Integrating Driving Forces into the Development of Adaptive Virtual Organizations

Sergio Esparcia García

Departamento de Sistemas Informáticos y Computación

Universitat Politècnica de València

A thesis submitted for the degree of

Doctor of Philosophy in the subject of Computer Science
Under the supervision of:
Dr. Estefanía Argente Villaplana and Prof. Vicente J. Botti Navarro

February 2015

PhD Thesis

Title:	Integrating Driving Forces into the Development of Adaptive Virtual Organizations
Author:	Sergio Esparcia García
Advisors:	Dr. Estefanía Argente Villaplana Prof. Vicente J. Botti Navarro
Reviewers:	Prof. Juan Luis Pavón Mestras Dr. Jomi Fred Hübner Dr. Markus Schatten
Examination Board:	
President	Prof. Juan Luis Pavón Mestras
Secretary	Dr. Miguel Rebollo Pedruelo
Member	Dr. Gauthier Picard

Date of the defense: 24 February 2015

Abstract

Organizations have become the backbone of the society. Humans live around all kinds of organizations, such as neighborhood communities, businesses, schools, unions, political, sports, and religious organizations, etc. These organizations have a set of members, each playing a specific role, which determines their duties and functionalities within the organization. One of these functionalities is to offer a range of services to members of the organization and external people. These members must follow a set of norms to ensure the proper functioning of the organization and should pursue the global goals of the organization.

A feature that is repeated in organizations is that they are not static but dynamic, resulting in changes in both its structure and the way in which they behave. In an organization, any of its elements is prone to change due to situations that occur in the organization itself or its environment. Researchers in the field of social sciences and organizations have studied such situations, the reasons why they appear and solutions and actions to be taken to ensure that this situation does not damage the organization or to take advantage of the situation. These situations are known as 'Forces that drive organizational change'.

Human organizations are the main source of inspiration for the Multi-Agent Systems (MAS) based on organizations. These systems are computational abstractions that are populated by agents instead of people, but take into account organizational elements such as roles, services, goals, norms, etc. However, the proposals that have been presented up to now to define this type of MAS are focused mostly on static systems, without changes in its structure. Moreover, in the few proposals that take into account organizational changes, they just state that changes occur, but without specifying the reason for change. Thus, the concept of 'forces that drive organizational change' (and their features) is not considered.

Therefore, the objective of this PhD thesis is to translate the knowledge of the forces that drive organizational change available in human organizations to MAS-based organizations. These forces will be formally expressed with the factors that help to detect them. The solutions to be taken when a force is detected will also be presented. To correctly perform this task, a formalization for virtual organizations is designed, named *Virtual Organization Formalization* (VOF). Moreover, the *Artifacts for Organizational Mechanisms* are proposed, which are a tool to help in the representation of organizational knowledge and in the modeling of the environment of the organization. This tool is based on the *Agents & Artifacts* (A&A) framework.

Resumen

Las organizaciones se han convertido en el eje central de la sociedad. Los seres humanos vivimos alrededor de organizaciones de todo tipo, como comunidades de vecinos, negocios, escuelas, sindicatos, organizaciones políticas, deportivas, religiosas, etc. Dichas organizaciones poseen una serie de miembros, cada uno jugando un rol determinado, que establece sus obligaciones y funcionalidades dentro de la organización. Una de estas funcionalidades es la de ofrecer una serie de servicios tanto a miembros de la organización como a personas externas a la misma. Estos miembros deben seguir una serie de normas para asegurar el correcto funcionamiento de la organización y deben perseguir los objetivos globales perseguidos por la organización.

Una característica que se repite en todas ellas es que no son estáticas, sino dinámicas, produciéndose cambios tanto en su estructura como en la forma en la que se comportan. En una organización, cualquiera de sus elementos es propenso a sufrir una modificación debido a situaciones que se producen en la propia organización o en su entorno. Investigadores en el campo de las ciencias sociales y las organizaciones han estudiado dichas situaciones, los motivos por las que aparecen y las soluciones y acciones a tomar para asegurar que esta situación no dañe a la organización o para sacar provecho de una situación. Estas situaciones se conocen como 'Fuerzas que mueven el cambio organizativo'.

Las organizaciones humanas son la mayor fuente de inspiración para los Sistemas Multiagente (SMA) basados en organizaciones. Estos sistemas son abstracciones computacionales que están poblados por agentes en vez de por personas, pero tienen en cuenta elementos organizativos como roles, servicios, objetivos, normas, etc. Sin embargo, las propuestas que se han presentado hasta ahora para definir este tipo de SMA se centran en su mayoría en sistemas estáticos, sin cambios en su estructura. Además, las pocas propuestas que toman en cuenta cambios organizativos sólo especifican que se producen una serie de cambios, pero sin precisar el motivo de dicho cambio. Así, el concepto de 'fuerzas que conducen el cambio organizativo' (y sus características) no se considera.

Por lo tanto, el objetivo de esta tesis doctoral es trasladar el conocimiento que se tiene en las organizaciones humanas sobre las fuerzas que mueven el cambio organizativo a los SMA basados en organizaciones. Se presentarán dichas fuerzas expresadas formalmente con sus correspondientes factores que ayudan a su detección. Las soluciones que deben tomarse cuando una de ellas es detectada también serán presentadas. Para realizar correctamente esta tarea, también se presenta una formalización para organizaciones virtuales llamada *Virtual Organization Formalization* (VOF). Además, se han propuesto los *artefactos para mecanismos organizativos*, que son una herramienta que ayudará en la representación del conocimiento organizativo y en el modelo de entorno de la organización. Esta herramienta está basada en el *framework Agents & Artifacts* (A&A).

Resum

Les organitzacions s'han convertit en l'eix central de la societat. Els sers humans vivim al voltant d'organitzacions de qualsevol tipus, com a comunitats de veïns, negocis, escoles, sindicats, organitzacions polítiques, esportives, religioses, etc. Les dites organitzacions posseïxen una sèrie de membres, cadascú jugant un rol determinat, que establix les seues obligacions i funcionalitats dins de l'organització. Una d'estes funcionalitats és la d'oferir una sèrie de servicis tant a membres de l'organització com a persones externes a la mateixa. Estos membres han de seguir una sèrie de normes per a assegurar el funcionament correcte de l'organització i han de perseguir els objectius globals perseguits per l'organització.

Una característica que es repetix en totes elles és que no són estàtiques, sinó dinàmiques, produint-se canvis tant en la seua estructura com en la forma en què es comporten. En una organització, qualsevol dels seus elements és propens a patir una modificació degut a situacions que es produïxen en la pròpia organització o en el seu entorn. Investigadors en el camp de les ciències socials i les organitzacions han estudiat les dites situacions, els motius per les quals apareixen i les solucions i accions a prendre per a assegurar que esta situació no danye a l'organització o per a traure profit d'una situació. Estes situacions es coneixen com 'Forces que mouen el canvi organitzatiu'.

Les organitzacions humanes son la major font d'inspiració per als Sistemes Multiagent (SMA) basats en organitzacions. Estos sistemes son abstraccions computacionals que estan poblats per agents en compte de per persones, però tenen en compte elements organitzatius com a rols, servicis, objectius, normes, etc. No obstant això, les propostes que s'han presentat fins ara per a definir este tipus de SMA se centren majoritàriament en sistemes estàtics, sense canvis en la seua estructura. A més, les poques propostes que prenen en compte canvis organitzatius tan sols especifiquen que es produïxen una sèrie de canvis, però sense precisar el motiu del dit canvi. Així, el concepte de 'forces que mouen el canvi organitzatiu' (i les seues característiques) no es considera.

Per tant, l'objectiu d'esta tesi doctoral és traslladar el coneixement que es té en les organitzacions humanes de les forces que mouen el canvi organitzatiu als SMA basats en organitzacions. Es presentaran les dites forces expressades formalment amb els seus corresponents factors que ajuden a la seua detecció. Les solucions que han de prendre's quan una d'elles és detectada també seràn presentades. Per a realitzar correctament esta tasca, també es presenten una formalització per a organitzacions virtuals anomenada *Virtual Organization Formalization* (VOF). A més, es proposen els *artefactes per a mecanismes organitzatius*, una eina que ajudarà en la representació del coneixement organitzatiu i en el model d'entorn de l'organització. Aquesta eina es basa en el *framework Agents & Artifacts* (A&A).

To my family.

Acknowledgements

This is it. My PhD thesis is over, and in the book you have in your hands (or in the PDF file you have downloaded) you can find the results. However, this thesis could not have been possible without the people that helped and supported me. I have to say thank you to many people from different countries. Therefore, let me use this section for this purpose, and let me use different languages.

En primer lugar, gracias a mis directores, Fanny y Vicent. A Fanny le agradezco su dedicación en el día a día de la realización de la tesis, introduciéndome en el mundo de la investigación y la docencia y transmitiéndome sus conocimientos, consejos y ayuda para haber podido llevar a buen puerto este trabajo. Muchas gracias a Vicent, sobretodo por darme la oportunidad de trabajar e integrarme en el GTI-IA hace ya unos 6 años. En este sentido no puedo olvidarme de Vicente Julián, responsable de la beca FPI y del proyecto que han hecho posible esta tesis. Además, junto con Carlos me dieron la oportunidad de impartir docencia a su lado y a formarme también desde esta perspectiva.

Je tiens à remercier Olivier leur aide lors de mes visites à l'École des Mines de Saint-Étienne. Les trois mois, ont été très productives et très important de finir la thèse. Même la neige pourrait rendre le séjour est satisfaisante dans tous les sens. Merci beaucoup.

東京の国立情報学研究所では、多くのご支援と心からのおもてなしを賜りました、佐藤一郎教授に深く感謝いたします。同所では、多くの事を学び、リサーチ、コンピューティング、エージェントのみならず、日本文化も学ぶことができました。私自身日本の文化に非常に魅了されました。

同所のスタッフの皆様、特に国際インターンシッププログラムのご担当の皆様に心より感謝申し上げます。

Gracias a mis coautores Ramón y Roberto de la Universidad Rey Juan Carlos (al menos, cuando empezamos a trabajar juntos ambos trabajaban allí y Juan Carlos era el rey de España), por su colaboración en uno de los trabajos más importantes de esta tesis doctoral. Además, gracias por su buena acogida y los buenos momentos tanto en Valencia, en Móstoles, o en Boca Raton (la ciudad donde es más difícil encontrar un taxi de todo el mundo).

Queridos compañeros de laboratorio (más conocido como Frikilab) durante este tiempo, os agradezco vuestra compañía, ayuda en el trabajo y diversión en los momentos más relajados. Habéis sido muchos los que habéis pasado, y hemos estado más o menos apretados, pero siempre ha estado bien. Lo mismo digo para el resto de compañeros del GTI, tanto becarios como profesores.

Y de todos los compañeros que he tenido durante mi tiempo de tesis, debo destacar a los Miserables (Alejandro, Sergio, Pablo y Víctor) con los que también he compartido muchos momentos fuera de la universidad. La filosofía miserable se ha extendido además por todo el mundo, desde Milán hasta Boston. Un recuerdo especial para Sonia, Clara,

Lucía, Sergio y Javi, que han trabajado en otros grupos del departamento y con los que también hemos pasado y pasaremos mucho tiempo juntos. Por supuesto, no me olvido de parejas y consortes de los susodichos que tampoco se pierden una y que también forman parte de esta tropa aunque nunca hayan trabajado en nuestro departamento.

No puedo olvidarme de otros compañeros, los del NII, en especial los españoles, que hicieron mi estancia en Tokio mucho más agradable. Desde Pilar, que me enseñó tanto los entresijos del NII como sus alrededores en los primeros días de mi estancia, hasta los miembros del autodenominado NII Elite Team por cosas como las eternas sobremesas en los sofás de la planta 18 con vistas a partes tan emblemáticas de Tokyo como Akihabara y el Tokyo Skytree. Gracias también a la gente que conocí fuera del entorno de trabajo, en especial a Elena, Kaoru y Phil. Espero veros pronto en cualquier lugar del mundo.

Thank you to all the people from the ISCOD in Saint-Étienne, specially to Reda, Andrei, Alex, Jeremy, Bissan, and Gauthier. You made my visit to France really comfortable and I wish the best to you in the future. Who knows if we can work in the same place again for a postdoc.

Saliendo del entorno universitario, tengo que dar las gracias a los Depras. Nos conocimos en septiembre de 2003, el primer día de carrera, y todavía seguimos ahí, juntos como el primer día y viendo como dejamos una juventud atrás (por no decir adolescencia) para hacernos mayores y pasar a una vida adulta. Escribo estas líneas pocos días después de una cena que creo que nos hizo darnos cuenta de lo que ha llovido, pero que estoy seguro que será la primera de muchas cenas de este tipo. No sé donde me llevará el futuro cercano, pero lo que está claro es que siempre estaréis presentes.

Para terminar, debo dar las gracias a mi familia, a mis padres y a mi hermano. Gracias no sólo por la ayuda y apoyo durante los cuatro años de la tesis, sino desde el primer día de mi vida. Además han sido partícipes de mis estancias, con sendas visitas a Meguro y a la Residence des Arts de Saint-Étienne. El camino hasta aquí ha tenido sus baches pero lo hemos conseguido. Gracias por todo.

This thesis was developed and written in Valencia, Golosalvo, Tokyo, and Saint-Étienne.

P.S.: Let me also thank other people which do not really had direct influence in this thesis, but they are also very important to me. First of all, thank you to Dr. James Naismith for inventing the basketball. This great sport has been present in my life for many years. Coincidentally, I started the university and got my first season ticket for Valencia Basket in the same year, 2003. During the years of my thesis I had the opportunity to attend and enjoy basketball games in 5 different countries, especially following Valencia Basket, Ros Casares Valencia, and Toyota Alvark Tokyo, and I even made a pilgrimage to Springfield, MA (USA) where the Game was invented.

Gracias a los camareros y cocineras de la cafetería del Conservatorio por alimentarme con sus patatas bravas, *manhattans*, calamares, macarrones y demás *delicatessen* que me han dado energía para afrontar las tardes trabajando en la tesis.

Videogames were one of the main reasons for choosing computer science as my career. Since I received my first Game Boy back in 1991 I wanted to know how they are developed, not only graphically but also its background. This is why I selected Artificial Intelligence as my specialization. I want to thank the people that made the industry grow and become one of the most important of our days.

Finally, here you have a small tracklist. This is like the soundtrack of my thesis, containing the songs that I listened the most, or had a special meaning, during this period:

- Get Lucky - Daft Punk
- Suite Escapism - Hiromi
- So long - AKB48
- El rock de la paella - El tio Fredo
- Nightcall - Kavinsky
- Karma Chameleon - Culture Club
- Don't stop believin' - Journey
- Apache - The Sugarhill Gang
- Let it flow - Grover Washington Jr.
- Je veux - Zaz

'Aquí estoy porque he venido, porque he venido aquí estoy. Si no le gusta mi canto, como he venido me voy.' (Héctor del Mar)

GAME OVER.

Contents

List of Figures

List of Tables

1

Introduction

Our society is structured around organizations. An organization is a social entity that has a collective goal and is linked to an external environment. The word 'organization' is derived from the Greek word *organon*, itself derived from the better-known word *ergon* which means 'organ'. There are different types of organizations, like corporations, governments, non-governmental organizations, armed forces, charities, cooperatives, universities, political parties, or even sport clubs. All kinds of organizations are provided with a set of members (i.e., agents) that populate the organization and play a role (i.e., a position) inside the organization. Different roles inside an organization include clients, managers, providers, etc. These members are aimed to fulfill the goals that the organization has associated. The goals are related to the reasons that motivated the creation of the organization. For example, the goal of a basketball club, which is created to allow its members to play basketball in an organized way, is to participate in different competitions. To achieve its goals, the organization has a set of services (provided by some of its members) that supply with functionalities to other members of the organization, and to possible external clients that could be attracted by these functionalities. Additionally, the members of the organization have to follow a set of norms that regulate the behavior of an organization and to drive its behavior to the accomplishment of the organizational goals. The members (as well as relationships between these members such as decision and information flows, power relations, etc.), roles, goals, services, and norms of an organization define its structure. Depending of how they are defined, organizations have a different type of structure such as bureaucracy, hierarchy, matrix, teams, etc.

Organizations are not static, but they might be able to change any of their elements during their life-cycle in order to be adapted to environmental and internal requirements. Howard Aldrich (Ald99), Kurt Lewin (Lew51), and other researchers of Organizational Theory[1] (Daf03) have established that changes in an organization appear because it exists a set of *forces that drive the organizational change*. Depending on where the source of change is located, they classify these forces between internal and external forces. When any of these forces is detected in the organization, in a usual situation, this organization has to change to be adapted to the new conditions,[2] to take advantage of the forces, or to avoid any damage that could be caused by them. Forces are an important issue to be taken into account by organizational managers, which have to make sure that forces are monitored, and to provide the means to assure that the organization adapts to the new environmental or internal circumstances.

