goals at the agent level.
The building manager sends out an *"init_complete" tell* message once it has finished instantiating all the needed workspaces and artifacts. Receiver agents perceive this as an event and trigger a plan to join their corresponding workspaces and focus on their required artifacts.
The scheduler agent and all the room manager agents begin their reasoning cycle by sending the building manager a *performingReorganization(Flag)* message of type *askOne*, demanding if any agents in the organization are performing changes to its specification. The answer to this question is given in the *Flag* parameter which gets its value directly from the belief base of the building manager.
After determining that the schedule of a room allows it to host a new requested event, the scheduler agent will send a *isRequirementsOk(EvReqStr, OkMessage)* message of type *askOne* to the room agent managing that room, asking it to verify that it meets the requirements specified by *EvReqStr*. The event generated as a consequence at the room agent level, will then fill in the value of the *OkMessage* response parameter.
The above listed messages are exchanged as part of the normal functioning of the system. In the next section we will also give an enumeration of the locutions exchanged during the monitoring and reparation phases of the reorganization process.

As concluding remarks for this section, one can notice that all the above descriptions of workspaces, artifacts and messages used in the room booking application to implement its functionality are a proof for the usefulness and appropriateness of the *JaCaMo* platform to the requirements of our scenario. From a practical point of view, it allows for a uniform and seamless access to actions that help agents to manipulate and reason about the environment (artifacts), being guided to do so in the recommended and responsible way (organizational actions and exchanged messages). Because these aspects are essential to the AmI governance model we have proposed, the use of the JaCaMo platform has proven to be of great value.

4.3 Adaptation Model implementation

Whereas the previous section was concerned with detailing the aspects of environmental architecture and agent messaging within our application, the current one will focus on the mechanism used to coordinate the monitoring, design and implementation phases of the reorganization process.

In section 3.3 we introduced our model for reorganization within organization-centric systems and emphasized the fact that such systems need a so called "organization for reorganization" to help guide the adaptation process. Section 3.4 then continued showing how the proposed model applied to the concrete scenario of the room booking application, detailing the algorithms used by agents during the key phases of the reorganization process: monitoring, design and implementation.
As mentioned earlier, this section will complete the above descriptions in chapter 3 by showing how the JaCaMo platform provides the means necessary to implement the presented models.

In the previous section, we saw that each room agent operates in its own workspace that contains the *RoomState* artifact. During the monitoring phase, this artifact provides the necessary information that the agents needs to look at. Every hour, the artifact sends out an observable event from which the ratio between expected event requests and remaining time slots (for the current day) can be computed. If the agent notices a local fault, it will send a KQML message with the *tell performative*, to its right neighbor, informing it of the *request_trend_growing* as detailed in section 3.4. The use of the JaCaMo platform allows these messages to be automatically treated as Jason events, which can then be handled at agent level. When other room agents on the same logical ring (i.e. which play the same managing role) receive the initiators message, they will make use of an operation provided by their *RoomState* artifact, called *getEventCountInfo*, which computes their own local ratio of expected requests to free slots. As the message is passed around the ring a more global picture is formed and when it comes back to the initiator it can decide whether or not to signal a need to adaptation to the building manager (like explained in section 3.4).

When such a need does occur, the building manager is responsible for the reparation phases: design and implementation of the reorganization plans. In the previous section, we mentioned the existance of the *OrgSpecWSP* which holds all the artifacts relating to organizational specifications. Whenever a reorganization is required, the building manager will create inside the *OrgSpecWSP* an artifact, the *ReorgBoard*, to help guide this process. It is used both as a mean to set up the sequence of commands needed to accustom to the changes imposed by the signaled fault, as well as the key mechanism to oversee their correct implementation.
Figure 4.3 shows a series of class diagrams related to the functioning of the *ReorgBoard* artifact.

When designing the list of changes, the building manager has 3 stages in mind: one that holds commands that tell targeted agents to stop their current work (*OEStop*), one that lists changes at the specification level (*OSChange*) and one that contains commands for the targted agents to assume the new role they have been assigned to (*OEStart*). The three stages form a stop-change-reinstantiate strategy.
Each stage can contain several *ReorgSections* which can be seen as containers of commands for which there exist no particular dependence of precedence constraints, i. e. the commands of a section can be executed in whichever order. On the other hand, precedence relations exist between the different *ReorgSections*.
As explained in section 4.2, there exists an instance of a *GroupBoard* for each organizational group specification and an instance of a *SchemeBoard* for each social goal scheme. Therefore, each section can contain several *ReorgPlans*, each of which contains reorganzation commands targeted at an above mentioned artifact instance and specifying actions that need to be undertaken by the agents that play a certain role (made known by a group artifact instance) or are committed to a certain mission (managed by a goal scheme artifact instance).

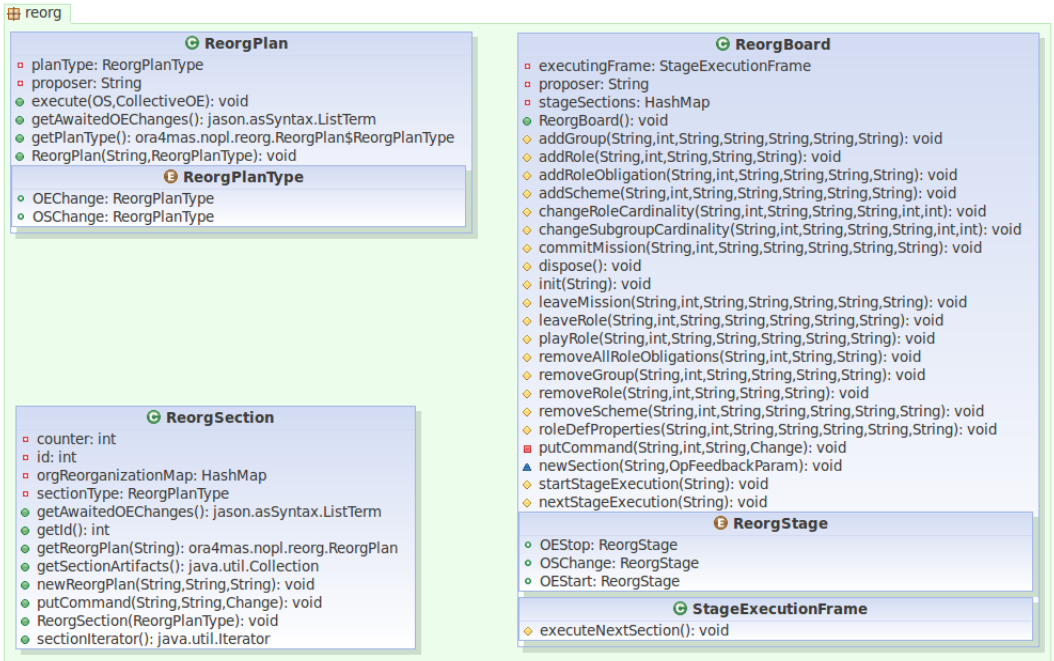


Figure 4.3: Architecture of the ReorgBoard used for the design and implementation phases of the reorganization process

The operations listed in the *ReorgBoard* class are all handles that allow the addition of a command to a reorganization plan. Methods can target changes that have to occur on the organizational entity level (e.g. textitleaveMission, *leaveRole*, *playRole*) or at the structural level itself (e.g. *changeRoleCardinality*, *changeSubgroupCardinality*). These operations also permit the setting of the *strictness level*(explained in section 3.4) of the implied change obligation.

Figure 4.4 defines the structure of the reorganization commands. On can notice that OE (organizational entity, i.e. instance of the organizational specification) level commands contain a field which

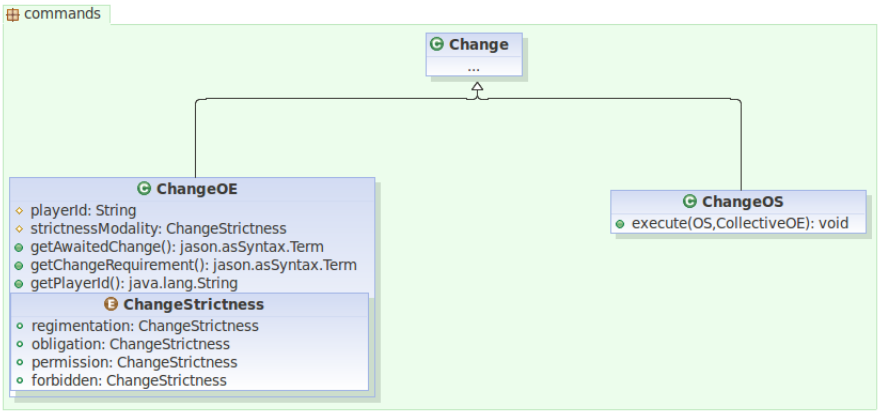


Figure 4.4: Class diagram for the reorganization commands - entity level and structural level

sets the *strictnessModality*(regimentation, obligation, permission or forbidden). In the case of the *obligation* strictness level, where agents have to conform to changes but are not forced to do so right away, the commands feature a method called *getAwaitedChange* which tells the agent playing the *OrgManager* role what modifications he needs to wait for, until the targeted agent actually complies, in order to continue with the next actions.