Human organizations have been the main inspiration for the *Organization-Centered Multi-Agent Systems* (OCMAS) (FG98), which are a specific type of Multi-Agent Systems (MAS) (WJ95) where the organization is explicitly defined,

[1]Organizational Theory is the sociological study of formal social organizations, such as businesses and bureaucracies, and their interrelationship with the environment where they operate.

[2]In other situations, the organization is able to change its environment to suit its current structure. For example, government organizations are able to change its institutional environment to adapt it to their needs.

by using elements taken from human organizations such as roles, norms, services, as well as an explicitly-defined structure. Different proposals model these systems from different points of view, but always keeping the organization in mind. Some of these proposals (such as (APDD08, AJGF12)) describe how to carry out changes during the execution of the organization, and the associated cost that they would have associated. However, these works do not take into account the forces that drive organizational change, nor which are the reasons that provoke the change of an organization. Since these forces are an important feature in human organizations, it is interesting to consider them when developing an OCMAS.

1.1 Motivation

The forces that drive organizational change are an important issue to consider when defining changes inside an organization. As it has been stated in the introduction, these forces have not been explicitly considered and adapted into the OCMAS domain, even when OCMAS that may change its structure, roles, goals, norms and other organizational elements during its execution (i.e., adaptive OCMAS) are designed and developed. Only Hoogendoorn (HJST07) takes into account the existence of a set of forces that drive organizational change, but he does not detail the forces individually. Since OCMAS take inspiration from human organizations and Organizational Theory, it is interesting to also take these forces into consideration, and to 'translate' them to a computational domain so as to improve how adaptation is carried out in these systems. However, this is not a trivial work, and different methods and tools would be required in order to assure a correct insertion of the forces into an OCMAS definition. In this PhD thesis, these forces are included into the OCMAS domain. For correctly representing the forces in this proposal, it is necessary a modification of the representation of the environment to make it closer to the human environment. Also, a formal language is required so as to describe the forces, which are normally defined in natural language in the human domain.

Since the environment is one of the main sources where change comes from, it is necessary that designers and developers of OCMAS may be provided with a proper tool to describe the environment. One of the proposals to define the environment of a MAS is the *Agents & Artifacts* (A&A) conceptual framework (RVO07), which is really interesting for modeling the forces because it models the environment in a similar way to how the real world is modeled. Designers of the A&A framework define a MAS by means of three types of entities: (i) agents, the proactive elements of the system that have associated goals to achieve; (ii) artifacts, reactive entities that provide functionalities to the agents that will use them to achieve their goals; and (iii) workspaces, which structure the environment, presenting an exact location inside the environment and can be intersected and nested between them. Although this representation is appropriate to correctly design a MAS, it is necessary to adapt the ideas of the A&A framework in order to be able to represent OCMAS in a proper way. Therefore, this PhD thesis defines a new set of artifacts, based on *Organizational Mechanisms* (CBHO09) that provide functionalities for implementing a virtual organization and its environment.

The forces that drive organizational change are expressed in the real world using natural language. Since this is a language that computers cannot understand, it is then necessary to define a formal language that approaches these forces to the computational domain, making them understandable for agents and systems. Current formal descriptions of OCMAS (such as (HJST07)) do not provide with enough tools for describing the forces that drive organizational change. This thesis intends to close this gap, and presents the Virtual Organization Formalization

(VOF) to define virtual organizations taking into account all the required elements to monitor and solve a force (e.g., roles, norms, goals, services, etc.).

Using VOF it is possible to define and model the forces that drive organizational change inside the OCMAS domain, thus providing the designers and developers with the means to identify and then find a solution for the aforementioned forces. In this thesis, a definition of the forces that drive the organizational change for the design step of the software development process is proposed.

This PhD thesis has been developed in the framework of several research projects of the Spanish Government and the European Commission that focus on Agreement Technologies (LM08) and Virtual Organizations (VO). These research projects are:

- “THOMAS: MeTHods, Techniques and Tools for Open Multi-Agent Systems” under grant TIN2006-14630-C03-01 (Main researcher: Vicente Botti Navarro, funded by the Spanish Government from 2006 to 2009). The main goal of this project was the development of techniques and methods suitable for the creation of open MAS that are capable of solving problems in an autonomous and flexible way. These systems are characterized by the heterogeneity of their participants; their limited trust; a high uncertainty; and the existence of individual goals that might be in conflict. In these scenarios, norms are conceived as an effective mechanism for ensuring social order and avoiding conflicts.

- “OVAMAH: Adaptive Virtual Organizations: Mechanisms, Architectures and Tools” under grant TIN2009-13839-C03-01 (Main researcher: Vicente J. Julian Inglada, funded by the Spanish Government from 2009 to 2012). The aim of this project is to provide a virtual organization with autonomy capabilities that allow it to have a dynamic response in view of potential changing situations by means of the adaptation or evolution of the organization. In this way, it will be able to detect situations of interest (e.g., operation errors) and to manage them maximizing the adaptation flexibility and capacity. The author of this thesis was awarded with a 4-year FPI (*Formacion de Personal Investigador*, Research Staff Training) grant related to this project.

- "Agreement Technologies" CONSOLIDER-INGENIO 2010 under grant CSD2007-00022 (Main researcher: Carles Sierra, funded by the Spanish Government from 2007 to 2014). Agreement technologies (AT) refer to computer systems in which autonomous software agents negotiate with one another, typically on behalf of humans, in order to come to mutually acceptable agreements. Agents are grouped around organizations, where they develop these negotiations and other processes related to their life-cycle.

- COST Action IC0801 on Agreement Technologies (Main researcher: Sascha Ossowski, funded by the European Commission, from 2008 to 2012). The overall mission of the COST Action was to support and promote the harmonization of nationally-funded high-quality research towards a new paradigm for next generation distributed systems based on the notion of agreement between computational agents, fostering research excellence and sowing the seeds for technology transfer to industry.

- "iHAS: Human Agent Societies: Design, Formation and Coordination" under grant TIN2012-36586-C03-01 (Main researcher: Vicente J. Julian Inglada, funded by the Spanish Government from 2012 to 2015). This project proposes the development of mechanisms, algorithms, tools, and models that allow the creation of open systems where virtual agents and humans coexist in a totally integrated environment.

1.2 Objectives

The main objective of this thesis is to introduce the forces that drive organizational change in the OCMAS domain, thus allowing the development of virtual organizations that are dynamic (i.e., they can change its elements and structure at runtime) but they are also aware of the sources of this change, so they can react and adapt to the change in the best possible way to take advantage or to avoid damage from the situation that provoked the change. This objective is split in these sub-objectives:

1. **Study of the state-of-art of Adaptive Virtual Organizations.** Prior to start working on the content of this thesis, it is necessary to give the definitions this thesis focuses on. In order to provide a complete state-of-art, it is necessary to review terminology for organizational-related concepts like adaptation, including the different types of adaptation that can appear in an organization, and the mechanisms that give support to the adaptation process. It is also necessary to identify the external and internal forces that are responsible for the organizational change. Finally, proposals that support adaptation and self-organization in MAS are studied, and examples of Adaptive MAS are provided. Organizational Theory researchers have provided with different definitions related to organizational change. Therefore, a wide research on these definitions gives an accurate state of the art that clarifies concepts related to Adaptive Virtual Organizations.

2. **Formal description of the organization.** Differently from the human perspective, to define an OCMAS it is necessary not only a natural-language description, but a more formal description. In this thesis, a new formal language, named *Virtual Organization Formalization* (VOF), is proposed to formally define a Virtual Organization.

3. **Definition of forces that drive organizational change in the OCMAS domain.** After the analysis of the forces that drive the organizational change from a human organization point of view, they require to be translated into the OCMAS domain. A set of patterns for the analysis of the forces is presented. The patterns use the defined formal language to bring the forces definition closer to the computational domain. Three types of patterns are defined: (i) the description of the force; (ii) the set of factors that trigger each force, and (iii) the set of solutions to take advantage or to prevent damage from the forces. Finally, the implementation of the means to monitor and solve forces is carried out using Jason agents (BHW07), the artifacts for organizational mechanisms implemented in CArtAgO (RVO06) and the organizational definition from VOF.

4. **Development of a new environmental definition.** The newly developed environmental definition based on Artifacts for Organizational Mechanisms (and also workspaces to structure the environment) provides the appropriate framework to deploy the monitoring elements for detecting the forces, and the solutions for them.

1.3 Contributions

The specific contributions of this thesis are:

- **Virtual Organization Formalization.** Proposes a formal approach to define a virtual organization including all its elements and entities, such as roles, goals, services, norms, and organizational units.

- **Templates to define forces in the design phase.** These templates are intended to extract the information from the social sciences domain, usually unstructured and written in natural language. Each force is characterized by two tables: (i) a table specifying the general information of the force; (ii) a set of factors, where each factor that affects the force is represented by another table; and (iii) a set of solutions, where, similarly to the factors, each solution is represented individually in a table. The templates are then used to specify the implementation phase of the organization.

- **Artifacts for organizational mechanisms and new environmental definition.** Defining the forces for the OCMAS domain requires the usage of elements that help on the identification and solution of these forces. The Organizational Mechanisms present properties for the management of an organization, and based on the Agents & Artifacts conceptual frameworks, this thesis presents the *Artifacts for Organizational Mechanisms* that encapsulate features of the Organizational Mechanisms (CBHO09) into artifacts. Also following this framework, a new environmental definition is presented using these artifacts and workspaces to structure the environment, making it closer to the real-world definition.

1.4 Document structure

This document first presents in Chapter 2 the background and state of the art related to this thesis. Then, the following chapters present the contributions of this thesis. Chapter 3 presents the Virtual Organization Formalization, a formal definition for virtual organizations. Chapter 4 presents the main contribution of this thesis, which are the templates to model the Forces that Drive Organizational Change at the design time of a Virtual Organization (VO), making it possible to define Adaptive VOs. Chapter 5 presents an abstraction that facilitates the modeling and implementation of an Adaptive VO, named *Artifacts for Organizational*

Mechanisms. Chapter 6 presents a case of study, based on a smart virtual building, modeled as an Adaptive VO, thus taking into account the Forces that Drive Organizational Change. This modeling is carried out using not only the proposed templates to identify the forces, but also the Virtual Organization Formalization and the Artifacts for Organizational Mechanisms to model the building and its environment as an Adaptive VO. Finally, Chapter 7 presents the conclusions, contributions, and future work on the topics developed in this thesis.

2

State of the Art

This chapter describes the state of the art on different concepts and proposals related to adaptation in organizations in general and, more specifically, to Organization-Centered Multi-Agent Systems (OCMAS). Initially, the main concepts on the topic of adaptation from both human organizations and Multi-Agent Systems (MAS) are detailed in Section 2.1. Next, the different types of organizational change that have been studied at the business field are presented at Section 2.2. Additionally, it has been largely studied the set of forces that drive organizational change. These forces can be external to the organization, such as market forces, or laws; or internal, such as growth, power or goal achievement. A brief description of these types of forces is included in Section 2.3. The different types of change that an organization is able to suffer are depicted in Section 2.4, being these changes taken into account either from a temporal or a structural point of view. Finally, this chapter makes an overview on the two types of adaptation that are carried out in MAS. On the one hand, self-organization, which involves bottom-up changes from agents to the organization itself, is described in Section 2.5. On the other hand, reorganization is the type of change that modifies the organization, and then agents populating it have to adapt to these changes. Reorganization, which follows a top-down approach, is described in Section 2.6. Then, Section 2.7 focuses on additional proposals for the design and development of MAS with adaptivity properties. This topic is studied from two different points of view to develop this kind of systems. On the one hand, different Agent-Oriented Software Engineering (AOSE) methodologies to develop MAS and OCMAS are presented. On the other hand, proposals based on formal and logic frameworks are described. Finally, Section 2.8 gives the conclusions on this chapter.

2.1 Concepts

This section clarifies the main concepts that this work is supported by, such as Organization-Centered Multi-Agent Systems, Virtual Organizations, Adaptation, Adaptive Virtual Organization, and Organizational Mechanisms.

2.1.1 Organization-Centered Multi-Agent Systems

Multi-Agent Systems (MAS) (Woo02) is a general software technology which is motivated by fundamental research topics about autonomy, cooperation, group formation, etc. Currently, MAS have been applied in a wide number of domains, such as negotiations (SAJBGF11), tourism (ESAA^{+}10), or even videogames (ABC09), with interesting results. Methodologies for designing Multi-Agent Systems (MAS) were initially focused on the concept of agent, in a way that all the analysis and design of the system was carried out from the point of view of individual agents. Therefore, in those methodologies, the concept of agent's organization (DW04) was not defined in an explicit way, but the structure was defined in an implicit manner by means of relationships and the internal needs of the agents that populate the system. This type of methodologies, known as Agent-Centered MAS (ACMAS) methodologies, are used to design closed MAS, where agents are provided with totally defined behavior and functions, being cooperative and benevolent. However, in these systems there is no possibility to have external agents or agents having a behavior that is not adjusted to the rules (ZJW01). For example, methodologies like Gaia (ZJW03) and PASSI (Cos05) are focused on designing ACMAS.

In the last years, MAS researchers presented a new design trend, named open systems (GPL07), which allows external agents to enter the system, having the same rights to access system's functionalities than internal agents. External agents can have totally different objectives than the ones from internal agents, enacting interested, not benevolent nor cooperative behavior over the rest of the system. On the other hand, internal agents are implemented to achieve MAS global objectives (HSB02b). In order to regulate an open MAS, the concept of **organization** is adopted as an entity of the system that has its own objectives (called organizational or global goals), defined in an explicit way, and that organizational agents must help to achieve. For assuring this situation, the organization is enhanced with norms (BdT04) and restrictions which allow regulating the behavior of internal and external agents that populate the system. A fulfilment or violation of a norm will cause the agent to be rewarded or sanctioned, respectively. Moreover, the system is organized following a structure or topology that agents will adopt, and that will be useful to define communication protocols (FFMM94) and hierarchy inside the MAS (IGG99). Therefore, recent MAS methodologies are focused on the design of this kind of systems, named Organization-Centered Multi-Agent Systems (OCMAS) (FGM03). For example, some methodologies that support OCMAS are SODA (Omi01), INGENIAS (PGSF05) or GORMAS (ABJ11).

2.1.2 Virtual Organization

The concept of Virtual Organization (VO) firstly appeared in the business field. *BusinessDictionary.com* defines **Virtual Organization** as ‘an organization that does not have a physical (bricks and mortar) presence but exists electronically (virtually) on the Internet, or an organization that is not constrained by the legal definition of a company, or an organization formed in an informal manner as an alliance of independent legal entities’.

Aguer-Hortal (AH05) defines a virtual organization as ‘an organization built by people that is not located in the same physical space, even not in the same city or country. This organization has no frontiers, where its clients are not able to know its employees’. In general terms, he states that a virtual organization is an organization that develops its whole activity (or a part of it) in a virtual way.

DeSanctis and Monge (DM98) define a virtual organization as ‘a collection of geographically distributed, functionally and/or culturally diverse entities that are linked by electronic forms of communication and rely on lateral, dynamic relationships for coordination’. Despite its diffuse nature, a common identity holds the organization together in the minds of members, customers, or other constituents. The virtual organization is often described as one that is replete with external ties (CCS95), managed via teams that are assembled and disassembled according to needs (GM95), and consisting of employees who are physically dispersed from one another (Cla94). The result is a ‘company without walls’ (Gal95) that acts as a ‘collaborative network of people’ working together, regardless of location or who ‘owns’ them (GM95).

Katzy (Kat98) cites Byrne *et al.* (BBP93), who defines virtual organizations as ‘a temporary co-operation of independent companies (suppliers, customers, even erstwhile rivals) linked by information technology to share skills, costs, and access to other markets'. Katzy also states that virtual organizations center on continually restructuring to capture the value of short time windows of opportunity. Restructuring becomes the normal day-to-day business.

Other approaches define virtual organizations as an ad-hoc overlay. Barnatt (Bar95) says that these organizations exist in the media in which electronic communication and computer programs exist, that they develop proportionally with the development of information and communication technology and that they can be found within conventional organization structures. Bača *et al.* (BSD07) define a virtual organization as a target-oriented suprastructure of geographically separated entities (organizational units) that are specialized for a predefined area of activity and are interconnected through space, time, and organizational limitations.