During the implementation phase it is this interaction of the *OrgManager* with the *ReorgBoard* artifact that insures the proper and ordered execution of the reorganization actions. Earlier we mentioned the existence of three stages for the implementation process: stop current workings, apply structural changes if needed, instantiate modified organization. The *OrgManager* used the *startStageExecution* operation provided by the *ReorgBoard* to start a stage. Remember from previous descriptions that a stage can contain several sections. The *ReorgBoard* keeps an internal field which stores the current section that needs to be executed. The artifact uses linked operations to send the reorganization commands contained in a section to the corresponding organizational artifact instance (a *GroupBoard* or *SchemeBoard*). When executing a section the *OrgManager* is informed of all the modifications he needs to wait for (as per number of commands with an *obligation* strictness modality). After all changes have been carried out by the targeted agents the, *OrgManager* can proceed to running the next section. When all sections in a stage have executed, he is again informed via an observable event sent out by the *ReorgBoard* that it is possible to proceed to the next stage.

The overview of implementation details for the model of adaptability considered in the room booking application is again a proof for the usability of the *JaCaMo* platform. It is notable that the guidance of the reorganization process implies a simultaneous work spanning all three dimensions (agent, environment and organization) and the technical specifications mentioned in this section show how the *JaCaMo* platform provides an easy manipulation of all the intertwined dimensions.
In the last section of this chapter we shall see how the framework can be extended to explicitly accomodate an ambient intelligence environment which uses many sensors and actuators to pull and push contextual information.

4.4 JaCa-Arduino - an AmI proxy

The stated aim of the thesis is to provide an agile governance of ambient intelligence environments. Though the room booking application is in itself a simulation that shows how an organization-centric multi-agent system can successfully manage the functionality and adaptability requirements of our scenario, the actual ambient part of the application has not been neglected.
As part of the thesis, a project called *JaCa-Arduino* has been started, which aims at developing prototype applications that allow for easy access and control of an ambient intelligence environment (through different sensors and actuators) from the agent side.
Arduino boards [1] are simple micro-controllers which can integrate a multitude of sensors and actuators. The numerous libraries and applications that have been developed by an active open source

community, make Arduino controllers an ideal candidate for the role of a facilitator between high-level agent programming and low-level sensor and actuator workings.
Since in the JaCaMo platform the CArtAgO framework is responsible for the programming of the *environment dimension*, the *JaCa-Arduino* project is an attempt to provide a means for the CArtAgO platform to work with Arduino based applications. By this integration we aim to allow the creation of artifacts that directly control (read or set values) the different sensors and actuators that can be attached to an Arduino board.

Functioning principal and configuration

The idea behind the project is simple and elegant. A tiny web server implementation is run on the Arduino that accepts GET and POST requests. Handlers are then written for each type of request, depending on the URL, using GET requests to retrieve values from a sensor and and POST requests to set values for an actuator. In this way it is easy to write a CArtAgO artifact that manipulates the reads and writes of values from and to the board.
An important feature is the fact that this wrapper artifact is designed as a generic component that handles all the defined set and retrieval operations. The end users are only required to write the following types of files:

- an XML file that describes the sensors you have on the board and the operations that you can perform on them
- a PDE file which will be loaded on to the Arduino and which implements handlers for the different operations (requests) you have (how to read/write values to sensors/actuators connected to the board)

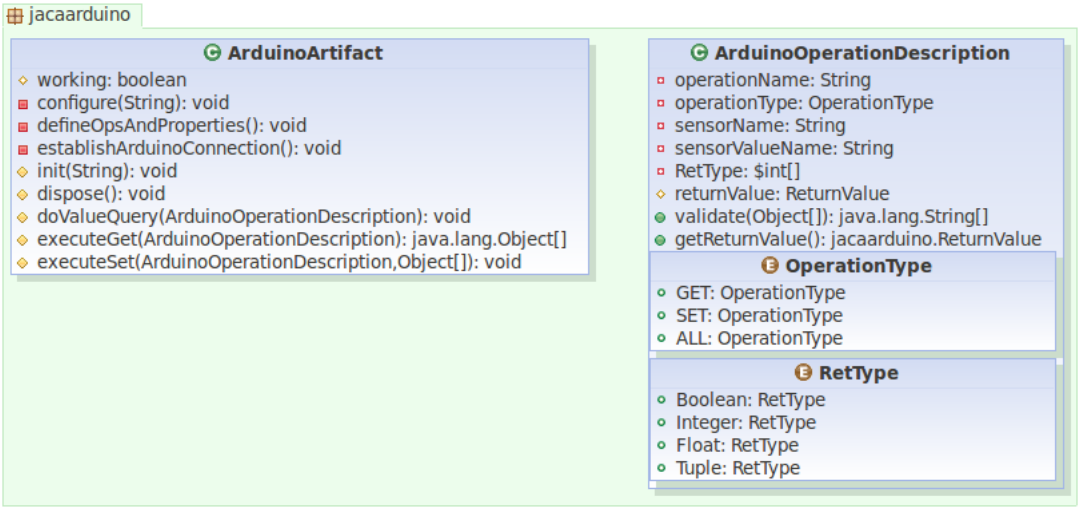


Figure 4.5: Class Diagram for the main components of the ArduinoArtifact

Figure 4.5 presents two important classes that make up the architecture of the Arduino artifact. The *ArduinoArtifact* class represents the generic wrapping artifact that is configured by means of the data given in the XML file passed as a parameter at initialization. The *defineOpsAndProperties* method is invoked to read the XML file and dynamically create the specified operations and observable properties. The dynamically added operations then work by delegating calls to the two generic getter and setter methods (*executeGet* and *executeSet*), to which they must pass a description of the target directive to be invoked on the Arduino board. Such a directive, modeled by the *ArduinoOperationDescription* class, contains details about the operation name, its type (GET or SET), the targeted sensor and sensor value. In the case of a GET operation, the class also facilitates the validation and retrieval of the data returned by the Arduino controller.

As part of the thesis, a proof of concept application was developed, which sees a Jason agent control

a simple LED, turning it on and off over a fixed time period (detailed in [2]). Though the project is still in its infancy, we believe that it has a good potential to become an important tool for the easy construction of prototype applications that aim at providing agent-side control capabilities over ambient intelligence environments.
In the room booking application, for instance, it could be used as the mean by which room manager agents control utilities (turn lights and heating on and off) or by which they account for the equipment present in their room (using an active RFID reader to sense all the RFID-tagged equipment).

4.5 Summary

This chapter has given technical insights into the work done as part of the thesis. We have presented the *JaCaMo* platform and shown the usefullness of its promoted integrated approach to multi-agent system application development. We then moved on to point out how the platform was used to implement the multi-agent management structure of the room booking application and further, how it helped construct the reorganization guidance mechanism. Finally, we have presented a new project (JaCa-Arduino) which holds promise to help a lot in the development of agent controlled ambient intelligence applications.
As mentioned previously, the room booking application is currently at the state of a simulation, so the next chapter will give details about the simulation architecture and configurable parameters and about the obtained results.

Chapter 5

Tests and Results

In previous chapters we mentioned that the room booking application was built as a simulation focusing on the request variance conditions which trigger a need for dynamic room management role reorganization.
In this chapter we present the workings of the simulation, describing modifiable parameters and implementation details, as well as show a sample run and explaining the observed dynamics.

5.1 Simulation description

This section is dedicated to the description of the simulation environment for the room booking application. It will start by presenting the important simulation parameters and end by showing implementation details that show how the simulation runtime is controlled by means of the JaCaMo platform.

The simulation environment is specified by directives stored in an XML configuration file, an example of which can be found in appendix C.1. To understand how the application runtime is controlled by the simulation we will take a look at the modifiable parameters and their meaning:

- the tag <org-agents> lists all the organization's agents (the ones that do the managing and monitoring of the building). Each <agent> has a name and a number (which shows how many such agents there are in the organization).
- the tag <user-agents> is a container for agents that propose and participate to events. The tag <proposers> holds the proposer agents. The number of "user_agent" must be high enough such that it can cover the amount of possible concurrent events (i. e. the total number of room_-agents) The tag <participant> holds the participants. Since these are used only to simulate the "register to event" actions, their number is not relevant.
- the <sim-config> tag holds the important simulation configurations.
  - <day-duration> sets the number of days (in simulated time) that the application should run. In the current version one hour is equal to 10 seconds (1h = 10s).
  - <room-management> gives the initial distribution of room_agents to their roles. The default values are: 4 teaching room managers, 4 meeting room managers, 1 brainstorm room manager
  - <request-management> tag sets the total number of requests of each type (teachingEvent, meetingEvent, brainstormEvent) and their distribution on an intra-day and inter-day basis. An example explanation will be given next:

```
1 <teachingEvent number="6" distIntraDay="uniform" distInterDay="
    uniform" />
```

There are a total of 6 events to be scheduled within the number of days specified by "<day-duration>". The inter-day distribution is uniform meaning that there will be *6/num_days* teaching requests made each day. The intra-day distribution is also uniform. Request are made in the simulation from 8 to 13 inclusive (6 hours). It means that the number of request within a day will be uniformly distributed in the interval 8-13.

```
1 <meetingEvent number="76" distIntraDay="arithmeticDecrease"
    distInterDay="arithmeticIncrease" />
```

There are a total of 76 requests to be scheduled within the number of days specified by "<day-duration>" (2 days default simualtion value). The inter-day distribution is set to "*arithmeticIncrease*". This means that the number of requests per day is going to be given by an increasing arithmetic progression the sum of elements of which will be equal to 76. The intra-day distribution is set to "*arithmeticDecrease*". This says that the number of requests within each day will be given by a decreasing arithmetic progression with 6 elements, the sum of which will be equal to the number of events that need to be scheduled that day. In this case, it practilcally means there will be more requests made in the morning than close to noon.

The above mentioned parameters govern the simulation environment of the room booking application. Of particular influence are the intra- and inter-day distribution parameters. They directly relate to the way request variations are generated for each event type and, consequently, provide the required trigger for a desired reorganization behavior.
In addition to the mentioned runtime parameters, the simulation also sets the reference for time units. In simulated time, an hour is equal to 10 actual seconds and all differences in minutes or hours are computed in accordance with this conversion rate. At every moment, the simulation environment allows the agents of the room booking application to know the current simulated day, hour and minute units.

Looking at technical details of the component implementing the simulation functionalities, we again turn to the usage of the JaCaMo platform. The entity managing the simulation is an instance of the *SimulationArtifact* presented in Figure 5.1.

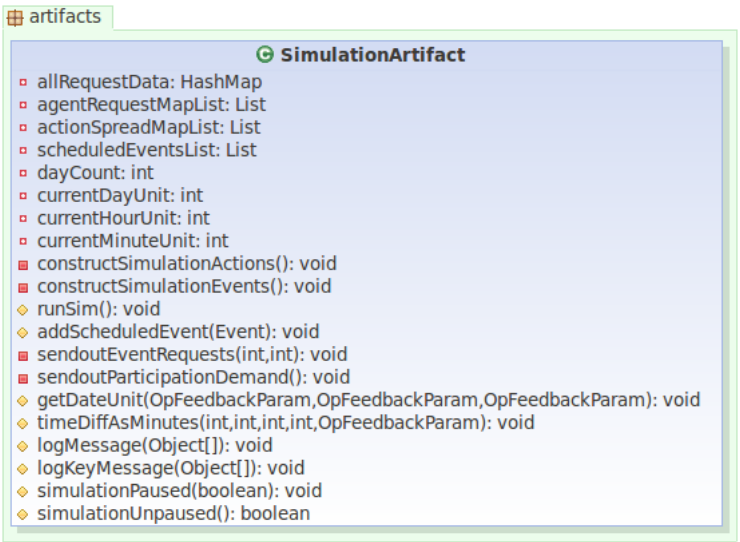


Figure 5.1: Class Diagram for simulation artifact

In the diagram we can observe that the artifact holds fields for all requests that have to be made (*allRequestData*) which are then mapped onto the individual request making agents (*agentRequestMapList*). The *actionSpreadMapList* member holds counters for each event type, that specify the amount of requests that have to be sent out every hour of every day. Building this list implies taking the intra-

and inter-day distribution parameters into account. The setup of these structures is done when the *constructSimulationActions()* and *constructSimulationEvents()* methods are called at initialization.

After initialization, the *runSim()* is called and the simulation starts. An internal operation of the artifact is used to run the operative cycle of the simulation. Inside the loop the first action is to update simulation time. Then, the *sendoutEventRequests* method is used to send signals to the proposer agents that have to make an event schedule request at the current moment. Proposer agents automatically listen for such signals coming from the artifact and once they receive them, they join the *RequestNotificationWSP* workspace of the room booking application and use the *RequestService* artifact to submit their event scheduling request. Upon confirmation of successful processing, the proposer agent uses the *addScheduledEvent()* operation of the simulation artifact to insert the newly scheduled event to the list of handled requests (*scheduledEventsList*). In the main loop, this list will be processed by the *sendoutParticipationDemand()* method in order to issue signals for participant agents to attend the event.
The simulation ends once the number of days specified in the configuration file have passed.
During all this time, agents of the room booking application can invoke the *getDateUnit()* operation to obtain the current simulated day, hour and minute units. Additionally, the artifact helps create a grafic interface which displays charts showing the total number of requests and number of denied scheduling attempts as well as the current distribution of room manager roles (how many teaching, meeting and brainstorm rooms). A window that displays messages logged by user and room agents is also included.

This section has presented the way the room booking simulation works and how its functionality is implemented. Next we are going to take a sample configuration and comment on the observed dynamics by interpreting the output of the earlier mentioned charts.

5.2 Simulation analysis

In this section we are going to analyze a sample simulation run (shown through a succession of graphical chart outputs) and comment on the observed changes in the system.
As explained in the previous section, the default distribution of room management roles leads initially to 4 teaching rooms, 4 meeting rooms and 1 room for brainstorming sessions. In the sample run, we are going to use a 2-day total period. In the first day there will be a surplus of brainstorm session requests (relative to the number of such events that can be hosted in the one room dedicated to them). The second day will see a decrease in brainstorm events and a strong increase in requests for meetings. Teaching events (courses) will be kept uniform and low throughout both days. The actual values parameters are given below:

```
1 <teachingEvent number="6" distIntraDay="uniform" distInterDay="uniform"/>
2 <meetingEvent number="76" distIntraDay="arithmeticDecrease" distInterDay=
    "arithmeticIncrease"/>
3 <brainstormEvent number="34" distIntraDay="arithmeticIncrease"
    distInterDay="arithmeticDecrease"/>
```

This form of inverse situations (increase of brainstorm sessions in the first day and increase of meetings in the second) as well as the stronger compatibility between meeting rooms and brainstorm rooms (from the physical layout point of view) tends to predict a dynamic shift of agent roles from meeting room managers to brainstorm room managers and vice versa.
Though such a scenario is not often likely to happen in the real world, it was useful to develop it in order show the functionality of the modeled adaptation process and test the implementation aspects that make it possible.

Figure 5.2 shows the number of issued and denied requests (the corresponding brainstorm events could not be scheduled because of the lack of appropriate rooms) as well as the initial default distribution of room management agents. One can observe that up to 10 o'clock there have already been 3 denied scheduling requests. This growing trend continues, and as a consequence the agent playing the *brainstorm_room_manager* role notices this and signals a reorganization need. The building manager analyzes the fault and looks for suitable agents (from a schedule and room compatibility perspective)

to change their role and become aditional brainstorm room managers. Figures 5.4 and 5.5 show what the outcome of this analysis is.

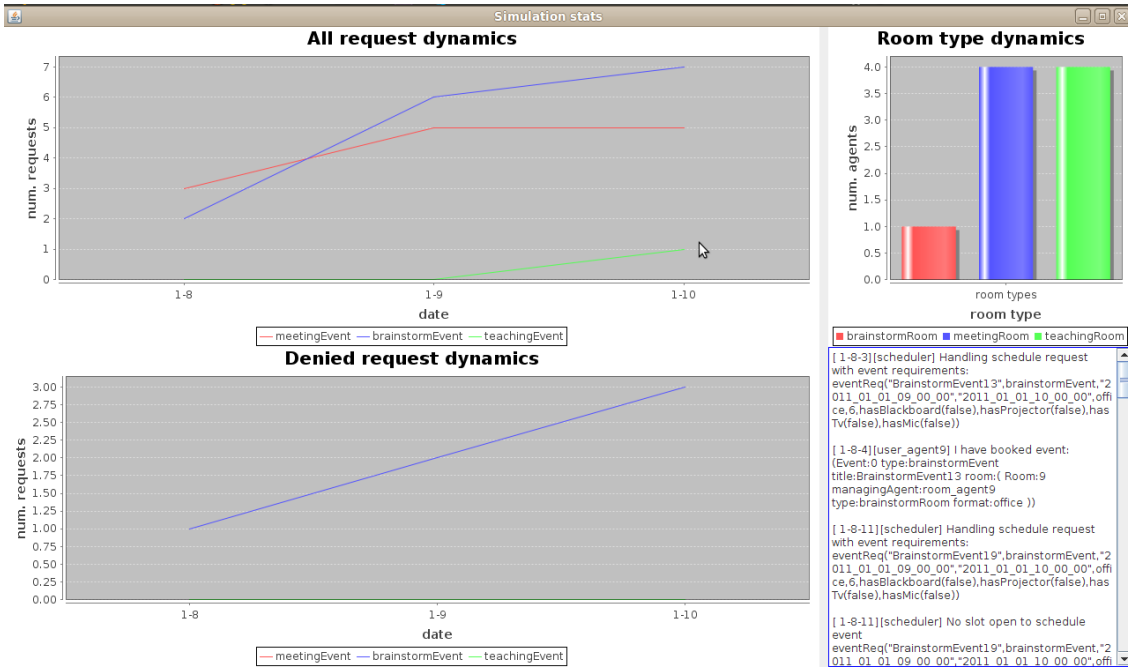


Figure 5.2: First day request dynamics and initial room management role distribution

As it was expected, the number of brainstorm room managers has increased by one by means of the role change carried out by *room_agent5* who was previously in charge of managing meeting events. The message logs shown in figures 5.4 and 5.5 also point out the 3 stages of the reorganization implementation process discussed in section 4.3. After this change, no more requests are denied, because there now are enough rooms that can handle the expected number of brainstorm events.

Figure 5.3: *room_agent5* found suitable to change his role from meeting (a.) to brainstorm room manager (b.)

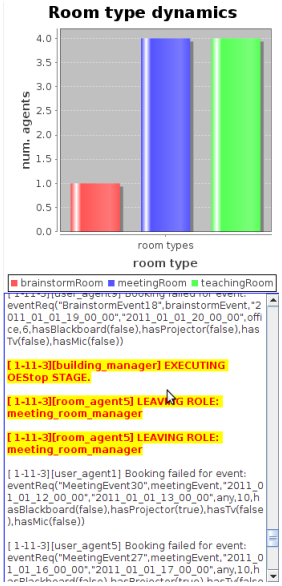


Figure 5.4: a.