Blecker (BN00) made a survey on different definitions of virtual organizations and he yielded the following features of VOs in business field:

- intended temporary cooperation of legally and economically independent companies, whose participants concentrate on their core competencies as well as the temporally limited and order-related bundling of these core competencies
- reciprocal supplement of the actors during the production process
- mutual agreement of targets
- high reciprocal trust
- high value of customer orientation
- no centralized and/or formalized organizational structure

- real structures of the virtual organization and its participants are only limited observable by the market partners
- intensive application of modern information and communication technologies
- individualized products

The term Virtual Organization was later taken to be used in the computer science field of research. More precisely, in one of the most trending topics in distributed computed, Grid Computing (FK01). This is a field of distributed computing that focuses on large-scale, high-performance and innovative systems. Foster *et al.* (FK01) define a VO as 'a set of individuals and/or institutions defined by a sharing of computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science, and engineering'. They also state that VOs vary tremendously in their purpose, scope, size, duration, structure, community and sociology.

The term Virtual Organization was also used in Multi-Agent Systems (NPC$^+$04), which is another type of distributed computing, and also a research topic inside Artificial Intelligence. In this case, the 'Virtual' concept of the Virtual Organization term normally refers to its 'virtuality', i.e. its software existence. Thus, 'Virtual Organization' or 'agent organization' terms are usually used without distinction. Ferber (FGM03) defined an agent organization as 'a set of agents that interact together to coordinate their behavior and often cooperate to achieve some collective goal'. Dignum (Dig09) describes that an organization in multi-agent systems can be understood as complex entities where a multitude of agents interact, within a structured environment aiming at some global purpose. Additionally, agent organizations demand the integration of organizational, intermediate level and individual perspectives, the dynamic adaptation of models to organizational and environmental changes, and rely on a great extent on the notion of openness and heterogeneity of MAS. From an organizational perspective, the main function of an individual agent is the enactment of a role that contributes to the global aims of the organization. Argente (ABJ11) states that a virtual organization is employed to describe a set of agents that, using roles and interaction patterns, coordinate themselves in order to achieve the global goals of the system.

Summarizing, a Virtual Organization in the MAS field presents the following features:

- It is composed by agents, independently from their internal features and individual objectives.
- It follows a global goal, which is not dependant from the agents' individual objectives.
- Tasks to be executed by agents are divided by means of roles, which describe the activities and functionalities of the organization.
- The system is distributed in groups or organizational units where interaction between agents takes place.
- Its bounds are clearly defined, determined by the environment of the organization, the internal and external agents (internal agents are the ones that have been implemented for the organization, while external agents are the ones that come from the environment and joined the organization), as well as the functionality and services offered by the organization.

2.1.3 Adaptation

The Merriam-Webster dictionary (Mis97) defines adaptation as 'the act or process of adapting, the state of being adapted'. This word has other meanings, such as 'adjustment to environmental conditions', 'adjustment of a sense organ to the intensity or quality of stimulation' or 'a modification of an organism or its parts that makes it more fit for existence under the conditions of its environment'. Despite an organization is not a living being, this last definition is the one that best describes the objective followed when using adaptation inside an organization.

In the field of Organizational Theory (GS98), adaptation is defined in different ways, ranging from an *strategic choice* (HJ85), referred to the planned pursuit of ends based on a rational assessment of available means and conditions, resulting in an explicit decision to change the organization; to an *environmental determinism* (HJ85), where environmental change is a response to environmental requirements.

In the field of MAS, Dignum (Dig09) considers adaptation as a design factor that requires a decision, giving as a result the modification of some features of the organization. These decisions may come from two different aspects: temporality and intentionality. On the one hand, there are two types of temporal change (DSD04). A temporal change is *proactive* when it prepares the system for an unpredictable event that can be produced in the future, and *reactive* if the change in the organization is carried out after a given event has occurred. On the other hand, change can be classified by its intentionality (DSD04). In this case, the change can be *offensive*, if the organization wants an advantage to compete, or *defensive*, if the organization looks for its survival.

In order to make a proactive adaptation, agents must be equipped with mechanisms that allow them to reason about and to evaluate the current organizational behavior, the desired organizational behavior, and the utility of the organization. In case of reactive adaptation, agents must be able to sense and react to environmental changes, so they can be simpler agents.

Both factors can be combined to obtain the four possible ways for an agent organization to change (EAP97):

- **Preventive** (proactive, offensive): takes advantage of possible future events and it is useful when the future is unpredictable and when innovation is an element of competition. For example, the management system of a university can introduce new agents playing the role of a teacher in anticipation of a growing in the number of students for the next academic year, even without being sure of that growing.

- **Protective** (proactive, defensive): applied to limit the damage produced in the future. This kind of change is carried out, for example, when the security of a specific place grows due to the visit of authorities, being necessary to add new 'police' agents, also modifying the organization of the regular security of that place.

- **Exploitative** (reactive, offensive): it happens after an event, to exploit the opportunities that have been arisen because of its execution. For example, after the celebration of a big event, like the America's Cup in Valencia's marina, these facilities were used as a leisure area, which was also part of the Formula 1 street circuit, being necessary to adapt structural aspects of the city such as bus and tram networks in order to facilitate the access of citizens to this area.

- **Corrective** (reactive, defensive): used to prevent more damage, when other strategies fail, trying to assure the existence of the organization. For example, in a business environment, if a company is losing large amounts of money, its structure will be reorganized by deleting departments and firing employees to minimize its losings.

2.1.4 Adaptive Organizations

After defining what a virtual organization is and what adaptation means it seems necessary to know what an adaptive organization is, focusing on both human and computational (more specifically, Organization-Centered Multi-Agent Systems) organizations.

BusinessDictionary.com defines an adaptive organization as a chameleon like organization able to keep up with the rapid changes in its environment. One of the strategies these organizations employ (to stay fast and flexible) is entrusting of more decision making powers and associated resources to their employees. The organization theory refers to this process as *entrusting* (JS93).

Barnett and Carroll (BC95) state that a change in the world of organizations occurs mainly through the adaptive responses of existing individual organizations to prior changes in technology, environment, or another organizational element. When an environment changes, some organizations fail and some others appear. Organizational change involves a transformation of an organization which is carried out between two points in time. An organization changes due to internal and external factors (Ald99). External factors involve features related to the environment, while internal factors include, for instance, organizational changes that are applied when and how organizational managers decide, maybe following a pattern.

Aldrich (Ald99) expresses that organizations not only react to environment changes, but also alter their structures in adaptive ways, with changes in goals, boundaries and activities. A transformation is a major change in an organization, involving a break with existing routines and a shift to new kinds of competencies that challenge organizational knowledge.

In a general way, an adaptive organization in the business field is an organization that is able to change as a response to a modification in its environment or in any structural element (goals, activities, etc.). This change will be produced under the pressure of internal or external forces, which are described in Section 2.3.

Summarizing, an adaptive organization in the business field has the following features:

- It is located in an unpredictable and changing environment.
- There are internal and external forces that act upon it and drive organizational change.
- Its managers have enough knowledge about organizational structure and behavior and have to be able to decide how the organizational change will be deployed.
- The organization prefers transformation instead of extinction.

Therefore, after studying these definitions, we can define an adaptive organization from a consensus point of view as an organization that is able to respond to changes produced either in its environment or in any of its structural elements. This change will be provoked by a set of internal and external forces that drive organizational change.

In the field of computer science, some definitions were also presented for adaptive organizations, focusing on adaptive OCMAS. For example, Aldewereld *et al.* (APDD08) define adaptive software systems as 'those that must have the ability to cope with changes of stakeholders' needs, changes in the operational environment, and resource variability'.

DeLoach *et al.* (DOM08) define adaptive organizations as distributed systems that can autonomously adapt to their environment. The system must be provided with organizational knowledge and it will design its own organization, based on the current goals and its current capabilities.

Picard *et al.* (PHBG09) describe that an OCMAS is adaptive when it changes whenever its organization is not adequate, i.e. the social purpose is not being achieved and/or its structure is not adapted to the environment. This situation occurs when the environment or the MAS purposes have changed, the performance requirements are not satisfied, the agents are not capable of playing their roles in a suitable way or a new task arrives and the current organization cannot face it. In this case, adaptation implies modifying both organization specification (modifying tasks, goals, structure) and role allocation.

Dignum and Dignum (DD06) state that in order to keep effective, organizations must maintain a good fit with the environment. Changes in the environment lead to alterations on the effectiveness of the organization and therefore in a need to adapt themselves, or at least, the need to consider the consequences of the change to the organization's effectiveness and, possibly, efficiency. On the other hand, organizations are active entities, capable not only of adapting themselves to the environment but also of changing that environment.

An Adaptive Organization in MAS presents the following properties:

- The organization adapts itself if its environment forces it to do so.
- Changes in goals, internal requirements, etc. of the organization could also force an adaptation of the organization as a whole.
- The organization is considered to be an open system since the environment can change and external agents can join the organization.
- The organization is populated by agents playing different roles, some of them being responsible of deciding about adaptation of the organization.

As summary, it can be checked that many different definitions of Adaptive Organization in MAS have been given up to this moment. Based on these definitions, we propose a definition for Adaptive Virtual Organization, which will be employed during the development of this thesis:

Definition 1. *An* ***Adaptive Virtual Organization*** *is a virtual organization that is able to modify both its structural (topology, norms, roles, etc.) and functional (services, tasks, objectives, etc.) dimensions in order to respond or to be ahead of changes produced in its environment, or by internal requirements, i.e., if it detects that its organizational goals are not being achieved in a satisfactory way.*

2.1.5 Organizational mechanisms

Organizational Mechanisms (CBHO09) are mechanisms introduced in a multi-agent system (MAS) with the aim of influencing the agents' behavior towards more effectiveness with regard to the global purpose of the system. They rely on the assumption that agents participating in the system are rational, i.e., agents try to maximize their utility with any action they perform. In order to clarify formalization we first give a definition of MAS on which the remainder formulae rely. We adhere the definition given by Centeno *et al* (CBHO09).

Definition 2. *A Multi-Agent System (MAS) is a tuple* $\langle \mathcal{A}g, \mathcal{A}, \mathcal{X}, \Phi, x_0, \varphi \rangle$ *where:*

- $\mathcal{A}g$ *is a set of agents;* $|\mathcal{A}g|$ *denotes the number of agents in the system;*
- $\mathcal{A}$ *is a finite action space that includes all possible actions that can be performed in the system.* $\mathcal{A}$ *includes an action* a_{skip}*, the action of doing nothing;*
- $\mathcal{X}$ *is the environmental state space;*
- $\Phi : \mathcal{X} \times \mathcal{A}^{|\mathcal{A}g|} \times \mathcal{X} \to [0..1]$ *is the MAS transition probability distribution, describing how the environment evolve as a result of agents' actions;*
- $x_0 \in \mathcal{X}$ *stands for the initial state of the MAS;*

- $\varphi : \mathcal{A}g \times \mathcal{X} \times \mathcal{A} \rightarrow \{0,1\}$ *is the agents' capability function describing the actions agents are able to perform in a given state of the environment.* $\varphi(a,x,ac) = 1$ *(*$\varphi(a,x,ac) = 0$*) means that agent* a *is able (not able) to perform action* ac *in the state* x*.*

Upon this definition of MAS, two different types of organizational mechanisms may be defined: *informative* and *regulative* (that has two different types, incentive and coercive) (CBHO09). Relationships between these mechanisms are depicted in Figure 2.1.

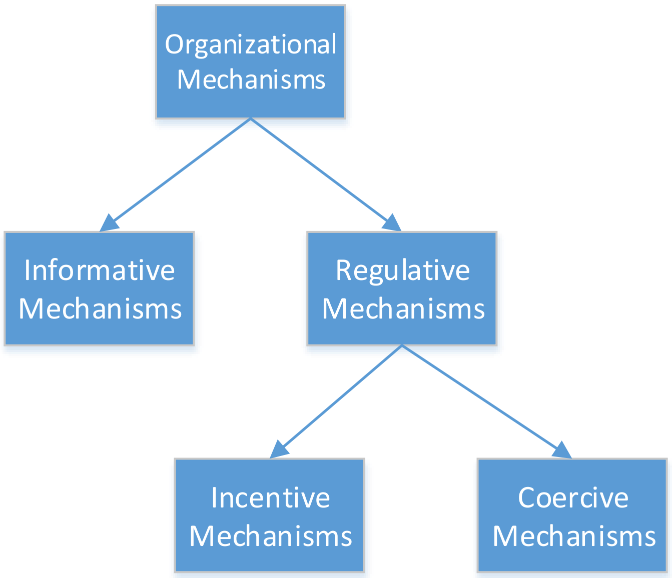

Figure 2.1: *Types of Organizational Mechanisms*

Informative organizational mechanisms (CBHO09) (from now on informative mechanisms) are defined as a function that given a partial description of an internal state of an agent and taking into account the partial view that the mechanism has of the current environmental state, it provides information, which may consist of a set of actions an agent can take but it is possibly not aware of, a recommendation of a particular action which is eventually a good action for the agent, or information about the consequences that a given action may have. Formally it is defined as follows:

$$\Gamma : \mathcal{S}' \times \mathcal{X}' \to \mathcal{I} \tag{2.1}$$

where:

- $\mathcal{S}'$ represents the set of possible partial descriptions of agents' internal states;
- $\mathcal{X}'$ is the set of partial views of environmental states;
- $\mathcal{I}$ represents an information space.

All informative mechanisms have in common that their usage is not imposed. Agents are free to use such mechanisms at their own discretion. To use an informative mechanism, an agent should provide it with part of his internal state. In fact, when rationality of agents is assumed (RG97), agents must use a given informative mechanism if and only if they expect that the usage of the mechanism will be advantageous for them. Informative mechanisms may improve the performance of individual agents and may have effects on the global performance of an OCMAS with respect to a global utility function. The information provided by this type of mechanisms will improve the knowledge of an agent, since the latter includes some extra information for reasoning and thus making him to better choose his future actions.

Regulative organizational mechanisms (CBHO09) (from now on regulative mechanisms) share the same objective as informative mechanisms, but they focus on introducing changes into the environment in order to keep agents from undesired behaviors that drive the system to non-profitable states, that is, these mechanisms are in charge of producing changes in the system so as to reach states that improve the system's global utility. The rationale behind introducing changes in the environment is that agents perceive those changes, so possibly altering their reasoning to decide which action perform next. Such type of mechanisms rely on the existence of a system designer, which defines the preference relation over system states represented through the global utility function, and that has sufficient authority to impose certain changes in the system.

Two types of possible changes in the environment are considered: (i) introduction of incentives in order to make agents follow a desired behavior, and (ii) changes in the agents' action space. Accordingly, two types of regulative mechanisms have been defined (CBHO09):

- An **incentive mechanism** is a function that given a partial description of an environmental state of MAS produces changes in the transition probability distribution of MAS. Formally:

$$\Upsilon_{inc} : \mathcal{X}' \rightarrow [\mathcal{X} \times \mathcal{A}^{|\mathcal{A}g|} \times \mathcal{X} \rightarrow [0..1]] \tag{2.2}$$

- A **coercive mechanism**, Υ_{coe}, for MAS is a function that given a possibly partial description of an environmental state of MAS produces changes in the agents' capability function of MAS, thus adding or deleting actions from the action space of an agent. This function is:

$$\Upsilon_{coe} : \mathcal{X}' \rightarrow [\mathcal{A}g \times \mathcal{X} \times \mathcal{A} \rightarrow \{0, 1\}] \tag{2.3}$$

where $\mathcal{X}'$ represents the set of possible partial descriptions of the environmental states of MAS.

- Any regulative mechanism is either incentive or coercive.

Incentive mechanisms may produce changes in the consequences of agents' actions by introducing rewards and penalties. Obviously, rewards and penalties may produce variations in the expected utility of an agent's actions and, hence, rational agents would change their decisions accordingly (if they know about such incentives). Therefore, agents must be informed about the rules governing the system. In the case of coercive mechanisms the changes in the system are produced through a modification of agents' action space. New actions may be added or existing actions may be eliminated.

Both types of mechanisms emerge as an important contribution to MAS. Since nowadays MAS have progressively become open and heterogeneous, and it is possible that non-collaborative agents populate a system, it is necessary to endow them with mechanisms that help the administrators of the system to keep the MAS under control. Both informative and regulative mechanisms are very useful to afford this task. Aside from this, organizational mechanisms need to be implemented into the environment of OCMAS.

2.1.6 Summary

This section has presented definitions for terms that will be useful during the development of this Thesis. First, a brief description of Organizational-Centered Multi-Agent Systems has been given, which is the computational abstraction that this PhD work is based on. Then, definitions for elements from the business field that are also used in the computer science domain such as virtual organization, adaptation and adaptive organization have been given. Finally, an explanation about a concept that is an important part of our work, organizational mechanisms, have also been given. They are a useful tool for promoting coordination inside an OCMAS. This issue will be handled in this PhD thesis in Chapter 5.

2.2 Historical models of organizational change

Aldrich (Ald99) analyzed the different **historical models of organizational change**, i.e., the different approaches followed by the organization in order to adapt itself. There are three main types of models: (i) life-cycle metaphor models, (ii) non life-cycle models, and (iii) evolutionary models.

The *life-cycle metaphor* is an approach that defines that an organization follows a cycle of emergence, growth, maturity, and decline. Three refined models for the life-cycle metaphor have been defined (KM^{+}80):

- **Developmental model**, when an organization adapts on the basis of the potential inherent to its founding, which grows and decreases during life cycle. That is, an organization must grow at the same time that its available resources and budget grows, thus increasing the complexity of its structure. For example, there are many companies that start as family businesses but as they obtain profits, they hire more employees, need to have bigger offices and to access to a wider variety of resources, etc. However, in case of an economical crisis, profits will decrease, thus being necessary to dismiss employees, etc.

- **Stage model**, when change proceeds in stages (between two stages the organization pauses and there must be a decision about an adaptation process) during which members must solve new problems. The stages of the organizational life cycle are predefined and can be established by the organizational management. In every stage, there must be required to take the necessary organizational actions in order to face the new challenges that managers propose to the organization. For example, in a professional sports team, every season is a stage. After each stage, organizational managers decide whether making organizational changes, depending on the fulfilment of the organizational goals and the new environmental conditions that could have appeared.

- **Metamorphosis model** is similar to the stage model, but changes occur in discontinuous periods of time provoked by a mismatch with context, i.e., the environment changes, thus being necessary to adapt the organization to this new environmental conditions. These changes cannot be planned by the organizational management. For example, a new structure could be required in an organization after a change in the environment happens, for instance, in economical crisis.

The *non life-cycle models* follow the assumption that an organization can achieve progress, i.e., an organization is modified without following a predefined cycle. There are two main refined models for this approach:

- **Teleological model**, in which an organization's purpose drives organizational actions. This model treats change as a cycle of goal formulation, implementation, evaluation, and modification of goals based on what was learned by the entity. This sequence emerges through the purposeful social construction among individuals within the entity.

- **Dialectical model**, in which change is a never-ending shift between confrontation and temporary reconciliation. Conflicts emerge between entities espousing opposing thesis and antithesis that collide to produce a synthesis,

> which in time becomes the thesis for the next cycle of a dialectical progression. A pattern of confrontation and conflict between opposing entities sets a dialectical cycle in motion. Reconciliation is always temporary, never permanent. An example of this model is a government formed by the coalition of two different political parties.

Finally, in the *evolutionary model* an organization is built on previous models, but adding elements of ambiguity and uncertainty. This type of model is not standalone since it can be built together with life cycle and non-life cycle models, allowing much more latitude for choice and chance.

All these models are related to some of the forces that are presented in the following section.

2.3 Forces that drive organizational change

An organizational change is produced by one or some forces that can be differentiated by their nature. *External forces* are referred to the environment where the organization is located, and they are due to elements such as other organizations that populate the same environment (and some of them can suppose competition) or the different heterogeneous agents in the same environment. However, *internal forces* are those that bring about change from inside the organization. Among them, there are forces like the organizational growing or the needs and petitions of agents populating the organization.

Some organizations are more vulnerable than others to the pressure of change, such as organizations with diffuse objectives, uncertain support, unstable values and those that face a declining market for their products and services.

2.3.1 External forces

The external forces are those that promote change inside an organization due to changes in its environment. Among external forces, the following forces can be found: (a) Obtaining resources, (b) Market forces, (c) Generalization, (d) Decay and deterioration, (e) Technological changes, (f) Competence, (g) Demographical features, (h) Laws and regulations, (i) Integration and externalization processes (j) Globalization.

Obtaining resources. An organization requires a set of resources for developing its tasks and processes and to achieve a correct performance (Ald99). A failure when obtaining resources may drive the organization to an adaptation so as to guarantee its survival to improve the way resources are obtained. Solutions can go from extending the organization to a place where resources are easily obtained to reaching an agreement with another organization that has a better access to the required resources.

Market forces. Requirements of products and services of an organization by internal and external agents may change through time, so the number of requests for a product or a service that an organization is offering is not constant. Distribution channels are an important reason for this force to appear because the way an organization markets its products is a key factor for determining if products and services are enough required. Therefore, organizations that offer services or products that nobody is requiring have no reason to exist, so they will disappear if they do not decide to adapt themselves in order to offer new products and services that are currently being demanded (Ald07). This type of adaptation is more frequent in big organizations, which have a better access to different resources. Additionally, there can be changes not only in the requirement of certain products and services, but also in expectations about quality, productivity and customer satisfaction. An organization can build new associations by negotiating with other organizations, in order to create new required products and services that actually fulfil market requirements.

Specialist to generalist: Pressures of an enrollment economy. (Ald07) Some organizations that are unable to acquire enough resources by specializing themselves in a limited range of products or services manage to survive by becoming generalists, i.e. by offering a set of products and services that are oriented to a more general purpose, thus increasing their number of potential customers. By offering a wider range of products and services, an organization is able to appeal the diverse segments of an heterogeneous population and compensate for the low environmental support that it received with its original form.

Decay and deterioration. The decay and deterioration force is active in situations where environmental issues make the organizational goals to change without any notice to the organization. In a normal situation, the set of services of an organization is able to achieve all the goals of the organization. However, when a change in the set of organizational services happens, the organization may not be able to achieve the new organizational goals, so modifications in the set of services would be required so as to comply with the organizational goals.

Technological changes. (BC95) An organization can adopt new technology in order to improve its productivity inside the market where it is developing its activities, and improve the situation of the organization regarding its competitors. This new technology would replace the current one, but a replacement usually implies a cost for the organization. Therefore, organizational managers have to reason and decide about this type of change, measuring the pros and cons of investing on new technology.

Competition. One of the reasons for the organizational change is the existence of organizations with a similar purpose, turning into competition for them

(BC95). That is the reason why some organizations must change their objectives to improve their condition. This kind of change would mean a reaction to a given event. An organization could also change its objectives and/or services and products that is offering in a preventive way to avoid competitive environments. In order to check whether an environment is competitive it is necessary to check the organizational density of the environment where the organization is located, i.e. the number of organizations located in the close environment that pursue similar objectives than those pursued by a given organization. In different scenarios, an organization will decide to merge with one or some organizations in order to get a better situation to compete with other organizations.

Demographical features. Since organizations are open systems, agents populating them and their environment are heterogeneous. An organization must control this diversity in an effective way, paying attention to the different needs of these agents, but trying to avoid malicious and/or interested behaviors by them (MTB$^+$02). For instance, by assigning determined roles to these malicious agents, which can limit their actions.

Laws and regulations. There can be external laws that might affect the environment of an organization or its neighbors organizations (BC95). Additionally, structural elements from an organization such as topology, norms and agents can be affected by the effects of these external laws. Moreover, the society and the environment can persuade an organization in order to change its behavior by pressuring it with the aim of modifying its objectives, products and services. Additionally, another element to take into account is the national culture. For example, Italy and France usually prefer high formalization and centralization, others like India, low formalization but still centralized, or Germany, high formalization, but decentralization.

Integration and externalization processes. This force refers to three different types of processes: (i) mergers and acquisitions (LF99); (ii) strategic alliances and partnerships (MOS96); and (iii) outsourcing (Qui13) and spin-offs or organizational splits and joint ventures. The first one refers to the merging of two or more organizations, or the acquisition of one organization by another, leading to bigger organizations where their structure and members should be reorganized. Merging will allow to compete from a better position with other organizations, providing products and services that could not be provided if they were separate. The second is similar, but does not include merging of actual organizations. Still objectives and strategy of partners has to be taken into account in making business decisions. The third one refers to organizations that look for opportunities in their environment and create a spin-off company to deal with the situation, make an organizational split, or start a joint venture with a partner organization.

Globalization. This term refers to the increasing unification of the world's economic order through reduction of such barriers to international trade as tariffs, export fees, and import quotas (Rob92). The goal is to increase material wealth, goods, and services through an international division of labor by efficiencies catalyzed by international relations, specialization and competition. Globalization describes the process by which regional economies, societies, and cultures have become integrated through communication, transportation, and trade. Therefore, an integration of an organization inside an economically globalized market can force the organization to change, in order to be adapted to the needs and requirements of a globalized market.

2.3.2 Internal forces

The internal forces of an organization are signals produced inside an organization, indicating that a change is necessary. Thus, it is important to clearly define these forces, in order to monitor them and to achieve this adaptation process in the most appropriate form and moment. The internal forces are: (a) Growth, (b) Power and political factors, (c) Goal succession, (d) Life-cycle, (e) Human resources, (f) Decisions and managers behavior, (g) Economical restrictions, and (h) Crisis.

Growth. When an organization grows in either members or budget, it is necessary to change its structure to a more hierarchical organization, with higher levels of bureaucratization and differentiation among its members (Ald07). New subunits will appear, with members carrying on very specialized tasks, and differentiating between agents dedicated to production from management agents. This force is related to the *developmental model* (explained in section 2.2) since it states that as an organization grows, certain structural transformations should occur. The solution to this force may also be seen as splitting the organization. One organization will give birth to two or more organizations, each of them making differentiated tasks.

Power and political factors. The most powerful members of an organization may have different objectives than agents in a lower hierarchical level, which can be even different from the organizational objectives. The organization may assure (for instance, by means of observers) that manager agents do not impose their objectives above organizational objectives (Ald07).

Goal succession. There are certain organizations that disappear after achieving its strategic objectives. However, some other organizations look for a new strategy. Therefore, these organizations will continue with their existence. It may seem strange to think about an organization that fulfills its objectives and then changes them in order to continue its life cycle. However, there are organizations that are created with such specific strategy, so after achieving these goals, its strategy, tactics and goals could be changed to others that are similar to the former ones, trying to take profit of the capabilities and abilities of the organizational members (Ald07). An organization that follows this force is one that has a non-life cycle organizational change model like the *teleological model* (explained in section 2.2), where changes in the organizational strategy drive the organizational actions to be modified. Organizations that follow this model are focused on strategic objectives. In some cases, these organizations follow a *Management by objectives* (MBO) (Odi65) approach, where the organizational strategy is discussed by both managers and subordinates, thus increasing the motivation of subordinates.

Life-cycle. Some existing organizations follow the classical *life-cycle model*, explained in section 2.2. Thus, they appear, grow, change, and disappear, to give way to other organizations (BC95). Carley and Svodoba (CS96) state that, as the organization matures, the number of changes decreases. That is, over time, organizations become more staid and less likely to adopt changes that do not have obvious foreseeable benefits; they become locked into a certain way of doing business.

The life-cycle metaphor is connected to the *structural inertia theory* (BC95) which describes that organizations reach stability once the procedures, roles and structures that compose them are stabilized, i.e. they are in an *inertia momentum*. To check whether an organization is inside an inertia momentum, it is enhanced with an *'inertia clock'*. This mechanism takes into account the moment when the organization reaches an stability state. The inertia clock is reset when a change in one of the above mentioned elements occurs. Regarding what kind of organizations are more prone to adapt, some experts point out that big organizations are not likely to adapt due to its bureaucratized structure, while some other experts state that bigger organizations are more likely to adapt since they have an easier access to resources (Car97).

Human resources: Problems and perspectives. An organization must be aware of the needs of their agents, trying to achieve their satisfaction with the tasks they are developing. Managers of the organization must control that their agents are committed with the organization, present an adequate behavior and their performance is acceptable. If any of these factors is not achieved, it could be necessary a change in the organization, for example by modifying the reward system or the working conditions associated to these employees. In general, dissatisfaction of agents is a gauge for the organizational change.

Decisions and managers' behavior. Industrial disputes between agents and their supervisors inside organizations are an important force for change. If a subordinated agent disagrees with his/her supervisor, he/she could ask for new tasks to develop inside the organization. If the management approves his/her petition, an action must be carried out, for example changing the supervisor of this subordinated employee. Moreover, an unfair reward system imposed by organizational leaders will bring about undesired reactions and changes in subordinated agents, which will modify their behavior, thus decreasing their productivity.

The forces *power and political factors*, *decisions and managers behavior*, and *human resources* are following one of the non-life cycle organizational change models, the *dialectical model*, where change is a never-ending shift between confrontation and temporary reconciliation between agents populating the organization.

Economical restrictions. Organizations want to maximize their performance and utility, but they are required to comply with the available budget of the organization. Therefore, in the case that the cost produced by the organization is higher than the budget, then a reorganization is necessary to reduce such cost.

Crisis. If an organization is in a crisis due to a sudden drop of its efficiency, a possible solution is a deep organizational change, modifying structural and/or functional elements, depending on the specific needs of the organization.

2.4 Types of organizational change

This section depicts the different types of changes that can be carried out inside an organization in a general way, from both human and agent organizations. There are two main approaches to classify these changes, depending on the point of view we are studying: (i) from a temporal point of view, which classifies changes regarding the temporal manner in which they are applied, and (ii) from a structural point of view, classifying changes between those that modify the structure of the system, and those that only change behavioral (and non-structural) entities of the organization.

2.4.1 Types of change from a temporal point of view

Depending on the way that organizational changes are produced through time, the Organizational Theory classifies these changes in three different categories (Kri01): (i) adaptive, (ii) evolutionary, and (iii) revolutionary.

2.4.1.1 Adaptive changes

Organizations are considered to be open systems whose environment is able to interact with them, also forcing them to respond to external pressures. Thus, external forces are responsible for driving an organization to carry out adaptive changes. Norms, structure and technology of the organization drive it to a predictable behavior that can delay the answer of the organization to a modification in the environment. This habit could lead the organization to a failure.

Environmental factors that could be altered are general or specific. General factors can be technological, legal, political, economical, demographical, or cultural. A modification on any of these factors affects relationships between members of the organization, or between organizations.

Specific factors are related to the creativity of the members of the organization, as well as the need for overcoming the rejection of introduction of new techniques. A common example occurs when an organization modifies its functional structure (e.g., from a pyramid structure to a matrix structure) in response to market forces or the pressure from other organizations, due to competence or law restrictions. Currently, globalization is one of the most powerful external forces that provokes adaptive changes in an organization.

2.4.1.2 Evolutionary changes

Evolutionary changes imply an incremental modification of the parameters that measure organizational performance. They are suitable to maintain and reactivate the cycle of a product, without innovating and changing the paradigm the organization follows. Evolutionary changes are continuous and incremental. They are applied following an evolutionary organizational change model. When an organization introduces an evolutionary change, the organization basically maintains its previous design, but adds elements of ambiguity and uncertainty to its structure and entities.

This type of change starts from these assumptions:

- An organization is normally in stability.
- Each certain time, these stability periods are interrupted by a transformation process, requiring to make an adjustment.
- The key point of strategic management is to keep stability or at least an adaptable strategic change. However, periodically recognizing the need of changing and being able to manage this process through time is required.

2.4.1.3 Revolutionary changes

Revolution is characterized by certain features that distinguishes it with clarity from other kind of changes. First, it is characterized by considerable and discontinuous modifications in the shape, structure and nature of the organization, not by progressive adaptations or improvements of the current situation. Revolutionary changes include the introduction of a big scale change that implies important modifications in technological processes, in the structure of the organization or in the behavior and abilities of all its agents.

These changes are deep and general, rather than superficial and restricted, affecting to the whole organization in almost all its levels. Decentralization, reductions, and spatial relocation of functions and activities exemplify the changes that modify structural relationships in a wide and deep way.

Transformations require significantly different (or totally new) actions carried out by the members of the organization. In this case, organizational changes are characterized by the fact that the whole organization must develop actions that are radically different from current actions.

Transformations start further from the current organization, in the sense that they are related to environmental factors, including feedback for the mission, strategy and structure of the organization. These factors demand the reformulation of the culture and the behavior of the complete organization.

2.4.2 Types of change from a structural point of view

When executing an adaptation process in an OCMAS, two types of change can be distinguished: dynamical (behavioral) and structural (Dig09). Dynamical changes are those in which the structure of the system remains fixed, while agents and aspects like role enactment are modified. Structural changes are produced in structural elements of the system like roles, topology or norms.

Regarding dynamical changes, there are two types that must be considered:

- *Agent fluctuation.* Changes regarding agents joining or leaving the system. On the one hand, when a new agent joins the system, it is necessary to reach an agreement to join the organization, playing a particular role that indicates the rights and duties of the agent that plays that role. On the other hand, each time an agent leaves the system, it is necessary to determine if this operation is possible, taking into account certain imposed conditions by the MAS management. Sometimes, it could not be appropriate to allow an agent playing a specific role to leave the system. In other moments, it may be convenient to reassign that role as soon as the agent leaves the system.

- *Instantiation of the interaction pattern.* A change of this kind consists of two agents that carry out a certain interaction pattern and reach an agreement to follow a protocol adjusted to this interaction pattern. In this kind of changes there are included, for example, changes related to the role enactment process, changes in the agents that are providing a service or in the set of active norms, etc. These changes force agents to modify their interaction pattern.

Regarding structural changes, there are two ways to carry out a structural change in an organization (Dig09):

- Self-organization: implies the emergency of changes, appeared because of the interaction between agents in a local level, that generates global-level changes in the organization.