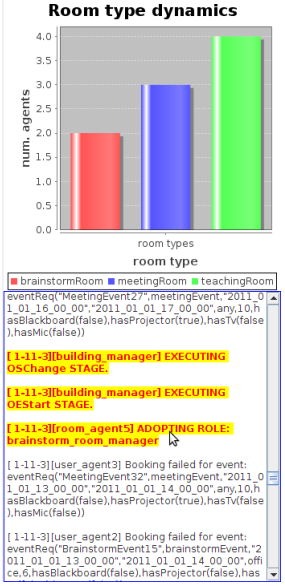


Figure 5.5: b.

Day 2 of the simulation brings the turn-around, with a strong increase of demand for meetings. Requests for brainstorm sessions and teaching events are lower. The awaited outcome is that uncon-

strained agents managing brainstorm or teaching rooms will change their role to accustom the growing number of expected meetings. Figure 5.6 confirms this hypothesis. At the start of day 2 there are 3 meeting room managers in the building. Since events can be scheduled only in the interval 9-19 (10 hours), by 12 o'clock the meeting room agents have already surpassed their total capacity to host events (43 requested events, where only 30 are possible). Therefore, at 12 o'clock a reorganization process is triggered and the building manager identifies room agents 3 and 9 (managing a teaching room and a brainstorm room respectively) as suitable helpers to host the additional expected meeting events. The logged messages displayed in the figure show the actions undertaken by each agent during the adaptation process. After the change, one can observe that no more requests continue to be denied since there are now sufficient rooms with free slots for meeting events.

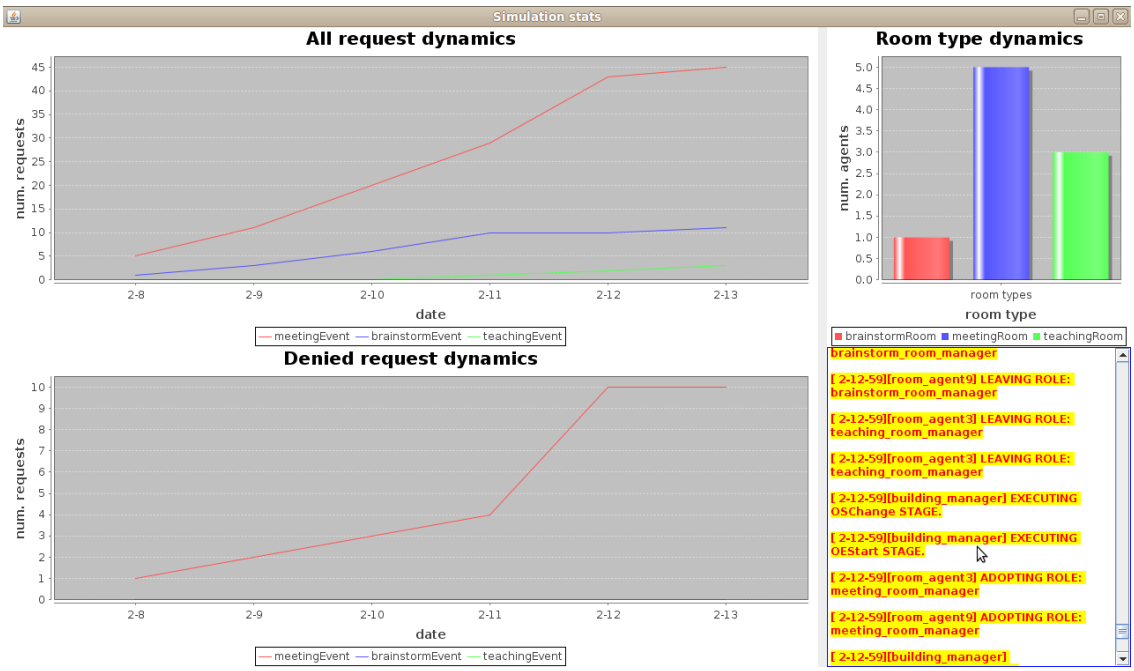


Figure 5.6: Second day request dynamics and final room management role distribution

The overall simulation might not be complex, but the modeled scenario is good for demonstrating and observing the behavior of the considered adaptation process and its implied implementation aspects. From the sample run presented, as well as several others that were carried out, it becomes clear that the application behaves exactly as expected and that the JaCaMo platform proved to be a sound and solid support for the programming of a multi-agent system charged with the management and adaptation requirements encountered in the room booking application.

The results and the analysis carried out here have shown the correct behavior of the room booking application in the face of its required functionality. The next and last chapter of this thesis is dedicated to the report of conclusions and the potential theoretical and practical perspectives of this work.

Chapter 6

Conclusions and Perspectives

This final chapter presents the conclusions of this thesis and perspective future work. Both sections (conclusions and perspectives) will discuss ideas along the lines of the 3 important topics that have been handled in this work: the application adaptation analysis grid, the OCMAS reorganization model and the MOISE and Arduino based implementations.

6.1 Conclusions

Conclusions on Application Adapation Analysis Grid

Section Section 3.1 has shown a detailed analysis of attributes specific to the input / output data of an application and to their processing that make a given problem naturally approachable from an agent-centered self-organizing view, a coalition-formation view or an organization-centric view regarding both normal system operation, as well as the accompanying adaptation component (seen as the change of the agent organization). The foundation on which we base our analysis is the belief that there exists a mapping between characteristics of an application scenario and the most appropriate way of achieving adaptation within that scenario. By "way", we understand the totality of notions and organizational levels of abstraction that the agents of a MAS have to be aware of and change. We consider the *type* of these notions (structural, behavioral, normative), the way in which they are reasoned about (reactive, proactive) and the *locality* and *direction* of interactions in which such notions are discussed about or exchanged. Our belief is that, since all of the above criteria are in a sense particular to a given view of MAS organization modeling (from the 3 types we have mentioned), the mapping implicitly extends to relations between application characteristics and multi-agent system techniques of providing adaptation.

As it happens with every classification attempt, the one proposed here must not be considered as being strict. The dimensions of the application grid being continuous, it is possible to envision approaches that are crossing two categories. Still, the detailed review of the presented application scenarios show that distinctions (though not definite) can be made between the above described mappings.

Conclusions on the model of adaptation

In section 3.2 it was established that, considering the proposed analysis grid, the room booking scenario falls into the category of OCMAS based adaptation models. Section 3.3 then presented a general model of adaptation in such organization centric systems. The general stages of a reorganization process were discussed and the utility of a so called "organization for reorganization" was pointed out. The key aspects to be remembered about the proposed adaptation model were discussed in section 3.4 where we showed how the general model maps onto the room booking application system. It was established that the roles of the "*organization for reorganization*" can be played by endogeneous agents and that the required adaptation functionality involves both bottom-up (distributed monitoring) as well as

top-down (solution design and implementation) processes. This mixture proves to be very important in relation to scaling and autonomy capabilities.

Conclusions on the JaCaMo platform and JaCa-Arduino

An important topic of this thesis was the implementation of the proposed ambient intelligence governance and reorganization model. Though the final result of this work is an incipient simulation module, some very important features of the underlying JaCaMo development platform were put into perspective and an interesting, potential holding project (JaCa-Arduino) was started, which could lead to an easy way of prototyping MAS governed AmI applications.
The most important aspect of the JaCaMo framework that can be identified from the descriptions of its use given in sections 4.2 and 4.3 is its completeness. The platform allows for seamlessness and ease of programming in all the 3 important dimensions that have to be modeled in a complex scenario like that of the room booking application: the *environment*, the *agents* and their *organization*.
Equally relevant for the purpose of this work was the unfolding of the JaCa-Arduino project, presented in section 4.4, the continued development of which might in the end result in a fast and integrated (thanks to the JaCaMo platform) scheme for the programming of multi-agent system governed ambient intelligence applications.

Remarks on self-organization

In this final section we also want to give some additional remarks on the self-organization approach and present some future directions of investigation.
Going through the characteristics of the applications we have studied we notice that when it comes to self-organization there are two different interpretations that one can give to this aspect. The first one is that of the driving force that creates the organization itself. From this point of view, the main difference that establishes itself is between systems with imposed organization (Organization centric system view) and those with emergent ones (emergent organizations - in the sense of an observed specialization of behaviors and activities like in cooperative mechanisms such as AMAS or reinforcement mechanisms). This introduces the need to make a closer analysis of distinguishing attributes on the agent behavior level. Whereas in emergent organization agents are engineered to have a simple, reactive behavior, organization-based systems impose predefined roles and functionalities that allow for a proper guiding of the system behavior towards the desired direction. Agents have to pro-actively reason whether they can fulfill the requirements of the role they want to play inside the organization and, possibly, whether joining the organization brings them any advantage.

The second interpretation that one can give to self-organization is that of adaptation done from within the system, without any external help. It is a view that wants to study self-organization without the emergence component. In such a case both the strong and weak definitions of self-organization [24] could apply in a scenario of imposed organizational structures by acting on the structural level. Agents would reason about the structure of their organization more than on its behavior. Thus, the difference would come from the kind and complexity of notions that the agents work with (from simple environment or peer-to-peer links in agent-centered systems, to group membership and inter-group directed dependence relations in organization-centered systems). The main focus of the self-organization component would then become the evaluation of appropriate algorithms that allow the agents to change these structural components as required by the change occurring in the environment.