- Reorganization: designed societies are adapted to modifications in the environment by adding, deleting or modifying their structural elements (roles, dependencies, norms, ontologies, communication primitives, etc.).

Self-organization changes are bottom-up, where an adaptation in the individual behavior of the agents will lead to a change in the organization in an emergent way. Thus, self-organization is an endogenous process (carried out by the agents). Agents are not aware of the organization as a whole, they only work with local-level information to adapt the system to environmental pressures by indirectly modifying the organization. Therefore, agents, using local interactions and propagation, modify the configuration of the system (topology, neighbors, influences, differentiation). In section 2.5, a deeper description of self-organization is provided.

Reorganization is a top-down approach so that a modification in an organizational aspect will produce changes in agents composing the organization. Reorganization can be both an endogenous or an exogenous (controlled by the user or by an external system) process, referred to systems where the organization is explicitly modified through specifications, restrictions or other methods, in order to ensure a suitable global behavior when the organization is not appropriate. Agents are aware of the state of the organization and its structure, being able to manipulate primitives to modify their social environment. This process can be initialized by an external entity or by the agents, directly reasoning over the organization (roles, organizational specification), and the cooperation patterns (dependencies, commitments, powers). A more complete description of reorganization is found in section 2.6.

2.5 Self-organization

There are many definitions, approaches, frameworks and methods of the concept of self-organization. This section focuses on reviewing some of the most interesting ones, in order to provide an overview on the field of self-adaptive MAS.

Different authors have enumerated the features that a system must be provided with in order to be considered as a self-organization system.

2.5.1 Self-organization concept

On the one hand, De Wolf *et al.* (DWH05) establish that a good requirement analysis must be made prior to decide whether using a self-organized system or not. According to the authors, these systems must be capable of maintaining some properties at a macroscopic level during all the system execution, as well as to allow that these properties must be measurable in order to develop an evaluation of the behavior of the system. They also establish that a self-organizing MAS must fulfill the following properties:

- Available information is located decentralized or distributed, thus needing only local interactions to get this information.
- The system must be capable of dealing with dynamical aspects, being necessary to be robust, to keep the macroscopical behavior without changes, and having flexibility and adaptability in order to carry out local reconfigurations in the system.

Robustness is preferred over optimality, due to the fact that optimality can be only achieved when the system remains static. De Wolf also describes that self-organizing, emergent MAS promise to be scalable, robust, stable, efficient and with low latency (Ant04), but they behave in a non-deterministic way, although they have tendencies that are predictable after checking the evolution of the macroscopic behavior after a series of system executions.

On the other hand, Calta (Cal09) states that self-organizing systems must present the following features:

- **Structure**. An organized system should have a structure. Having a structure restricts the degrees of freedom of the system components, minimizing the possibilities of obtaining non-desired behaviors inside the system.
- **Acceptability**. If the system structure is affected by any event, the system must be able to adapt itself to a new structure.
- **Decentralization**. There is no centralized element that controls self-organization, so its elements are autonomous.
- **Interaction and local knowledge**. Organizational components do not have a global view of the system. Each component has only a vision of itself and of its nearest environment.
- **Homogeneity**. System elements have the same abilities. If an element with different abilities exists (for instance, a leader), these higher abilities would appear due to the system's self-organizing properties.
- **Non-deterministic system's behavior**. The goal of a self-organizing system is to achieve a desired behavior at a global level. Steps to be followed to reach this behavior are not known in advance, and different combinations of actions can be given in order to reach the desired global state.

Both De Wolf and Calta agree on adaptability, decentralization, locality and non-determinism factors, being able to consider these as the properties that a MAS must show in order to be considered as a self-organizing system.

Kota (KGJ09) presents another definition of self-organization: 'mechanism or process that allows the system to change its organization without the need of an external and explicit action'. Thus, self-organization can consist on building an organization from disordered entities or by reordering an existing organization. Moreover, Kota presents a framework that consists on a simple organizational model, where the adaptation process is carried out by all the agents to improve the system behavior. Modifications are carried out at an inter-agent level, by modifying intra-agent interactions, and it does not bring any modification in the agent's internal behavior.

2.5.2 Self-organization MAS approaches

One of the most influential concepts for self-organization studies is *stigmergy* (Gra59). This concept was firstly introduced by Glassé at the end of the 50s while he was trying to study the behavior of social insects and it was defined as a type of mechanisms that mediate in interactions between animals, which is essential to get emergent ways of behavior coordinated at social level. Originally, the concept of stigmergy was used to explain the so called *coordination paradox* between individual and social levels: that is, although a group of insects seems to behave in an organized and coordinate way at a global level, when we notice on the behavior of a single individual, it can be checked that he behaves as if he was alone, without interacting with other insects or being evolved in any collective behavior. The explanation on this phenomenon is that insects interact in an indirect way, for instance using pheromones, like ants that want to build a path to be followed by the rest of their community. Therefore, insects change their behavior according to the environment. Additionally, for carrying out changes over it that drive to a global change, insects are provided with tools such as pheromones.

In the field of computer science, and more specifically, in the field of MAS, stigmergy has been used as a technique to solve complex problems, but in the last years it is becoming into an approach to design and develop systems. In a general way, the use of stigmergy in MAS is focused on trying to imitate the behavior of ants, without taking into account that agents are cognitive entities (unlike ants) and that environments where they exist are more complex and have a bigger functionality than being a simple pheromone container.

For this reason, different researchers have started to introduce cognitive elements when studying stigmergy, for example researching stigmergy in human activities (VDP06). An interesting work on this subject, where cognitive stigmergy is defined, was developed by Ricci *et al.* (ROV^{+}07). In their work, agents with cognitive capacities are considered, which can be rational, heterogeneous, and with capacity of adaptation and learning. One of the important elements of this work is the environment, which is more complex, and where agents know: (i) their field of work, (ii) that they share the environment with other agents, and (iii) the functionalities and opportunities of the environment that they can explode to reach their objectives. Therefore, the environment is no longer seen as a container, but it becomes a first order entity that contains tools, which can be used in both individual and social levels. These tools can be implemented by means of artifacts, explained in detail in Chapter 5, Section 5.2.

Gardelli *et al.* (GVCO08) also use artifacts as a tool to introduce self-organization inside a MAS. In order to achieve this, the concept of environmental agents is introduced in the system. They are in charge of controlling the artifacts. The use of artifacts is carried out through a management interface, usually closed to the user agents that populate the system. These agents are not able to perceive the existence of environmental agents, but they perceive and are able to access the artifacts.

In the model presented by Kota *et al.* (KGJ09), the agent organization comprises a group of problem solving, cooperative agents situated in a task environment. By problem solving, we mean agents that receive certain input tasks to achieve, execute these tasks and return the outcomes. Correspondingly, the task environment presents a continuous dynamic stream of tasks that have to be performed. In more detail, the tasks are modeled as workflows composed of a set of service instances. The organization consists of a set of agents that provide these services. Agents must establish relationships between them, in order to make pairs that are able to solve a problem by means of a service. Therefore, using this self-organization method, a pair of agents estimate the utility of changing their relation and take the appropriate action accordingly. This adaptation method redirects agent interactions (via the organization structure) and does not entail any modification to the agents themselves or their internal characteristics.

ADELFE (BGPP02) is a methodology that follows RUP (Kru04) (Rational Unified Process) conventions, but oriented to the software engineering based on adaptive MAS (AMAS), defined by means of the AMAS theory (GCG99). The AMAS theory focuses on achieving the accomplishment of a task by means of merging the behavior of different agents, being more important the group rather than the individual. By the employment of this theory, the global function of the system must not be coded inside the agents, but it emerges from the collective behavior of the different agents that compose the society. Every agent has the capability of changing his interactions with other agents depending on the individual activity to be solved. This self-organization capability that agents have is based on the capability that an agent has to be 'cooperative', which does not mean to always help the rest of agents, but being able to recognize cooperation faults named 'Non-Cooperative Situations' (NCS), and how to deal with them. An agent is able to detect three different types of NCS: (i) when a perceived signal from its environment is not understood without ambiguity; (ii) when the perceived information does not induce an agent to an activity process; and (iii) when the results drive an agent to act in the environment in an useless way.

Since an adaptive MAS is not a technical solution for every application, ADELFE methodology provides a tool to help the designer whether the AMAS theory is appropriate to develop his application or not.

Adaptation will be controlled by cooperative agents. An agent that locally detects cooperative faults changes its interaction with the rest of entities by eliminating its state. The NCS of an agent must be described by a designer, and it will be application dependant. ADELFE provides the designer with generic cooperation failures such as incomprehension, ambiguity, or conflict.

Tropos4AS methodology (MPP08) also adopts self-organization concepts. It is an extension of Tropos (BGG^{+}04), which is a general purpose Agent-Oriented Software Engineering (AOSE) methodology that gives support to environmental changes and failure prevention, enabling to model a process that should be able to detect the symptoms that will cause a fault in the system.

On the one hand, regarding environmental changes, different types of goals are considered in Tropos4AS, such as goals that have to be satisfied only in certain situations (or only once), and goals that simply give the means to perform specific activities without having a particular achievement condition. Tropos4AS also supports the modeling of environmental entities. Then, conditions are added in order to relate the state of an environmental entity to the state of a goal. The use of different types of conditions allows guiding the goal achievement process, triggering transitions from a goal state to another, and storing transitions between goal states. On the other hand, regarding failure prevention, in Tropos4AS it is necessary to define a process that should be able to deal with error recovery issues.

There is also an extension that integrates elements from ADELFE to Tropos4AS. In (MMG+09), bottom-up ADELFE cooperation rules are added into Tropos4AS, giving the runtime agent instances the knowledge for selection of their peers and cooperation with them, and thus achieve an emergent self-organizing behavior to adapt to a changing environment. Additionally, metamodel elements such as knowledge, or modeling steps such as the identification of Non-Cooperative Situations, are added into Tropos4AS.

MACODO (HWHJ09) is another framework that manages self-organization in MAS. It is a middleware that offers the life-cycle management of dynamic organizations as a reusable service separated from the agents. MACODO is provided with a model where concepts such as roles are defined, also establishing the concepts of RolePosition (similar to a job offer in the business field) and RoleContract (an agreement between an agent and an organization regarding a specific RolePosition), to control the access to an available role; agents (including their context, built by the current state of the local environment of the agent and his location) and organizations. In order for an agent to enter an organization, he should create his own organization, and then he must merge both organizations, being the agent integrated inside the original organization. To leave the organization, the agent must end all the active contracts that he has in the organization. MACODO is also able to cope with the organizational dynamics, including changes in the context of the system or in the set of agents of the system, additionally having rules to keep the right system performance. There are two types of rules. The first ones control the *intra-organizational dynamics*, which describe how an agent can join or leave the organization in a dynamical way, an important concept when referring to open organizations. The second ones refer to *inter-organizational dynamics*, including the organizational splitting and merging.

INGENIAS (PGSF05) is another methodology that introduced self-organization in a MAS metamodel (SP08). The self-organization model proposed by the authors is founded on the ability of the agents according to some reinforcement learning. This reinforcement comes from a positive or negative feedback from agent interactions with other agents, as a property of self-organizing systems. Agents form an interaction network called social network that represents the experience of a particular agent or the positive or negative impact an agent can have over another agent during their interaction. INGENIAS metamodel was then extended (SP08) by adding entities like the motivation of a role for performing a task, and the social network entity. Finally, authors state that not all the elements from the reinforcement mechanism are part of the metamodel, leading to ambiguities at design time. Also, this mechanism is tedious when defining complex systems.

De Wolf and Holvoet (DWH05) propose a modification for the Unified Process (UP) (Kru04) in order to give support to self-organizing MAS. In the analysis step, it must be decided whether is convenient to use a self-organizing MAS or not. This decision takes into account that the developer may need a certain autonomous behavior, the information must be distributed (only local interactions are possible), the system is required to be robust and the system will change through different and unpredictable scenarios.

In the design step, there are different strategies for achieving macroscopic properties that arise solely from local interactions between agents: (i) general guidelines, or ‘design principles’, (ii) reference architectures, and (iii) decentralized mechanisms that allow coordination, like pheromones and co-fields. The implementation step can be carried out by any of the tools that give support to this kind of systems.

2.6 Reorganization

Reorganization is a centralized process in which changes start on the organizational structure, differently from self-organization, which focuses on changes coming from agents to the organizational structure. Therefore, changes start in the topology, norms, or rules of the organization, and then these modifications are spread to agents, which must adapt themselves to the changing conditions of the organization.

When following this approach, it will be interesting to cope with structural adaptation changes. More specifically, this type of changes include modifications in the organizational specification, the way that the organization is built and the role allocation problem. Therefore, we should take into account changes in the shape that organizational units are structured, as well as changes in roles, norms, organizational goals, and in other common elements of virtual organizations. The addition, deletion or modification of any of these elements will provoke an adaptation in the organization, and so in the agents that populate it.

According to Picard *et al.* (PHBG09) there must be taken into account changes in the structure of the organization and in the role allocation of agents, as well as in the way that the OCMAS is built. Reorganization can consist on the generation and composition of agents in the system to reallocate their given roles. The reason for change would be to achieve that the structure is suitable adapted to the environment and to the tasks that agents must fulfil. According to Glaser (GM97) the organization adapts when a new agent joins the system playing a particular role, and this agent is only accepted inside if it improves the system's utility function.

Next subsections describe the requirements and reasons for a system to adapt itself; how the reorganization process is carried out; and finally, different examples about reorganization in MAS are presented.

2.6.1 Necessity of reorganization

Following temporal criteria, Picard *et al.* (PHBG09) define that the decision on when to start the reorganization process can be static or dynamic. In the first case, the process starts due to a beforehand defined criterium. In the second case, reorganization is triggered as a consequence of the system's performance. If agents do not achieve some specific objectives, the organization must adapt itself. The time when reorganization is carried out is subject to the utility of the organization and the agent.

The entity responsible for ordering a reorganization process in the system can be an external agent (CZ10) or an actor from the system (GJW08). This actor can be an agent, a group of agents of the system, or a human. Actors responsible for a reorganization process must have a special role inside the organization, for instance called 'system adapter' or 'change manager' that denote them as responsible for making changes in structural elements of the organization.

As seen in section 2.1.3, adaptation in a MAS appears when its conditions, requirements, or the environment change (PHBG09). Conditions and requirements (defined by the participant stakeholders (APDD08)) of the organization are reflected in the organizational goals. If it is detected that the organization is not currently achieving these organizational goals, some of its parameters would change. For example, topology, norms, or roles.

Another issue for deciding about a reorganization process are changes in the environment of the system (APDD08). It is important to take into account the events that happen in the environment, since they will affect the organization. That is the reason why other organizations that populate the environment must be controlled. A change in any of their features can affect the conditions of our organization. For example, other organizations can decide to send a big number of agents to our organization or they can decide to reach an agreement with an organization to get a particular goal. It is also necessary to check the behavior of agents that populate the environment (without being associated to any organization), i.e., whether they join our organization or not, thus being possible to anticipate a reorganization.

Other environmental elements, such as the available resources and services, also play a key role when deciding about the need of adapting our organization. If a resource or service that was provided becomes unavailable, it is possible that the system requires a change in order to afford this losing. A reorganization will be also necessary if the environment is providing new resources and/or services that cover system needs or if these external resources and services are 'cheaper' to obtain than the ones produced by our own system.

Changes in requirements and conditions, both internal and external to the system, will produce the need for adapting the functionality of the system, being necessary an adaptation process in order to get the system capable of offering the desired functionality (APDD08).

2.6.2 Reorganization process

The generic process of reorganization can be split in two steps (PHBG09): monitoring and repairing. This second step can also be split in the phases of design, selection and execution, or just in selection and execution. The monitoring phase is based on finding problems in the system, the organization and its environment. It is recommended to define a set of non-adaptation situations, in order to allow the system to act upon them once they are detected. Therefore, it is necessary to design a set of alternatives to carry out in the organization. It is also required to define an evaluation process for deciding the most adequate solution, and finally it is necessary to execute this solution.

This is the general process to apply adaptation in a MAS, but there are other ways for reorganizing a system:

- **Predefined change in the organization** (PHBG09): this type of changes is previously planned to occur in a specific moment. Monitoring is carried out by agents or an external entity; design phase is previously executed in MAS design time and the phases of selection and execution are immediately performed when the moment comes.
- **Controlled organizational change** (PHBG09): the system does not know when the organization will change, but it knows the conditions where change will be developed under them. It is necessary to define strategies to monitor the organization in both an endogenous way (by the own internal agents) or exogenous way (by the user or an external system). It must be known how to identify that the global objective cannot be achieved, and what part of the organization requires a change. Once identified the problem, it is necessary to define a set of alternatives, which can be given in a predefined set, or they can be designed under user request, so we must face the problem of the complexity when having such a big space solution to search inside.
- **Change directed by the system adapter** (HJST07): In this case, neither the system designer, nor the agents will previously know the conditions and the moment to make a change in a virtual organization. Therefore, it is not necessary for these agents to make a system monitoring. It will be the system adapter who orders when to start the process of change inside the system, being responsible for knowing the conditions and the moment for a change, besides monitoring the system. Once the order for reorganization has been given, the system adapter (also known as change manager) will provide the keys for a change, that can be from one of these types:

- The system adapter decides in an univocal way the change that must be carried out in the organization. Agents in the organization do not have the option of refuting this change.
- The system adapter gives different options for change, being necessary a selection step to decide the most beneficial option for the organization.
- The system adapter only provides a set of guidelines, advices, etc. that the organization should follow, and there will be organizational agents, responsible for designing the set of alternatives, making use of system adapter's guidelines. Then, the system adapter should select and apply the most appropriate solution.

2.6.3 Reorganization examples

In this section, the most relevant works of reorganization in the OCMAS field of research are described. More specifically, works carried out by Hoogendoorn *et al.*, Dignum and Dignum, plus the ALIVE and the MOISE+ frameworks. All of them constitute the direct background of the work we intend to develop within this thesis. They are first presented, then a discussion section is included.

2.6.3.1 Model for organizational change

Hoogendoorn et al. (HJST07) presented a model for organizational change based on the AGR methodology (FGM03). The three primary entities of this organizational model are agents, groups, and roles.

- Agents: An agent is only specified as an active communicating entity which plays roles within groups.
- Groups: They are an atomic set of agent aggregations. Each agent is part of one or more groups. These groups are able to overlap.

- Role: A role is an abstract representation of an agent function, service or identification within a group. Each agent can handle multiple roles, and each role handled by an agent is local to a group.

Additionally, this model is able to represent interactions between roles inside a group and interactions between different groups. All these elements are represented by the Structural Language (SL). Moreover, to represent the behavior of the organization, the Temporal Trace Language (TTL) is used. With an associated ontology, the temporal trace language can be employed to specify behavioral properties at different aggregation levels, according to the organizational structure.

In order to develop his organizational change model, Hoogendoorn takes inspiration from the ideas of Lewin (Lew51), who states that a change in an organization is provoked by two opposing forces: forces that resist the change, and forces that drive towards the newly desired organization. Taking these forces into account, an organization changes following a three-phase process. First, the driving forces are stronger than forces that resist changes, thus entering in the unfreezing phase. Second, the organization enters into the movement phase, where all the needed and/or planned changes for the organization are carried out. Finally, after all changes have been deployed, the system returns to a state of equilibria in the refreezing phase, where resistant forces are stronger than driving forces.

In order to model these forces, they are associated to new roles in an organization. Also, to control organizational change, a new role named Change Manager, provided with driving forces, can be added to the organization. An agent playing this role has to deal with agents playing the role Member, which will be provided with driving and resistant forces, depending on their personal opinions about change. The Change Manager is able to carry out changes in the whole organization or only in a local level.

This framework represents a good approach for reorganization, taking into account concepts from Organization Theory such as forces that drive organizational change. Also, it introduces the concept of Change Manager, an organization aware agent that manages modifications inside an organization. However, its model is not able to explicitly represent other organizational concepts like global goals or services.

2.6.3.2 ALIVE

The ALIVE project (APDD08) is a proposal that is aimed to create a framework for software and service engineering, based on combinations of coordination and organization mechanisms (providing a flexible, high-level means to model the structure of interactions between services in the environment) and Model Driven Design (providing automated transformations from models into multiple platforms). Although the main goal of the ALIVE project is to enhance the development and deployment of service-based systems, this approach can also be applied to the deployment of regulations in human organizations.

Systems developed within ALIVE project are provided with three layers:

- The Service layer augments and extends existing service models with semantic descriptions to make components aware of their social context and of the rules of engagement with other web services.
- The Coordination layer provides the means to specify, at a high level, the patterns of interaction between services, using a variety of powerful coordination techniques.
- The Organization layer provides context for the other levels, specifying the organizational rules that govern interaction and using recent developments

in organizational dynamics to allow the structural adaptation of distributed systems over time.

ALIVE is aimed to introduce mechanisms to cope with organizational change. The main sources of change in the context of an organization are the stakeholder needs, the environment conditions, and the system functionalities.

Current methods do not effectively map onto more open, service-based environments, taking into account not only the properties of individual applications, but also the objectives, structure and dynamics of the system as a whole. Hence, the aim of the ALIVE project is to change this situation and focus on bringing together the leading edge methods from Coordination Technology and Organizational Theory, together with leading edge new approaches to software design.

The applications developed in ALIVE are based on the notion of software services performed by agents having organization awareness and coordination capabilities. In that way the system can be dynamically reconfigured and adapted. The development process described in the ALIVE methodology encompasses various stages of the life-cycle of a MAS, including its design, implementation, deployment, validation and execution.

2.6.3.3 Formal semantics framework

Dignum *et al.* (DDF$^+$05) designed a formal semantics framework to represent an agent organization. This proposal is able to represent an organization from a static point of view, but the authors state that an organization must maintain a good fit with the environment, so they modified this proposal (DDF$^+$05) in order to manage organizational adaptation, considering the need to reorganize or the consequences of change on the organization's effectiveness and possible efficiency. Also, organizations are active entities, capable not only of adapting to the environment but also of changing that environment.

This approach treats adaptation as a design issue that requires an action resulting in the modification of some organizational features, whereas other approaches (e.g., ADELFE) are closer to the emergence of organizational patterns. Changes are represented as transitions between two different worlds, where a change in some propositions in a world can result in a different world. Reorganization consists of two activities. Firstly, the formal representation and evaluation of the current organizational state and its 'distance' to the desired state, and, secondly, the formalization of reorganization strategies, that is, the purposeful change of organizational constituents (structure, agent population, objectives) in order to make a path to the possible and efficient desired state.

Therefore, the organization must be monitored since the environment evolves, thus varying performance. In this approach, a value for the organizational performance is established as a function on the environment, agents and organizational capability, and on the desired state of affairs. Therefore, this function establishes the cost of achievement of a certain state of affairs, given the current state and group of agents. A possible strategy to decide when to reorganize says that if the cost of reorganization plus the cost of achieving the new state by the reorganized organization is less than the cost of achieving this state without reorganizing, then reorganization should be chosen.

After monitoring the organization, a change might be necessary. Reorganization activities can be classified in three groups:

- Staffing: Changes on the set of agents, like adding new agents or deleting agents from the set. This corresponds to human resources activities in human organizations.
- Structuring: Changes on the ordering structure of the organization. This corresponds to infrastructural changes in human organizations.

- Strategy: Changes on the objectives of the organization, like adding or deleting desired states. This corresponds to strategic (or second-order) changes in human organizations: on the mission, vision, or charter of the organization.

2.6.3.4 MOISE

MOISE (*Model of Organization for multI-agent SystEms*) (HBSS00) is a normative and functional modeling language, extended in MOISE+ (HSB02b) and MOISE^{Inst} (GBKD05). MOISE looks for identifying the rights and duties of the agents inside a society from four points of view: structural, functional, contextual, and normative.

- The *structural specification* allows discovering the organization based on roles, groups, and links between roles and groups. A set of restrictions expresses compatibilities between roles, reach of links and cardinality of roles and groups. A role represents restrictions over functionality, goals, plans, and relationships between roles, which an agent must follow when he is playing a role in a specific group. A group consists of a set of roles and links, also determining cardinality restrictions, inheritance relations and compatibility between roles. Additionally, subgroup specification is allowed. Finally, social links or relationships can be from one out of these three types: *acquaintance*, which indicates the agents that have a formal representation of other agents; *communication*, which specifies agents that have permission for communicating with other agents; and *authority*, expressing which agents have control over other agents.

- The *functional specification* expresses the global functioning of the system as a set of social schemes. A social scheme is composed of plans that are linked in order to look for collective goals. Goals are grouped in missions that agents must achieve. Agents that focus on achieving a specific mission must achieve all the objectives of this mission. Social schemes can be reused with other social schemes.

- The *contextual specification* describes a priori a set of contexts populated by the corresponding organizational entities with the transitions that govern the change of context. Each context specifies a state in which agents that play certain roles should respect determined roles.

- The *normative specification* allows to limit the behavior of the agents by means of norms. A norm defines a right or a duty for a role or a group for executing a mission in a particular context during a given time, supervised by an agent playing the role issuer, who is able to apply a sanction if the norm is violated.

MOISE (HBSS00), and its extensions MOISE+ (HSB02b) and MOISE^{Inst} (GBKD05), are well-known methodologies for designing an OCMAS. Agents inside MOISE+ designed systems are organized following groups. When a reorganization process starts, a set of roles (the **reorganization group**) is created in order to carry out with this process.

After assigning the roles from the reorganization group to agents, a **reorganization scheme** is created, featuring all the goals to follow during the adaptation process. These goals cover the complete adaptation process: monitoring, design, selection, and implementation.

MOISE+ represents a good approach to organizational change, where new roles join the system to carry out the adaptation process.

2.6.3.5 MTDM

The Multi-dimensional Transition Deliberation Mechanism (MTDM) (AJGF12) is a framework that calculates transitions of the current organization to other organizational structures with high expected utility based on the cost for transitioning to these structures. The benefits and costs of transitions are measured in terms of Organization Transition Impacts. Then, the MTDM decides which transformation is finally implemented and provides the sequence of changes required to carry out the transition.

The MTDM provides an accurate estimation of the transition impact since the organization that is to be achieved is calculated by each transition. Thus, the impact associated to each required change can be measured individually and more accurately than other approaches. The suitability of the adaptation must be considered taking into account not only the benefits obtained by adaptation but also the costs associated to this process. Another contribution of the MTDM is the possibility of including several transitions into the deliberation decision mechanism.

2.7 Additional Proposals for Designing Adaptive MAS

After presenting some proposals for developing Adaptive MAS from different perspectives, this section analyzes a set of additional Agent-Oriented Software Engineering (AOSE) development methodologies that allow designing OCMAS. These methodologies are GORMAS, O-MaSE/OMACS, MEnSA, PopOrg, LAO, Process Oriented Modeling Framework, MACODO, and the proposals by Grossi, Jonker, and Schatten.

2.7.1 GORMAS

GORMAS (Guidelines for ORganizational Multi-Agent Systems) (ABJ11), based on INGENIAS (PGSF05) and ANEMONA (BG08), defines a set of activities for the analysis and design of virtual organizations, including the design of their organizational structure and their dynamics. With this method, all the services offered and required by the virtual organization are clearly defined, as well as its internal structure and the norms that govern its behavior.

GORMAS is based on a specific method for designing human organizations, which consists on different phases of analysis and design. These phases have been adapted to the MAS field, in order to be able to catch all the design requirements of the organization from the agents' perspective.

GORMAS proposes to follow a basic sequence-guideline of organizational design, which allows to be integrated in a complete software development process, covering the phases of analysis, design, implementation, installation, and maintenance of the MAS. Taking into account the organizational perspective, the design phase is divided in two other phases: design of the organizational structure and design of the organizational dynamics. The system development process is iterative, since it is possible to return to a previous phase from any other.

To obtain the organizational model it is necessary to follow these phases:

- *Mission analysis*: it is carried out an analysis of the motivation for defining the organization, i.e. why this organization needs to be created; the results that are expected to be obtained; and the environment where it will be located, detailing products and/or services to offer, stakeholders, and their location.
- *Service analysis*: services to be offered in the system are analyzed in detail, including their requirements and associated processes. Tasks and objectives

associated to these services are also detailed.

- *Organizational design*:
  - The organizational dimensions (departmentalization, specialization, decision system, formalization, coordination) are analyzed, which impose certain requirements over the types of work, as well as over the diversity and interdependence of the tasks to be carried out.
  - The most appropriate organizational structure is selected, based on the analyzed dimensions. Organizational models are employed to specify roles, interactions and norms that are related to the structure.
- *Organizational dynamics design*:
  - For each identified service, needed interactions are detailed in order to carry out the service. Additionally, quality of service contracts for each service are defined.
  - Functionality as open system is established, which includes services to be published and the role enactment policies. Additionally, agents of the system are defined.
  - Tasks and activities are quantified or evaluated and mechanisms to determine whether the system goals are achieved are established. Also, organizational norms are revised to specify their supervisors.
  - The reward system is determined, in order to reward or penalized the organizational members that fulfil or violate the norms, respectively.

In conclusion, GORMAS takes the organization as a basis, providing a guideline that indicates the steps to be followed in order to model the organization, from its motivation to its dynamics, making emphasis on services.

2.7.2 O-MaSE/OMACS

OMACS (Organization Model for Adaptive Computational Systems) (DeL09) is the key component of a framework that allows a complex, adaptive system to design its own organization at runtime. OMACS defines the knowledge needed about a system structure and capabilities to allow it to reorganize at runtime in the face of a changing environment (which plays a critical role in the specification of adaptive organizations) and its agents' capabilities. However, the OMACS model is only useful if it is supported by a set of methodologies, techniques, and architectures that allow it to be implemented effectively on a wide variety of systems. Therefore, OMACS is related to the Organization-based Multiagent Systems Engineering (O-MaSE) methodology (GODOV08). O-MaSE is a methodology for creating processes during the development of OMACS-based systems.

The goal of the O-MaSE methodology is to allow process engineers to construct custom agent-oriented processes using a set of method fragments, all of which are based on a common metamodel. To achieve this, O-MaSE is defined in terms of a metamodel, a set of method fragments, and a set of guidelines.

The metamodel defines the main concepts used in O-MaSE to design multi-agent systems. It encapsulates the rules (grammar) of the notation and depicts those graphically using object-oriented concepts such as classes and relationships. The O-MaSE metamodel is based on the OMACS metamodel (DeL09) (featuring elements like Organization, Agent, Role, Goal or Domain Model) but adding new elements like Protocols or Environmental objects and properties.

Regarding method fragments, O-MaSE defines three main activities: (i) requirements engineering, (ii) analysis, and (iii) design. The requirement engineering activity is aimed at translating system requirements into system level goals. The Analysis activity focuses on modeling the relationships between the organization and its environment. Finally, the design activity defines the entities that build the system, such as agents.

One of the main advantages of OCMAS is that it is able to cope with adaptive systems. An OMACS system is able to control its adaptiveness by means of policies (similar to institutional norms) that can be used to specify either what the system should do or what the system should not do. These policies are essential in designing adaptive systems that are both predictable and reliable.

Developers of OMACS-based system have implemented several reorganization algorithms, most of which are centralized. These algorithms range from sound and complete total reorganization algorithms (with a very slow computational performance) to greedy algorithms (with low computational overhead, produce that generally produce adequate organizations).

2.7.3 MEnSA

MEnSA (MCCM10) is not only an AOSE methodology, but a whole project that is aimed to fill the gap between methodologies and infrastructures related to the software agent world. MEnSA authors are focused on proposing a methodology that has strong connections with the infrastructures in order to make the task of the developers easier in passing from the design phase to the implementation phase. For achieving this objective, MEnSA does not propose a new methodology, but a combination of existing methodologies, reusing their fragments (pieces of the process) following the Situational Method Engineering (SME) paradigm (BKKW07).

This project is based on four well-known MAS methodologies: Gaia (ZJW03), PASSI (Cos05), SODA (Omi01), and Tropos (BGG$^+$04). Using these four methodologies and their associated metamodels, a new process has been engineered. MEnSA metamodel combines elements from these four metamodels, and is adopted as the central element used to select and assembly fragments, and an algorithm to establish the instantiation order of the metamodel elements. This algorithm makes this method repeatable, reusable and as free as possible from the designer skills.

Up to this moment, this methodology employs sixteen different tasks, each one corresponding to a different fragment, divided between analysis and design phases. In the Analysis phase, there are fragments like Goal Definition (taken from Tropos), Analysis Environment (from SODA) or Choose Organization (from Gaia), while the Design phase is composed of fragments such as Constraint behavior (from SODA), Interaction Design (from SODA), or Agent Structure Definition (from PASSI).

However, MEnSA methodology is not yet completed (PC09) because the metamodel should be extended, if necessary, and finally MEnSA methodology has to be tested by means of a case study. Additionally, MEnSA developers state that tools will be provided in the future to support the methodology.

2.7.4 PopOrg

In order to manage the structural dynamics of a MAS, da Rocha and Pereira presented **PopOrg** (dRCD08), a model that is supported in two basic concepts: the *population* of an organization and its *structure*. The population of a MAS is the set of agents that inhabit it, as well as the behaviors, actions and exchange processes that agents are able to carry out. Each exchange process is determined by a function that, for each time unit, specifies a couple of actions and determines a sequence of exchanges that two agents will execute for carrying out these actions, in a simultaneous or alternate way. Moreover, the structure of the organization is composed of roles, representing behavior sets; and links between roles, which will be limited according to the capability that two roles have for being related between them. In order to link population and structure, the system is provided with a third element named *implementation* that relates roles with agents and links (or services) with exchange processes. These relations are not always one-to-one, so an agent can play different roles in the organization, different agents can play the same role, different links can be supported by different exchange processes, and different exchange processes can support a particular link.