6.2 Perspectives

Perspectives on theoretical analysis

There exist concerns that have not been addressed in the analysis on reorganization done in this thesis. Specifically, the issue of how to build up an organization that has explicit notions of groups, roles, obligations and required patterns of activity and communication starting out from a flat layer of agents. That is to say that agents self-organize to create the very organization that will later help them to keep things under control when trying to achieve a goal. Furthermore, they do this in a non-emergent way because they are aware of the structure that they are creating since it is them that are working out the details of the organization to begin with. The problem then shifts to finding out how agents can be endowed with the ability to recognize what is missing from their organizational state (if they have one already) that will better help them achieve a commonly decided (or preexistent) global goal, and then use this knowledge to *create* the necessary (additional) roles, groups and specific activity patterns the instantiation of which will ensure the correct achievement of the global goal. A simple way to put it is: how do agents interact with each other to come up with an abstraction such as the specification of an organization.

Another direction of investigation would be along the lines of the future work intentions expressed in [6]. As more and more application studies and implementations will be carried out, it would be interesting to refine the distinguishing attributes already mentioned in this work even further and possibly try to provide measurements for some of them.

Perspectives on the application reorganization model

The main line of improvements to the reorganization model of the room booking application comes from exploring different complex adaptation strategies and different context sources for triggering/helping the reorganization process its AmI setting. Examples of ideas that can be considered for investigation are the addition of user preferences in the request handling process (e.g. preferences for rooms) and the extension of the agent monitoring routine with the ability to detect patterns of events forming over longer periods of time so as to minimize the amount of times that a reorganization is performed by anticipating such a requirement. The latter idea would of course need the establishment of a suitable machine learning procedure.

Perspectives on the JaCaMo platform and sensor management using JaCa-Arduino

The room booking application is one of the first major use cases that benefit from the full expressiveness of the JaCaMo platform. The actual implementation can still be improved by making the simulation truly decentralized (currently it runs on a single machine). A stable, decentralized version can then be used to make various performance and profiling tests that could eventually lead to advancements in the JaCaMo development platform itself.
A last but obvious line of future work is the continued advance of the JaCa-Arduino project notably with a better error handling mechanism and debugging options as well as the creation of multiple real life sensor management applications which validate the usefulness of the framework.

Appendix A

Grid Analysis - Application Overview

A.1 Analysis of self-organization MAS applications

1. **Self organization of User Communities using Middle Agents**

   The application **[39]** is meant to support information exchange between user communities by providing each user with an agent representing their interests. The application also features middle agents which act as brokers for different user agent queries. The role of the middle agents is to form groups of user agents with similar interests around them in the attempt to make the system scalable and provide a timely and efficient response to user queries.

- – Inputs
  - ∗ user interests - user model
  - ∗ structured information
- – Output
  - ∗ list of user with similar interest
  - ∗ user community based on similarity
  - ∗ basically a division of agents based on possibility of interaction
  - ∗ a clustering of user agents - cluster centers are common middle agents
  - ∗ Purpose
    - · lower system query overload
    - · logical structure adaptation of a network
- – Process
  - ∗ find user with similar interests
  - ∗ use a middle agent as broker for info requests - they perform search
  - ∗ user agents carry and seek to acquire info for their users
    - · contact a middle agent
    - · issue all queries to middle agent
  - ∗ middle agents can talk between themselves to satisfy a search request
  - ∗ mark for successful or unsuccessful interaction between user agents
  - ∗ distribution (transfer) of user agents between middle agents based on above marks

  - Characteristics
    - almost flat structure - user agents and middle agents
- focus of agents is not on notions of role or obligations
- global goal: cluster similar user to same middle agents to reduce message overload and increase search performance
- focus on improving characteristics of the network scalability (not what the network does) in the meanwhile satisfying individual component requests.
- clustering based on similarity - the notion that agents reason about and try to improve
- predominant horizontal interaction between agent types - application model sees interaction between agents of the same kind

2. **Self-organisation through Evolving Agent Populations**

This application in **[26]**is similar to the previous one in the sense that it tries to form groups of users that share similar interests. Here, the user interests are represented as an environment. User agents stay in their own environment and send out scout agents to search for users sitting in environments of similar user interests. When they return home, scout agents are destroyed but their "genome" is used in an evolutionary algorithm to evolve the capabilities of future scout agents to find relevant environments and better interact with other scout agents.

- - Inputs
    - user interests - information retrieval - structured information
    - user agents
    - scout agents
    - interaction context
  - Outputs
    - clustering of agents into similar environments based on interests and common contexts
  - Process
    - user agents spawn scout agents
    - scout agents are modeled using an evolutionary model
    - similarity of scout agents with user agent environment or other scout agents based on this model
    - evolutionary algorithm uses agent's succes rate at finding required info as criteria for selection
    - with time scout agents converge on useful environments
  - Characteristics
    - flat structure - user agents, scout agents
    - focus is on notion of similarity between components - something that is not global to the system - user centric
    - focus on network scalability, structure - not so much on what the network does
    - predominant horizontal interaction between agent types - application model sees interaction between same kind agents

3. **Self-organisation for the School Timetabling Problem**

The problem **[29]** is a classical example of a constraint satisfaction problem (CSP). A timetable for a certain duration has to be found while respecting the explicit constraints (availability, specialization, equipment needed etc) of different stakeholders: teachers, student groups, classrooms. In the system representative agents (RA) hold stakeholder constraints and launch several cooperative booking agents (BA) that are able to interact and construct the schedule in an emergent manner.

- – Inputs
  - ∗ timetabling problem for a certain duration
  - ∗ constraints: availability, specialization, equipment needed
  - ∗ stakeholders: teachers, students, rooms
  - ∗ 3D grid of cells representing slot for a given day, hour, and room
  - ∗ large discrete domain
- – Outputs
  - ∗ timetable
  - ∗ tuples of form (teacher, student group, room)
- – Process
  - ∗ self-org engine is based on AMAS cooperation theory
  - ∗ BAs explore the cells trying to find matching slots and matching partners
    - · e.g. teacher BA matches available room
    - · partners with a studentgroup BA
    - · detection of NCS (non-coop situations)
    - · incompetence - two teacher BAs meet => BA1 will change its location to look for a suitable partner
    - · after a NCS encounter between two BAs, ba1 will store the location and BAs it knows via BA2 for future exchanges
    - · in a coop. situation the BA books the cell in which it is in and partners with another BA
- – Characteristics
  - ∗ flat structure - Representative Agents, BookingAgents
  - ∗ behaves well in situations where constraints / stakeholders are added / removed - dynamic and open environment
  - ∗ adding supernumerary agents helps finding a solution and gives better results - can be explained by the fact that the added agents can disrupt others which are satisfied with a solution that could be optimised => improvement by agitation, stiring with increased number of searching agents => greater search space exploration
  - ∗ weakness - agents not aware - of global goal - constructing a timetable - of solution optimality => need for external observation
  - ∗ usual (reactive) behavior of BAs allows them to respond to perturbations - changes in stakeholder's constraints – NCS
  - ∗ constraint satisfaction based on:
    - · splitting into individual concerns

- cooperative interaction
- small number of possible interactions (what BAs can do) - the result of which has local implications from the individual concern side, but global implications in the long run (external observer)
- medium-large number of concerns (parameters) - handled by small to medium number of actions
- non-linear activity process - either governed by uncertainty or by unforseeable function
- predominant horizontal interaction between agent types - application model sees interaction between same kind agents

4. **Self-organisation for Flood Forecasting**

The application in **[18]** sees the creation of real-time simulator uses an adaptive model for flood forecasting, which is composed of two levels of self-organizing multi-agent systems. The environment is made of the sensors of the Garonne river basin.

- – Inputs
  - complex dynamic problem
  - parmeters: hygrometry, declivity, surface, nature and permeability of the ground, rain heights, stations topologies etc

    continuous inputs
- – Outputs
  - station forecasts adapted to environmental conditions
  - forecast of a value based on a model that is hard to express mathematically
- – Process
  - two level system operation
  - specialized agents for each sensor type
  - 2nd level agents used to aggregate data from sensor agents and compute water level variation during a unitary period (1h)
  - 2nd level agents use a weighted sum of agents in the lower level
  - behavioral model based on AMAS
    - adjustment of the weights decided from coop. between agents
    - according to the input data, the results coming from other agents and the error made on the forecast
- – Characteristics
  - large number of heterogeneous parameters that affect desired outcome
  - specialization and role assignment exists but agent logics is only concerned with cooperative adjustment of links and link-weights with other agents
  - role assignment is set at design-time, role assignment does not change
  - no behavioral change, but a cooperative structural adaptation
  - self-organization without emergence
  - real-time learning of a model - based on coop weight adjustment
  - predominant horizontal interaction between agent types - application model sees

* interaction between agents of the same kind
* important: no typology of links is created - it is just link association and link weight that is changed

5. **Self-organization for Land Use Allocation**

Based on a real-world problem **[13]**, the application applies a self-organizing approach to simulate the assignment of land use categories in a farming territory. The problem exhibits a function to optimize, while respecting a set of constraints, both local (compatibility of grounds and land-use categories) and global (ratio of production between land-use categories).