This system is able to cope with both static and dynamic components of the organization, storing the different states by which the system passes through its execution. Different types of changes are taken into account in the organization, defined according to the element of the organization (population, structure or organization) that they are affecting. Changes that are considered for the population are: (i) changes in the set of behaviors that an agent is able to carry out; (ii) changes in the set of exchange processes that two agents can execute; and (iii) changes in the population of agents. The types of considered structural changes are: (i) change in a role; (ii) change in a link; (iii) change in the set of roles; (iv) change in the set of links; and (v) change in the link capability of the pairs of roles. Finally, changes in the implementation can appear on: (i) the way that roles and agents are related; and (ii) the way that links and exchange processes are related. PopOrg has the concept of service (named exchange value), defined as an action or task that an agent executes, which influences in a positive or negative way other agents in their effort for achieving a goal.

2.7.5 LAO

Virginia and Frank Dignum presented the **logic for agent organizations** (LAO) (DD07), which is an extension of the CTL logic (Eme91). This logic is capable of expressing the structure and the strategies of an organization. In this work, the concept of role is expressed through capabilities, abilities, activities and attempts of the agents and groups that build the organization. The *capability* of an agent is defined as a set of atomic propositions that the agent is able to fulfill. *Abilities* of an agent express the potential that an agent has to act over the world. *Activities* are represented as the action that an agent executes. Finally, the *attempts* represent the uncertainty of the system, since an agent is able to have the capability of executing an action, but this action can not be always successful. This formalization also defines different states of the world where the system is located and their transitions between states, and also establishes the topology of the MAS, such as network or hierarchy.

2.7.6 Process Oriented Modeling Framework

Popova and Sharpanskykh presented in (PS06) a process-oriented formal framework, named **Process Oriented Modeling Framework** (POMF). This work is expressed in the formal predicate language L_{PR} (Man96) that includes techniques for the dedicated analysis to check the correctness of the defined models. This work is structured in four views: (i) process-oriented view, (ii) organization-oriented view, (iii) agent view, and (iv) performance oriented view. The main view is the *process-oriented view*, which defines tasks (including their name, length and description), processes (defined using a task as a pattern), workflows (defined as a set of time-organized services) and resources (whose definition includes its name, category, measurement unit, and life-time). It is related to the *organization-oriented view* by means of the role entity, which describes the set of functionalities of the organizational process in a particular workflow. These functionalities are then assigned to individuals (agents) that will carry them out. Moreover, the *performance-oriented* view describes the organizational goals and the indicators that help checking whether they are being achieved. POMF also defines a set of restrictions: *structural*, which will act over the process and the task and resource hierarchies; *generical restrictions of the physical world*, which are independent from the application domain; and *domain specific restrictions*, which can be imposed by the organization, or coming from outside or from the physical world. These restrictions are verified to check their correctness only when changes are produced (or just the most relevant ones in the case of generical restrictions), which makes the checking process computationally more efficient than a *model checking* algorithm (Cla99).

2.7.7 MACODO

The **MACODO** framework (HWHJ09) focuses on the *organizational dynamics*, using Z as formal specification language (ASM80). In order to keep control of the organization in a system with a high rate of change it is necessary to use a middleware that must be able to monitor and model the organization. In order to carry out this modeling, MACODO is provided with a model that defines concepts like *roles*, *position* (similar to a job offer of a company) and *role contract* (an agreement between an agent and an organization for a particular position), to control the access to an available role; agents (including their context, consisting of the current state of the local environment of the agent and his position) and organizations. MACODO is also able to cope with organizational dynamics, including changes in the context of the system or in the set of agents of the system. Moreover, it also has rules to maintain the correct performance of the system.

2.7.8 Grossi proposal

The formalization of organizations proposed by Grossi *et al.* (GDD$^+$07) represents the organizational structure by analyzing three dimensions in an implicit way: (i) *power*, with the meaning of activities delegation; (ii) *coordination*, related to aspects of knowledge and information; and (iii) *control*, related to monitoring and recovering issues. Organizations are represented by the restrictions imposed by these three aspects. The structure relates the roles of the organization with these three dimensions. Roles of the organization are conceived around three basic notions: objectives, norms and information. Objectives and norms express the commitments and restrictions of the agents that play a determined role, being also a tool for the control of agents. Regarding information, agents must have knowledge about the current state of the organization to know whether the roles are played in a correct way. In order to define the organization, authors use their own language *Org* (GDD+07), which is a multi-modal propositional logic language whose semantics is based on Kripke models.

2.7.9 Jonker proposal

Jonker *et al.* (JSTY07) define a framework for the modeling and formal analysis of an organization, based on a representation of organizations by means of a set of roles related between them. This framework allows defining both the *structure* of the organization and its *dynamics*.

This proposal specifies organizations as composite roles, which will be refined on each aggregation level in other simple or composed roles, which interact between them. The behavior of each of these roles is defined by means of a set of dynamical properties, which are expressed using a temporal logic language, TTL (JT03). In order to model organizational norms, these are defined as static and dynamic properties of the role in the upmost level of aggregation. Finally, the environment of the organization is populated by agents that, under certain circumstances, are able to take a role inside the organization.

2.7.10 Schatten proposal

The proposal by Schatten (Sch13) studies an organization from five perspectives: structure, culture, processes, strategy, and individuals. In this proposal, an organization is defined following the fractal organization principle, which allows to recursively define it. Therefore, Schatten states that any agent is an organizational unit. Additionally, a set of organizational units that collaborate with a common objective is an organizational unit. The same concept of the fractal organization principle stands for organizational processes, which are activities performed by some individual agent; for organizational strategy (any measurable objective that can be achieved by an atomic activity); and for organizational culture (any cultural artifact that may be any piece of organizational knowledge, such as norms).

An organization in this proposal is defined by an active graph grammar (AGG), which is capable of modifying the relations between its nodes by means of active graph rewriting rules (AGRR) over labeled graphs. Following these rules, a labeled graph that represents an organization is able to modify its structure. An organization is also able to be represented by means of an organizational unit hypergraph to check the relationships between organizational units from a higher level. As an example of reorganization, the author presents an amoeba organization, where the organizational structure is represented by organizational units that are amoebas that can split and merge if the number of employees populating them is greater or smaller than given thresholds, respectively.

Authors of this proposal are working on the development of a formal meta-model (SGKK14) of organizational design concepts that can be used for developing model-driven software development tools that take advantage of these concepts. They take a three-step approach: (i) development of a semantic wiki on organizational design techniques that allows to edit a large corpus of knowledge to get an initial formalization; (ii) development of a formal semantic web ontology based on the insights gathered in the previous step; and (iii) development of a formal metamodel as a knowledge base of best practices for Large Scale MAS (LSMAS) using the ontology.

2.7.11 Discussion

These last sections have presented some relevant works on adaptive MAS and reorganization. Three of the analyzed works (the ones from ALIVE, Dignum and Dignum, and Hoogendoorn) state that they base their knowledge about organizational change in the human Organization Theory. As we previously expressed, this theory is a very important inspiration for us. The GORMAS methodology, whose organizational representation is considered in the work proposed in this thesis, also bases its organizational design process in a method to define human organizations. Both human and agent organizations have many elements in common. Therefore, it is interesting for us to take attention to related works that also study Organization Theory.

These three proposals also conceive organizational change as an endogenous process, where agents populating the organization will be responsible for organizational adaptation. These agents can be all the agents populating the organization, or only a set of them (typically playing a management role), that are organization aware, and are provided with all the knowledge they need to understand modifications and to perform changes inside the organization.

Nevertheless, the approach followed by MOISE is different. In this case, MOISE was not initially conceived to give support to adaptation, but it was later adapted to provide support to reorganization. As explained before, roles inside MOISE are distributed in different groups, and a new group, external to the organization, was added, so as to give support to adaptation. This makes the process of change to be exogenous, making a difference with respect to the rest of proposals. However, this process still preserves the common steps for reorganization, including monitoring, design and implementation of change. Our personal view is closer to the proposals that follow an endogenous process since we consider that adaptation must be seen as an internal process inside the organization.

It must be noticed that all these proposals follow a formal approach to define change. Dignum and Dignum have an interesting background in formal and logic languages, with proposals like OperA (Dig03) or LAO (DD07). Hoogendoorn also works with a formal logic language, TTL, that makes easier to check the correctness of the definition of a system and its adaptation process.

Regarding the OMACS model, it also gives support to reorganization algorithms, which are focused on calculating the organizational structure that maximizes the organizational structure.

MTDM analyzes change from the point of view of cost. Then, using the calculated estimations for the cost of a transition, and calculating the expected utility of the reorganized and current systems, it provides a mechanism for the decision about a multi-dimensional change in an organization.

We can state that all these proposals are very interesting approaches for reorganization, all presenting top-down designs, which our approach will also follow. We are interested in proposals where change is endogenous, based on Organizational Theory.

Next, we will compare the proposals presented in section 2.7, using the Organizational Dimensions in order to check whether they are taking into account the entities and concepts from each dimension. When analyzing an OCMAS methodology it is important to take into consideration the different organizational elements that they model. Table 2.1 depicts which of these elements are taken into account, grouped by the organizational dimensions (CAJB09) that describe a Virtual Organization: structural, functional, dynamical, environment and normative.

- **Structural Dimension**. Details the roles and social relationships, i.e. the elements that are independent from the final entities that execute them.
- **Functional Dimension**. Details the functionality of the system based on services, tasks and objectives.
- **Dynamical Dimension**. Defines interactions between agents and the role enactment process, defining the roles that organizational units or agents are able to play.
- **Environment Dimension**. Describes the entities that populate the environment of a MAS and its topology.
- **Normative Dimension**. Describes normative restrictions on the behavior of the entities of the system, including sanctions and rewards.

The proposals have been separated in two different groups (as it can be seen in Table 2.1 and Table 2.2, respectively), the first one being the proposals that are methodologies for the development of OCMAS, and the second group covers the proposals defined as formal approaches for the definition of an OCMAS.

Regarding the analyzed methodological proposals, two of them present features to develop adaptive MAS from top-down, in the case of ALIVE, and bottom-up, in the case of ADELFE. Our proposal will be closer to ALIVE than to ADELFE, but this second methodology will be also taken into account since bottom-up changes are also interesting when considering organizational adaptation.

The rest of the analyzed proposals are also able to give support to reorganization. MOISE introduced an add-on that supported reorganization, using an external group of roles.

Regarding the implementation phase, GORMAS itself does not give support to the implementation phase, but it is provided with a tool named EMFGormas (GAG10), which generates code for the THOMAS framework (ABC^{+}11). THOMAS facilitates the deployment of open, organization-based multi-agent systems. MEnSA is provided with an implementation step extracted from PASSI. O-MaSE methodology does not have associated tools to carry out with its implementation step, but it provides an agent architecture, named Organization-Based Agent (OBA), which is the architecture to be used in an OMACS-based system. ADELFE, ALIVE and MOISE have specific tools that carry out the implementation of these systems.

Organizational concepts					
	ALIVE	MOISE	O-MaSE	MEnSA	GORMAS
Structural dimension					
Roles	✓	✓	✓	✓	✓
Groups		✓		✓	✓
Agents	✓	✓	✓	✓	✓
Relations	✓	✓		✓	✓
Topology	✓	✓		✓	✓
Functional dimension					
Capabilities	✓		✓		
Services	✓		✓	✓	✓
Objectives	✓	✓	✓	✓	✓
Dynamical dimension					
Interactions	✓	✓	✓	✓	✓
Transitions	✓	✓			
Environment dimension					
Environment	✓	✓	✓	✓	✓
Resource			✓	✓	✓
Normative dimension					
Norms	✓	✓	✓	✓	✓

Table 2.1: Comparison between different OCMAS methodologies

Taking a look at Table 2.1, at first sight we can check that the major part of the elements we need to consider when developing an organization is modeled by these methodologies. There are some key entities that are presented in all the proposals: roles, agents, objectives, interactions and environment, which are the main elements that define an agent organization. Norms are an important element to manage open, organization-oriented MAS, and are also taken into account in all the proposals. Regarding adaptation aspects, two elements have crucial importance: the environment, which has to be under control in order to develop adaptive actions; and the transitions, which will describe how an organization changes through time. As it can be checked, transitions are only represented in ALIVE and MOISE, so the systems represented using these methodologies are able to store not only their present state, but also their past state, and even their future possible states in the case of ALIVE.

Table 2.2 makes a similar comparison between the formal approaches that we have studied. Therefore, the Organizational Dimensions are used as a reference so as to check the different organizational elements that the formalizations for defining an OCMAS take into account.

MOISE^{Inst} is composed of four specifications, distributed in a similar way than the Organizational Dimensions are. The *Structural Specification* (SS) contains elements from the Structural Dimension, such as roles, role relationships and groups, but it does not model the topology of the system. The *Functional Specification* (FS), related to the Functional Dimension, only defines the goals that the system must achieve. The *Contextual Specification* (CS) defines the environment by means of contexts of the Environment Dimension, and transitions between contexts that represent the dynamics of the system, included in the Dynamical Dimension. However, the CS does not model the resources populating the environment or the interactions between agents. The *Normative Specification* (NS) defines the rights and duties of roles and groups inside the organization, which are known as norms

Organizational concepts								
	MOISE	**PopOrg**	**LAO**	**POMF**	**MACODO**	**Grossi**	**Jonker**	**Schatten**
Structural Dimension								
Roles	✓	✓	✓	✓	✓	✓	✓	
Groups	✓		✓		✓			✓
Agents	✓	✓	✓	✓	✓	✓	✓	✓
Relations	✓	✓	✓			✓	✓	✓
Topology			✓					
Functional Dimension								
Capabilities		✓	✓	✓	✓			
Abilities		✓	✓					
Services		✓		✓				✓
Objectives	✓		✓	✓		✓		✓
Dynamical Dimension								
Interactions	✓	✓				✓	✓	✓
Dynamics	✓	✓	✓		✓		✓	✓
Environment Dimension								
Environment	✓		✓		✓		✓	✓
Resources				✓				
Normative Dimension								
Norms	✓				✓	✓		✓
Syntax			CTL	L_{PR}	Z	Org	TTL	AGG
Semantics			CTL*	T_{PR}		Org		

Table 2.2: Comparison between different formal representations

in the Normative Dimension. Finally, the *Organizational Entity* (OE) controls the dynamic elements of the organization, including agents and all events that they generate, such as their interactions.

In PopOrg, the population of a MAS is its set of agents, as well as the behaviors and actions, which represent the capabilities and abilities from the Functional Dimension; and the exchange processes (services from the Functional Dimension) that agents are able to carry out. Therefore, the population of a PopOrg organization mainly takes concepts from the Functional Dimension plus agents from the Structural Dimension. Moreover, the structure of the organization is composed of roles and the links between them, which are elements that belong to the Structural Dimension. To relate the population and the structure, PopOrg has a third element called *implementation* that relates roles with agents, and links with exchange processes. Also, PopOrg stores the different states that the system goes through during its execution. Unfortunately, PopOrg does not model any of the entities from the Environment or Normative Dimensions.

In LAO, the Functional Dimension is completely represented, including agents, objectives, groups, topology and roles, which are represented by means of capabilities and abilities. LAO additionally defines different states of the world where the system is located (related to the Environment Dimension) and its transitions (related to the Dynamical Dimension). LAO is a very complete proposal, since it takes into account a large subset of the elements of the Organizational Dimensions, but it does not formalize the Normative Dimension.

In POMF, the *process oriented view* includes the concept of service from the Functional Dimension, being a workflow divided into processes that are split into tasks. It also includes the resources of the Environment Dimension. The *organization oriented view* is related with both Structural and Functional Dimensions. On the one hand, it includes the role entity, which describes the set of capabilities of the organizational processes in a specific workflow that are then assigned to agent entities. On the other hand, agent entities are defined in the *agent view*, which is related to the Structural Dimension. Finally, the *performance oriented view* describes the organizational goals, such as the Functional Dimension does. However, POMF does not provide a formalization for neither Dynamical nor Normative Dimensions.

MACODO offers a model with concepts related to the Functional Dimension, such as roles, contracts of roles, agents, and organizations. Relations of hierarchy and communication between roles are not considered. A role is described as a set of capabilities, which is the only entity from the Functional Dimension that MACODO takes into account. Since MACODO is focused on self-organization, dynamics of the system from the Dynamical Dimension, including changes in its context or in its set of agents, are formalized. To control the activities that the organization carries out, MACODO is enhanced with a set of laws, similar to norms from the Normative Dimension. Although MACODO does not model other relevant organizational concepts such as objectives, it deals with elements from all the Organizational Dimensions.