- – Inputs
  * large amount of discrete parameters
  * types of ground, land use categories, ratio of production between land-use categories
  * environment - available zones of farming territory, each zone is featured by its surface, its distance to the village, the kind of soil
  * agents - gathered into groups, each being associated to a land-use category
  - – Outputs
    * instance of a quadratic assignment problem
    * assignments of type: terrain parcel -- usage
  - – Process
    * A group has a goal to satisfy by conquering spatial zones in the environment while respecting some constraints
    * behavioral model is based on a few principles inspired by the eco-problem solving approach
    * agent can conquer a zone in the environment and then contribute to the satisfaction of its group
    * search for the more attractive zone

      if zone free => occupy it, otherwise => fight, winner depends on respective strengths of their groups
    * strength of a group decreases while its satisfaction increases
  - – Characteristics
    * optimization problem subject to given constraints
    * problem statement is: division of environment to groups of agents based on attractiveness parameter
    * results comparable to those obtained by simulated annealing => progressive local impact encounters
    * self-adaptation by solving local encounter problems based on local-neighborhood strength

6. **Localization and Tracking using Self-organization**

The localization task **[17]** can be defined as finding the position of an object (or more than one), mobile or not, in a well defined referential location. The tracking problem is to provide a

succession of positions that are spatially and temporally coherent. The application proposes a reactive model to tackle this issue.

- – Inputs
  - ∗ well defined referential location
  - ∗ environment: square grid, each state represents a target's possible position; is featured by an altitude representing the possibility of presence of a target at this position
  - ∗ continuous values
- – Outputs
  - ∗ finding the position of an object (or more than one), mobile or not, in a well defined referential location.

    provide a succession of positions that are spatially and temporally coherent
- – Process
  - ∗ reactive model
  - ∗ Agents are equivalent to weighted particles evolving in an environment of force field
  - ∗ behavioral model inspired by a model of flocking; expressed through a formulation taken from Newtonian physics - environment heavily dictates agent behavior
  - ∗ Agents are attracted by position according to their altitude and are mutually repulse each other
  - ∗ Focusing is an emergent phenomenon, and is the solution of the problem: a group corresponds to the detection of a target.
- – Characteristics
  - ∗ reactive model
  - ∗ flat structure - a single reactive behavior applied to each agent
  - ∗ interaction at a horizontal level
  - ∗ environment influence on agents is greater than agent influence on environment
  - ∗ no explicit structural interaction defined - links, weights on links
  - ∗ interaction occurs as a consequence of reactive predefined behavior

7. **Self-organization for Adaptive Meshing in Cellular Radio Networks**

In radio mobile networks **[34]**, a distinguishing feature is the rapid increase of consumer demand. Responding to this demand requires the space to be partitioned between a large number of service units (cells). The environment is discretized in meshes which contain a number of resources according to traffic statistics. The problem solving is done by building up holons which cover a resource.

- – Inputs
  - ∗ service units (cells) - range, maximum traffic load
  - ∗ discretely partitionable environment - mesh
  - ∗ each mesh - discrete properties (size, traffic statistics - resource)
- – Outputs
  - ∗ optimal division of space (meshes) amongst managing agents

    - clustering of physical (discrete) space
    - can also be viewed as an assignment problem

  - Process
    - adaptive group formation - holonic approach
    - standalone holon - ensures coverage of his resource (traffic)
    - holons join meshes
    - Holon satisfaction value used to control acceptance / removal of member holons
    - satisfaction value depends on: geometrical limits, size of covered resources
  - Characteristics
    - structural considerations - group formation
    - main behavior of agent is reasoning about group membership
    - limited role types - Holon head, Holon part
    - almost flat structure - single level hierarchy - group formation
    - interactions on local structural level - main adaptive behavior concerns only this aspect (structure)

8. **Self-organization for Traffic Simulation**

The application in **[35]** proposes the use of holarchies for the modeling of agent environments. The concrete example is that of simulating the traffic in an industrial plant by providing different context and detail information to the vehicle agents traveling through it depending on the execution or simulation constraints. The solving process is based upon the building and re-organization of holarchies as vehicles drive through the plant and change the holon they belong to.

- Inputs
  - environment to be modeled
  - environmental building blocks
    - segments - links
    - exchange points - crossroads
    - zones containing buildings and segments
    - have variable granularity
  - Outputs
    - environmental model - progressively decomposable
    - hierarchy of elements
    - Each holon of the holarchy represents a specific context - specific place in the plant
  - Process
    - granularity of each holon dictated by execution or simulation constraints such as which features can be observed

      simulation accuracy achieved by locality of changes - holarchy re-organisation
  - Characteristics
    - environment modeling for situated MAS

    * system has explicit role allocations
        · Environmental holarchy is predefined as it represents the real plant environment. Indeed, the latter can't evolve and the physical laws we need to enforce are known a priori

        hierarchy of same type nodes - differentiation lies in granularity levels applied to the defined node properties
    * re-organisation is done at locally influenced structural level
        · membership determination
        · defined hierarchy requires no common decision of membership change
        · re-organization change does not affects only an agent and next/previous Holon

A.2 Analysis of coalition-based MAS applications

1. **Distribution of tasks**

Given a set of agents and a set of tasks which they have to satisfy, this problem **[37]** considers situations where each task should be attached to a group of agents that will perform the task. Task allocation to groups of agents is necessary when tasks cannot be performed by a single agent. Also, situations often arise where groups will perform a task in a more efficient way with respect to single agent performance.

- – Inputs
    * Large number of tasks
    * Tasks decomposable into subtasks
    * scarce resources
    * each subtasks requires usage of a different resource
    * entities possessing a collection of resources - agents (workers)
    * revenue proportional to difficulty of task
  – Outputs
    * coordination for task solving
        · in given time constraints
        · minimal usage of resources
  – Process
    * construction of dependence network for each task
    * selecting the one with highest utility
        · negotiation approaches
        · game theory approaches
        · market based approaches - trust values
  – Characteristics
    * decomposability of tasks
    * high number of different tasks requiring a fixed set of resources
    * different subtask sets are disjoint

    * dependence networks on a horizontal level - flat structure
    * at most one-level hierarchy for task achievement control

2. **Electronic marketplace**

This application**[10]** focuses on self-interested agents that trade goods and knowledge in a large-scale multi-agent systems. It uses coalitions as a means of grouping agents together to improve cooperation among them leading to individual benefits. An important idea is the extension of the concept of task oriented coalitions to long-term coalitions based on trust relationships between the agents.

- – Inputs
    * seller agents
    * buyer agents
    * goods of common interest for buyers
  – Outputs
    * purchase of goods at best price
    * ensure high rate of sales for successful producers
  – Process
    * agents form consumer-consumer and consumer-producer coalitions
    * successful interaction (purchase) between agents in a coalition result in a discount on the price
    * agents form trust models based on previous interactions
    * agents change coalitions based on computed trust values
  – Characteristics
    * clear (mutual) dependency networks are observable
      · seller <--> buyer

      single type goal objective - purchasing of goods - but many dependency related alternative ways of achieving it

3. **Distributed Sensor Networks for real-time tracking**

In this scenario**[38]** a distributed, incremental approach to self-organization through bottom-up coalition formation is presented that is applied to the distributed sensor network (DSN) of the EW Challenge Problem. The setting consists of homogeneous sensor agents distributed throughout a region. The agents are fixed and communicate using an eight-channel RF (radio frequency) system in which each can use only one channel at a time. An organization in such a domain helps facilitate the efficient assignment of tracking tasks to particular agents and limit contention on communication channels.

- – Inputs
    * homogeneous sensor agents

        - agent are fixed
        - communicate through 8-channel RF system
      - large region
      - objects entering the region
  - Outputs
    - tracked trajectory of objects within the region
  - Process
    - one-level hierarchy
      - agents are distributed into sectors
      - each zone has a sector manager
    - manager models
      - what is currently tracked in his sector
      - internal states of agents in his sector
    - manager tries to minimize contention on any communication channel
  - Characteristics
    - large number of same capabilities agents
    - single, long lasting task - modeled as different spatially distributed similar sub-tasks
    - clusterization of necessary resources into active interest region
    - at most one-level hierarchy - the clusters themselves
    - dependency network formation on top level of hierarchy
      - dependence on cluster workforce - sensor agents
    - model to evaluate utility of buying/sharing/selling resources (sensor agents)

4. **Vehicle routing domain**

The application in **[36]** focuses on the task of delivering orders to target destinations. Vehicles start out from a depot, may pick up orders from other depots, deliver them and return to their original starting point. The vehicles form coalitions to help them share delivery orders and minimize the total route length.

- Inputs
  - dispatch centers with vehicles
    - each vehicle leaves and return from own depot
    - each vehicle has a weight and distance limit
  - delivery tasks
- Outputs
  - best way to handle all orders
  - lowest route lengths
  - lowest time to delivery
- Process

    - long routes are first distributed to each vehicle (agent)

      iterative process of alleviating route lengths of vehicles by forming coalitions of delivery sharing agents

  - Characteristics
    - high number of similar type tasks
    - medium-large number of working elements with complementary working capabilities
    - cooperative dependencies can be built based on the complementarity of capabilities

A.3 Analysis of organization-centered MAS applications

1. **Robot Soccer Team**

   This application in **[20]** sees the formation of an organization that manages the strategies and goal-scoring tactics of a robot soccer team. Agents play roles such as attacker, mid-field or defender and have specific missions that tell them how to score a goal.

- Inputs
  - robot soccer players, ball
  - attackers, mid-fields, defenders, goalkeeper, coach
- Outputs
  - process of scoring a goal
  - strategy of scoring and defending
- Process
  - groups
    - team

      attack: attacker, middle, leader defense: back, goalkeeper, leader coach role
  - clear defined ways of getting the ball from the defenders to the attackers via the mid-fielders
- Characteristics
  - separation of concerns easily identifiable
    - each group has clear defined attributions
    - each member within group has similar attributions, functionality
    - clear dependency mapping and flow of interactions
    - well laid out functional specification of the playing strategy
    - adjustment of current organization requires changes at its abstract definition, not just the instantiation =>
    - impacts entire organization or at least a group
    - requires reasoning on the abstract components of the org.
    - e.g. changing from 4-4-2 formation to 4-5-1
    - changing from defensive goals to attacking goals

2. **Dutch procedures for crisis management**

A description of the way in which crisis situations are handled in Holland is given **[4]**. The authors of the work explain the different organizational constructs that form in such scenarios and the way in which they change (role additions and changes in attributions) depending on the severity of the incidents.

- – Inputs
  - ∗ accidents
    - · minor traffic accidents
    - · major traffic accidents
  - ∗ natural disasters
    - · oil spills
    - · fires
    - · earthquakes

  – Outputs
  - ∗ handling strategy
  - ∗ objective: save human lives, reduce material damage

  – Process
  - ∗ definition of GRIP levels
    - · GRIP0: Routine accidents
    - · GRIP1: Incidents
    - · GRIP2: Large scale incidents
    - · GRIP3: Disasters concerning multiple regions
    - · GRIP4: Large scale disasters
  - ∗ definition of appopriate org structure for each level
    - · e.g. GRIP3

      GBTs 'Municipal Executive Team' ROT 'Regional Operational Team' CoRT 'Command Disaster Location'
    - · appropriate and specific roles included in each group
  - ∗ clear dependency and chain of command are defined

  – Characteristics
  - ∗ clear dependency mapping and flow of interactions
  - ∗ separation of concerns
    - · each group has very specific attributions and communication assignments
  - ∗ abundant interactions within a group, small (but important) amount inter group
  - ∗ clear hierarchy - predominant vertical communication and chain of interaction

3. **Paper review process**

In this example **[15]**, the process of submitting a paper to a conference is simulated. The agent organization maps to groups like the Program committee or the submission group. It simulates the entire process from article submission to acceptance or refusal notification.

- – Inputs
  - ∗ authors, papers, reviewers
- – Outputs
  - ∗ acceptance or rejection of a submitted paper
- – Process
  - ∗ three groups
    - · program committee group

      program chair program committee member
    - · submission group

      submission receiver author
    - · evaluation group

      reviewer manager reviewer flow: author submits - submission receiver - PC chair delegates PC member for review - PC member creates evaluation group - reviewers review - review manager centralizes - result back to PC member - submission receiver send back acceptance/denial
- – Characteristics
  - ∗ clear separation of inputs and actors manipulating the inputs can be observed
  - ∗ clear dependency mapping and flow of interactions
    - · e.g. described flow - agents can not interact freely with every member of the organization - e.g. authors cannot freely interact with reviewers
  - ∗ different attributions assigned to each group - separation of concerns

4. **Travel Agency - e-commerce example**

This application **[15]** is an instance of the travel agency example in which clients try to get products from an agency considering as a brokering agent for product providers. The system identifies clear groups (client group, provider group, contract groups) and interactions within and between these groups (asking the brokers for a given service, making offers, signing a contract). The AGR (Agents/groups/roles) model is used to describe the organization in this case.

- – Inputs
  - ∗ client elements, product elements, brokering agency for product providers
  - ∗ diverse clients, diverse producers, diverse products
  - ∗ discrete, high-level concepts
  - ∗ inputs can be given a type and have several characteristic attributes
- – Outputs
  - ∗ contract between a client(buyer) and producer (seller) for the purchase of a product
  - ∗ mediation of a contract between client and producer

    - outputs can be:
        - a high level process applied to the inputs - e.g. transfer of ownership, purchase
        - a high-level entity that can be split into member parts - e.g. the construction of a building
- Process
    - interaction between three group structures
        - client group structure
        - producer group structure
        - contract group structure
    - an agent enters the client group, talks to broker and asks about a product
    - broker issues call for proposals to producers
    - proposals presented to client, client chooses winning producer
    - client and producer join the contract group as buyer and seller and finish off the transaction
- Characteristics
    - clear separation of inputs and actors manipulating the inputs can be observed
        - e. g. members of the producer group have certain responsibilities (to produce goods and respond to broker requests), while members of the client group have certain permissions (they are allowed to ask for goods) and obligations (they must channel their request through the broker)
        - abundant interactions within a group, small amount inter group - e.g. clients only talk to producers via brokers
        - separation of concerns
        - a certain buyer-seller deal is hidden from other clients and producers
        - clear dependency mapping and flow of interactions
        - client-broker-producer-broker-client; buyer-seller

Appendix B

Grid Analysis - mapping of applications to grid values

B.1 Comparison on analysis grid of application input

Table B.1: Matching of described applications to the input attribute grid

Application	**Degree of abstraction Axis**	**Degree of locality Axis**
Self organization of User Communities using Middle Agents [39]	*Structured data – few subtypes*, Many instances	*self – environment Local peers*
Self-organization through Evolving Agent Populations [26]	*Structured data – few subtypes, Many instances*	*self – environment Local peers*
Self-organization for the School Timetabling Problem [29]	*Structured data – few subtypes, Many instances*	*self – environment Local peers*
Self-organization for Flood Forecasting [18]	*High amount physical Continuous data*	*self – environment Local peers*
Self-organization for Land Use Allocation [13]	*High amount physical Discrete data*	*self – environment Local peers*
Localization and Tracking using Self-organization [17]	*Low amount physical Continuous data*	*self - environment*
Self-organization for Adaptive Meshing in Cellular Radio Networks [34]	*Partitionable data – discrete physical Attributes*	*self – environment Local peers*
Self-organization for Traffic Simulation [35]	*Partitionable data – Discrete physical Attributes*	*self – environment Local peers*
Distribution of Tasks [37]	*Partitionable data – Conceptual attributes*	*self-superior Intra-group*
Electronic marketplace [10]	*Structured data – few subtypes, Many instances*	*self-superior Intra-group*
Distributed Sensor Networks For real-time tracking [37]	*Partitionable data – discrete physical Attributes*	*self-superior Intra-group*
Vehicle routing [36]	*Structured data – few subtypes, Many instances*	*self-superior Intra-group*
Robot Soccer Team [20]	*Structured data – many subtypes, few instances*	*self-superior inter-group*
Dutch procedures for Crisis management [4]	*Structured data – many subtypes, few instances*	*self-superior Inter-group*
Paper review Process [15]	*Structured data – few subtypes, many instances*	*self-superior Inter-group*
Travel Agency [15]	*Structured data – few subtypes, Many instances*	*self-superior Inter-group*

B.2 Comparison on analysis grid of application output

Table B.2: Matching of described applications to the output attribute grid

Application	**Degree of abstraction Axis**	**Degree of locality Axis**
Self organization of User Communities using Middle Agents [39]	*Clusterization / Repartition*	*self – environment – local peers*
Self-organization through Evolving Agent Populations [26]	*Clusterization / Repartition*	*self – environment – local peers*
Self-organization for the School Timetabling Problem [29]	*Multi-part optimization / Constraint solving*	*self – environment – local peers*
Self-organization for Flood Forecasting [18]	*Numerical estimation Values*	*self – environment – local peers*
Self-organization for Land Use Allocation [13]	*Clusterization / Repartition*	*self – environment – local peers*
Localization and Tracking using Self-organization [17]	*Clusterization / Repartition*	*Self-environment*
Self-organization for Adaptive Meshing in Cellular Radio Networks [34]	*Clusterization / Repartition*	*self – environment – local peers*
Self-organization for Traffic Simulation [35]	*Clusterization / Repartition*	*self – environment – local peers*
Distribution of Tasks [37]	*Single-part optimization / Constraint solving*	*self-superior Intra-group*
Electronic marketplace [10]	*Long-term strategy / Decision making*	*self-superior Intra-group*
Distributed Sensor Networks For real-time tracking [38]	*Single-part optimization / Constraint solving*	*self-superior Intra-group*
Vehicle routing [36]	*Single-part optimization / Constraint solving*	*self-superior Intra-group*
Robot Soccer Team [20]	*Long-term strategy / Decision making*	*self-superior Inter-group*
Dutch procedures for Crisis management [4]	*Long-term strategy / Decision making*	*self-superior Inter-group*
Paper review Process [15]	*Short-term strategy / Decision making*	*self-superior Inter-group*
Travel Agency [15]	*Short-term strategy / Decision making*	*self-superior Inter-group*

B.3 Comparison on analysis grid of application dynamics (processing)

Table B.3: Matching of described applications to the process attribute grid

Application	**Degree of abstraction Axis**	**Degree of locality Axis**
Self organization of User Communities using Middle Agents [39]	*Own structural dependence awareness / Reactive control*	*Self - peer Flat interaction*
Self-organization through Evolving Agent Populations [26]	*Own structural dependence awareness / Reactive control*	*Self - peer Flat interaction*
Self-organization for the School Timetabling Problem [29]	*Own structural dependence awareness / Reactive control*	*Self - peer Flat interaction*
Self-organization for Flood Forecasting [18]	*Own behavior awareness / Reactive control*	*Self - peer Flat interaction*
Self-organization for Land Use Allocation [13]	*Own structural dependence awareness / Reactive control*	*Self - peer Flat interaction*
Localization and Tracking using Self-organization [17]	*Own behavior awareness / Reactive control*	*Self-environment Interaction*
Self-organization for Adaptive Meshing in Cellular Radio Networks [34]	*Own structural dependence awareness / Reactive control*	*Self-peer univocity Interaction*
Self-organization for Traffic Simulation [35]	*Own structural dependence Awareness / Reactive control*	*Self-peer univocity Interaction*
Distribution of Tasks [37]	*Self and peer structure Awareness / reactive control*	*Self-superior Unilateral interaction*
Electronic marketplace [10]	*Self and peer behavior Awareness / proactive control*	*Self-superior Univocity interaction*
Distributed Sensor Networks For real-time tracking [38]	*Self and peer behavior Awareness / proactive control*	*Self-peer univocity Interaction*
Vehicle routing [36]	*Self and peer behavior Awareness / proactive control*	*Self-peer univocity Interaction*
Robot Soccer Team [20]	*Self and group behavior Awareness / proactive control*	*Group-group Univocity interaction*
Dutch procedures for Crisis management [4]	*Self and group behavior Awareness / proactive control*	*Group-group Unilateral interaction*
Paper review Process [15]	*Self and group structure Awareness / proactive control*	*Self-superior unilateral interaction*
Travel Agency [15]	*Self and group structure Awareness / proactive control*	*Group-group Univocity interaction*

Appendix C

Simulation and Results

C.1 Simulation Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<simulation>
    <org-agents>
        <agent name="building_manager" number="1" />
        <agent name="scheduler" number="1" />
        <agent name="monitor" number="1" />
        <agent name="room_agent" number="9" />
    </org-agents>
    <user-agents>
        <proposers>
            <agent name="user_agent" number="10" />
        </proposers>
        <participants>
            <agent name="participant" number="3" />
        </participants>
    </user-agents>
    <sim-config>
        <!-- specify duration in days -->
        <day-duration>2</day-duration>

        <room-management>
            <!-- specify the initial existing distribution of roles -->
            <teachingEvent>4</teachingEvent>
            <meetingEvent>4</meetingEvent>
            <brainstormEvent>1</brainstormEvent>
        </room-management>

        <request-management>
            <!-- specify the desired number of events to be generated in
                the simulation -->
            <!--
            <teachingEvent number="6" distIntraDay="uniform" distInterDay
                ="uniform" />
            <meetingEvent number="90" distIntraDay="arithmeticIncrease"
                distInterDay="arithmeticIncrease" />
            <brainstormEvent number="50" distIntraDay="arithmeticIncrease
                " distInterDay="arithmeticDecrease" />
            -->

```

```
36              <teachingEvent number="6" distIntraDay="uniform" distInterDay
                    ="uniform" />
37              <meetingEvent number="76" distIntraDay="arithmeticDecrease"
                    distInterDay="arithmeticIncrease" />
38              <brainstormEvent number="34" distIntraDay="arithmeticIncrease
                    " distInterDay="arithmeticDecrease" />
39
40              <!--
41              <teachingEvent number="2" distIntraDay="uniform" distInterDay
                    ="uniform" />
42              <meetingEvent number="10" distIntraDay="arithmeticIncrease"
                    distInterDay="arithmeticIncrease" />
43              <brainstormEvent number="2" distIntraDay="arithmeticIncrease"
                     distInterDay="arithmeticDecrease" />
44              -->
45          </request-management>
46      </sim-config>
47  </simulation>
```

Listing C.1: Simulation Configuration File (ami-room-booking-sim.xml)

Bibliography

[1] *Arduino*.

[2] *JaCa-Arduino*.

[3] E. Aarts, R. Harwig, and M. Schuurmans. Ambient intelligence, the invisible future: the seamless integration of technology into everyday life, 2001.

[4] H. Aldewereld, F. Dignum, L. Penserini, and V. Dignum. Norm dynamics in adaptive organisations. In *3rd International Workshop on Normative Multiagent Systems (NorMAS 2008)*, 2008.

[5] E. Argente, V. Botti, and V. Julian. Gormas: an organizational-oriented methodological guideline for open mas. *Agent-Oriented Software Engineering X*, pages 32–47, 2011.

[6] C. Bernon, V. Chevrier, V. Hilaire, and P. Marrow. Applications of self-organising multi-agent systems: An initial framework for comparison. *INFORMATICA-LJUBLJANA-*, 30(1):73, 2006.

[7] R.H. Bordini, J.F. Hübner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. Wiley-Interscience, 2008.

[8] S. Breban and J. Vassileva. Long-term coalitions for the electronic marketplace. In *Proceedings of Canadian AI Workshop on Novel E-Commerce Applications of Agents*, pages 6–12, 2001.

[9] D. Capera, J.P. Georgé, M.P. Gleizes, and P. Glize. The amas theory for complex problem solving based on self-organizing cooperative agents. In *Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, page 383. IEEE Computer Society, 2003.

[10] M. Kirley D. Cornforth and T. Bossomaier. Agent heterogeneity and coalition formation : Investigating market-based cooperative problem solving. *Information Sciences*, 2004.

[11] S. DeLoach. Omacs: A framework for adaptive, complex systems. *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages 76–98, 2009.

[12] G. Di Marzo Serugendo, M.P. Gleizes, and A. Karageorgos. Self-organisation and emergence in mas: An overview. *Informatica*, 30(1):45–54, 2006.

[13] A. Dury, F. Le Ber, and V. Chevrier. A reactive approach for solving constraint satisfaction problems. *Intelligent Agents V: Agents Theories, Architectures, and Languages*, pages 397–411, 1999.

[14] S. Birrell-C. Dodsworth E. Zelkha, B. Epstein. From devices to "ambient intelligence", June 1998.

[15] J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: an organizational view of multi-agent systems. *Agent-Oriented Software Engineering IV*, pages 443–459, 2003.

[16] T. Finin, R. Fritzson, D. McKay, and R. McEntire. Kqml as an agent communication language. In *Proceedings of the third international conference on Information and knowledge management*, pages 456–463. ACM, 1994.

[17] F. Gechter, V. Chevrier, and F. Charpillet. Localizing and tracking targets with a reactive multi-agent system. In *Second European Workshop on Multi-Agent Systems (EUMAS'04)*, pages 255–262, 2004.

[18] J.P. Georgé, M.P. Gleizes, P. Glize, and C. Régis. Real-time simulation for flood forecast: an adaptive multi-agent system staff. In *Proceedings of the AISB'03 Symposium on Adaptive Agents and Multi-Agent Systems*, pages 7–11, 2003.

[19] D. Grossi, F. Dignum, V. Dignum, M. Dastani, and L. Royakkers. Structural aspects of the evaluation of agent organizations. *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, pages 3–18, 2007.

[20] J. Hübner, J. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. *Advances in Artificial Intelligence*, pages 439–448, 2002.

[21] J.F. Hübner, O. Boissier, R. Kitio, and A. Ricci. Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agents and Multi-Agent Systems*, 20(3):369–400, 2010.

[22] J.F. Hübner, J.S. Sichman, and O. Boissier. Using the moise+ for a cooperative framework of mas reorganisation. In *SBIA*, volume 2004, pages 506–515. Citeseer, 2004.

[23] J.F. Hubner, J.S. Sichman, and O. Boissier. Developing organised multiagent systems using the moise+ model: programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, 1(3):370–395, 2007.

[24] AgentLink First Technical Forum Group Self-Organisation in Multi-Agent Systems. Reorganisation and self-organisation in multi-agent systems. Number 16, pages 23–24, 2004.

[25] Y. Labrou, T. Finin, and Y. Peng. Agent communication languages: The current landscape. *Intelligent Systems and Their Applications, IEEE*, 14(2):45–52, 1999.

[26] J.P. Mano, C. Bourjot, G. Lopardo, and P. Glize. Bio-inspired mechanisms for artificial self-organised systems. *Informatica*, 30(1):55–62, 2006.

[27] A. Omicini, A. Ricci, and M. Viroli. Artifacts in the a&a meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456, 2008.

[28] J. Pavon, J. Gomez-Sanz, and R. Fuentes. The ingenias methodology and tools. *Agent-oriented methodologies*, pages 236–276, 2005.

[29] G. Picard, C. Bernon, and M.P. Gleizes. Etto: Emergent timetabling by cooperative self-organization. *Engineering Self-Organising Systems*, pages 31–45, 2006.

[30] G. Picard, J.F. Hübner, O. Boissier, and M.P. Gleizes. Reorganisation and self-organisation in multi-agent systems. In *1st International Workshop on Organizational Modeling*, page 66.

[31] G. Picard and M. pierre Gleizes. Chapter 8 the adelfe methodology designing adaptive cooperative multi-agent systems. *Engineering*, pages 157–176, 2004.

[32] A. Ricci, M. Piunti, and M. Viroli. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, pages 1–35, 2010.

[33] A. Ricci, M. Piunti, M. Viroli, and A. Omicini. Environment programming in cartago. *Multi-Agent Programming:*, pages 259–288, 2009.

[34] S. Rodriguez, V. Hilaire, and A. Koukam. Towards a methodological framework for holonic multi-agent systems. In *Fourth International Workshop of Engineering Societies in the Agents World*, 2003.

[35] S. Rodriguez, V. Hilaire, and A. Koukam. Holonic modeling of environments for situated multi-agent systems. *Environments for Multi-Agent Systems II*, pages 18–31, 2006.

[36] T.W. Sandholm and V.R. Lesser. Coalition formation among bounded rational agents. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 662–671, 1995.

[37] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[38] M. Sims, C.V. Goldman, and V. Lesser. Self-organization through bottom-up coalition formation. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 867–874. ACM, 2003.

[39] F. Wang. Self-organising communities formed by middle agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, pages 1333–1339. ACM, 2002.

[40] W. Weber, J.M. Rabaey, and E.H.L. Aarts. *Ambient intelligence*. Springer Verlag, 2005.

[41] M. Weiser. The computer for the twenty-first century. *Scientific American*, 1991.