The organizational formalization proposed by Grossi *et al.* considers the concepts of *role*, establishing relations between them; and the *agent* concept from the Structural Dimension. The roles of the organization are conceived around three basic notions: objectives, norms and information. Objectives are the only elements related to the Functional Dimension that are presented in this proposal; and the Normative Dimension is taken into account using norms. Regarding information, knowledge about the current state of the organization can be given to agents by other agents, so this is a type of interaction from the Dynamical Dimension. Since this proposal is focused on modeling the organization structure, it does not take into account the Environment Dimension.

The proposal by Jonker *et al.* formalizes three concepts from the Structural Dimension: roles, agents and relations between roles. These relations enable the interactions from the Dynamical Dimension, which is completed by taking into account the dynamics of the organization. One of the main advantages of this work is that it is able to explicitly model the environment of the Environment Dimension, although it does not model environmental resources. The main lack of this formalization is that it does not formalize any concept from the Functional and Normative Dimensions, so designers are not able to model concepts such as objectives and norms.

Finally, the proposal by Schatten defines agents and relations between them, but does not define the concept of role. From a functional perspective, it defines the services as processes (activities and procedures of an organization). Regarding the Dynamical Dimension, this proposal presents a set of active graph rewriting rules (AGRR) that express the changes between different states of the graph (each state representing the interaction between nodes of the graph). Thus, the graph describes the environment where the organization is located. This proposal also presents the concept of norms, as part of the organizational culture, where other concepts such as reward systems and knowledge are also defined.

2.8 Conclusion

This chapter has presented a state of the art on the adaptation concepts that will be considered during this thesis. First, the models that have been taken into account from decades ago to develop organizational change, and the forces that drive this change have been presented. Then, the different types of changes, depending on how this change is carried out, from a temporal or an structural point of view have been depicted. Finally, there have been described different approaches to change an organization from a top-down (reorganization) or a bottom-up (self-organization) approach.

Additionally, this chapter has presented different methodologies and processes that define OCMAS. Elements that they define have been studied, in order to establish a comparison between them.

Finally, the most relevant formal approaches to define OCMAS have been briefly described in this chapter. Generally, all the analyzed formalizations present a good approach to define an organization in a formal way. Nevertheless, none of the proposals take into account all entities taken from Organizational Dimensions, but these formalizations propose different ways to structure an OCMAS. Thus, it seems interesting to be provided with an explicit description of the Organizational Dimensions, which are useful for representing organizational elements. Therefore, in the next chapter, our proposal to model organizations will be presented, which models an organization clearly defining its dimensions. This proposal also integrates some features taken from some works analyzed in this state of the art.

3

Virtual Organization Formalization

3.1 Introduction

During the last years, different trends to develop Multi-Agent Systems (MAS) have appeared. One of the most important approaches is the organizational approach. Organizations describe system functionality, structure, environment and dynamics. In Organization-Centered MAS (OCMAS), the organization exists as an explicit entity of the system (LE98), defined by its designers following a top-down approach. In OCMAS, agents are aware of the organization in which they are participating and they are provided with a representation of it. Agents can use this knowledge to reason about it and to establish relationships and interactions to reach their objectives.

OCMAS can be structured by splitting them in different dimensions (HSB02a). Specifically, OCMAS (which are also known as VOs, as stated in Chapter 2) can be structured by means of the Organizational Dimensions (CAJB09), which should be considered when modeling an organization. These dimensions are: *structural*, *functional*, *dynamical*, *environment* and *normative.* As explained in Chapter 2, current formal proposals only define a subset of the dimensions and concepts presented in the Organizational Dimensions. Thus, it seems necessary to be provided with a formalization that clearly models the Organizational Dimensions, making a clear difference between them.

The objective of this work is to present a formal framework to define a VO, taking the Organizational Dimensions as a basis. This formalization will be useful when dealing with self-adaptive and self-organization concepts, since it will be established how the system changes through time. The rest of this work is structured as follows: Section 3.2 describes the Organizational Dimensions. Section 3.3 presents the *Virtual Organization Formalization* (VOF), our formal framework to define VOs. Section 3.4 presents an example that uses VOF. Section 3.5 presents a discussion between our proposal and the analyzed frameworks. Finally, Section 3.6 gives our conclusions on this issue.

3.2 Organizational Dimensions

When modeling an organization, the following dimensions should be taken into account (CAJB09): (i) *structural*, describing the entities that organize the VO; (ii) *functional*, which details the functions, goals and services of the organization; (iii) *dynamical*, which considers the interactions between elements, and their effects; (iv) *environment*, describing the elements that surround the system; and (v) *normative*, which defines the mechanisms used by the society to influence the behavior of its members.

The **Structural Dimension** comprises all the elements of the organization that are independent from the agents that are part of it. Thus, it is based on roles, groups and their patterns of interrelationship (inheritance, compatibility, communication, and so on). Additionally, the topology of the system is established.

The **Functional Dimension** specifies the global goals of the organization, its offered functions and services, the goals followed by the different components of the organization and the tasks and plans that must be executed to reach these goals.

The **Dynamical Dimension** specifies how the organization evolves through time, detailing the way in which agents enter and leave it, how they adopt certain roles according to their capabilities and abilities, and how they can participate in the units or groups of the organization where they are admitted. This dimension also details the interactions that take place among internal and external entities.

The **Environment Dimension** describes how the organization and its agents are connected with external agents and other types of entities such as artifacts, applications or resources; and how agents can perceive and act on the environment.

Finally, the **Normative Dimension** determines the set of defined actions and rules to manage the behavior of the members of the organization. Norms are widely used to limit the autonomy inside societies and to solve coordination problems, especially when it is not possible to exercise a total social control.

3.3 Formal description of a Virtual Organization

In this section the concept of Virtual Organization (VO) will be defined in a formal way, taking into account its organizational dimensions. Initially, we need to define which are the components of an Organization-Centered Multi-Agent System.

Definition 3. *An Organization-Centered Multi-Agent System (OCMAS), at a given time $t \in T$, is defined as a tuple $OCMAS(t) = \langle \mathcal{VO}, \mathcal{R}, \mathcal{TA}, \mathcal{S}, \mathcal{G}, \mathcal{WS}, \mathcal{AR}, \mathcal{A} \rangle$, where:*

- $\mathcal{VO}$ *is the set of virtual organizations that populate the environment of the domain we are working on.*
- $\mathcal{R}$ *represents the set of roles available at the system.*
- $\mathcal{TA}$ *is the set of tasks available at the system. For a description of a task, see Definition 6, which will be lately defined.*
- $\mathcal{S}$ *is the set of services available at the system, where a service will be lately described in Definition 7.*
- $\mathcal{G}$ *is the set of goals available at the system.*
- $\mathcal{WS}$ *is the set of workspaces that build the environment of the domain we are working on. See Definition 10 for a description of a workspace.*
- $\mathcal{AR}$ *is the set of artifacts that populate the environment. A definition for an artifact is also given in Definition 10.*
- $\mathcal{A}$ *is the set of agents of the system. An agent is described in Definition 13.*
- T *is the set of all possible moments in time. Time is discretized in this definition.*

The *Virtual Organization Formalization* (VOF) is focused on three elements: (i) the Organizational Specification (OS), which details the set of 'static' elements of the organization; (ii) the *Organizational Entity* (OE), which represents the instantiation of the elements in OS; and (iii) the *Organizational Instantiation* (ϕ), which relates elements from OS with elements from OE.

Definition 4. [Virtual Organization] *A Virtual Organization $vo \in \mathcal{VO}$ is defined, at a given time t, as a tuple $vo(t) = \langle OS(vo,t), OE(vo,t), \phi(vo,t)\rangle$ where:*

- $OS(vo,t)$ *refers to the* ***Organizational Specification*** *of vo, which describes the structural definition of the organization, at a given time t. It is defined as $OS(vo,t) = \langle SD(vo,t), FD(vo,t),\ ED(vo,t), ND(vo,t)\rangle$ where:*

  - *$SD(vo,t)$ is the Structural Dimension of vo at a given time t. It defines roles and relations between them. A detailed description of SD is given in Definition 5.*
  - *$FD(vo,t)$ is the Functional Dimension of vo at a given time t. It describes the functionalities of the system, including goals, services and tasks. FD is defined in Definition 8.*
  - *$ED(vo,t)$ is the Environment Dimension of vo at a given time t, which describes the environment of a Virtual Organization. It is detailed in Definition 10.*
  - *$ND(vo,t)$ is the Normative Dimension of vo at a given time t. It defines the norms that rule a Virtual Organization. A definition for ND is given in Definition 12.*

- *$OE(vo,t)$ refers to the **Organizational Entity** of vo at a given time t, which represents the entities populating the system. It is detailed in Definition 14.*
- *$\phi(vo,t)$ refers to the **Organizational Instantiation** of vo at a given time t, allowing to relate $OS(vo,t)$ with $OE(vo,t)$. It has information about role allocation and active norms and services (see Definition 15).*

The following subsections will describe in detail these three elements: organizational specification, organizational entity, and organizational instantiation.

3.3.1 Organizational Specification

The Organizational Specification is composed of: (i) the *Structural Dimension*, which contains roles, organizational units and their relationships; (ii) the *Functional Dimension*, describing objectives, functionalities and services of an organization; (iii) the *Environment Dimension*, which describes the artifacts and workspaces from the environment of the organization; and (iv) the *Normative Dimension*, which defines the norms that rule the system.

3.3.1.1 Structural Dimension

The *Structural Dimension* describes the roles of the organization and their relations. It allows defining the structural components of an organization, i.e., all the elements that are independent from the entities that are finally executed. In a more specific way, it defines a pattern for an organization at a given time. Then, entities populating the organization, described in the OE (see Definition 14) will have to follow this pattern when building a virtual organization.

Definition 5. [Structural Dimension] *The Structural Dimension (SD) of a Virtual Organization* $vo \in \mathcal{VO}$ *is defined at a given time* t *as* $SD(vo, t) = \langle R, Relations \rangle$ *where:*

- $R \subseteq \mathcal{R}$ *refers to the roles that can be played by the organization, or by the entities populating it.*
- *Relations is a set of predicates that define relationships between roles in the organization. They allow defining elements of the organization such as role hierarchy, organizational topology or communication links between different roles, where:*

$$Relations = \begin{cases} inf(x,y) \\ mon(x,y) \\ sup(x,y) \\ comp(x,y) \\ roleInh(x,y) \end{cases}$$

where: ***inf(x,y)*** *(information) refers that there is an information relation between roles* x *and* y*, which allows communications between them;* ***mon(x,y)*** *(monitoring) allows a role* x *to monitor the activities of another role* y*;* ***sup(x,y)*** *(supervision) defines that an agent playing a specific role* x *can transfer or delegate one or some of his objectives to a subordinate role* y*;* ***comp(x,y)*** *(compatibility) depicts that an agent playing a role* x *can play another given role* y *in the organization at the same time; and* ***roleInh(x,y)*** *(role inheritance) represents the inheritance relations between roles, i.e., the role* x *that a given role* y *is directly inheriting from.*

Roles of a VO. The function $roles(vo, t)$ expresses the set of roles that belong to the Virtual Organization $vo \in \mathcal{VO}$ at a given time t. Formally,

$$roles : \mathcal{VO} \times T \to 2^{\mathcal{R}}$$

$$roles(vo, t) = \bigcup_{\forall r_i (r_i \in R \wedge R \in SD(vo,t))} \{r_i\} \tag{3.1}$$

Role inheritance. First of all, we define the function *inheritedRole*, that specifies all the roles that are inheriting from a given role $r \in R$ in a virtual organization vo at a time t. These inherited roles are more specific than r.

$$inheritedRole : R \to 2^{R}$$

$$inheritedRole(r) = \bigcup_{\forall r_i (r_i \in R \wedge roleInh(r_i,r) \in Relations)} \{r_i\} \tag{3.2}$$

Additionally, we define the function *genericRole* that relates a role with the set of roles that are in upper levels in the goal inheritance relations, i.e., roles that are more generic. Given a role $r \in R$, it returns a set (that could be empty) containing all roles from the upper levels of the role hierarchy. The firstly computed roles of this set are the most immediate roles in the hierarchy, i.e., the generic roles for a given role. Therefore, this function also allows traveling across the tree built by roles. The rest of the elements are recursively calculated. Therefore:

$$genericRole : R \to 2^{R}$$

$$genericRole(r) = \begin{cases} if\ r = \emptyset : \emptyset \\ else : \bigcup_{\forall r_i (r_i \in R \wedge roleInh(r,r_i) \in Relations)} \{r_i \cup genericRole(r_i)\} \end{cases} \tag{3.3}$$

Properties of the relations. The social relation inf is symmetrical (see Table 3.1, Equation 1), since a role can provide information to a second role, and viceversa; transitive (see Table 3.1, Equation 2), since agents can build an information chain, and reflexive (see Table 3.1, Equation 3) as an agent can send information to himself. The relations mon and sup are both asymmetrical (see Table 3.1, Equations 4 and 5), since an agent cannot monitor or supervise the agent which is monitoring or supervising him; reflexive (see Table 3.1, Equations 6 and 7), because an agent can collaborate or supervise himself; and transitive (see Table 3.1, Equations 8 and 9), allowing to create a command chain inside the organization.

The compatibility relation ($comp$) has reflexive (see Table 3.1, Equation 10) and transitive (Table 3.1, Equation 11) properties, because a role is compatible with itself and a role is compatible with the roles that have a compatibility relationship with its compatible roles.

It is interesting to notice that the $comp$ relation is not symmetrical (e.g., $comp(r_1, r_2)$ not always implies $comp(r_2, r_1)$). For example, the relation $comp(Professor, Teacher)$ is correct, because a professor can work as a teacher in every moment, but a teacher might not be capable of playing the role of professor. Finally, relations $roleInh$ and $comp$ are related, since an agent playing a specialized role is capable of playing its generalized role. Formally:

$$\forall r_1, r_2 \in R : roleInh(r_1, r_2) \rightarrow comp(r_2, r_1) \tag{3.4}$$

Let $r_1, r_2 \in R$ be two roles belonging to OS. The information, collaboration and supervision relations define the following relations in an implicit way:

$$sup(r_1, r_2) \rightarrow mon(r_1, r_2) \tag{3.5}$$

$$mon(r_1, r_2) \rightarrow inf(r_1, r_2) \tag{3.6}$$

This means that a supervision relationship between two roles implies that an agent playing a supervisor role will monitor the agent playing the supervised role. Also, a monitoring relationship between two roles implies that an information link between them must exist.

The *roleInh* relation is: (i) asymmetrical (Table 3.1, Equation 12), because a role cannot have an inheritance relation with itself; (ii) transitive (Table 3.1, Equation 13), because the relations between roles are transitive to allow defining a complete role hierarchy; and (iii) irreflexive (Table 3.1, Equation 14) because a subordinated role cannot supervise its supervisor.

Properties	
1) $\forall r_1, r_2 \in R : inf(r_1, r_2) \rightarrow inf(r_2, inf_1)$	2) $\forall r_1, r_2, r_3 \in R : inf(r_1, r_2) \wedge inf(r_2, r_3) \rightarrow inf(r_1, r_3)$
3) $\forall r_1 \in R : inf(r_1, r_1)$	4) $\forall r_1, r_2 \in R : mon(r_1, r_2) \rightarrow \neg mon(r_2, r_1)$
5) $\forall r_1, r_2 \in R : sup(r_1, r_2) \rightarrow \neg sup(r_2, r_1)$	6) $\forall r_1 \in R : mon(r_1, r_1)$
7) $\forall r_1 \in R : sup(r_1, r_1)$	8) $\forall r_1, r_2, r_3 \in R : mon(r_1, r_2) \wedge mon(r_2, r_3) \rightarrow mon(r_1, r_3)$
9) $\forall r_1, r_2, r_3 \in R : sup(r_1, r_2) \wedge sup(r_2, r_3) \rightarrow sup(r_1, r_3)$	10) $\forall r_1 \in R : comp(r_1, r_1)$
11) $\forall r_1, r_2, r_3 \in R : comp(r_1, r_2) \wedge comp(r_2, r_3) \rightarrow comp(r_1, r_3)$	12) $\forall r_1 \in R : \neg roleInh(r_1, r_1)$
13) $\forall r_1, r_2, r_3 \in R : roleInh(r_1, r_2) \wedge roleInh(r_2, r_3) \rightarrow roleInh(r_1, r_3)$	14) $\forall r_1, r_2 \in R : roleInh(r_1, r_2) \rightarrow \neg roleInh(r_2, r_1)$

Table 3.1: Properties of the relations

Summarizing, the Structural Dimension describes the roles of the organization, and the relationships between them, including role inheritance.

3.3.1.2 Functional Dimension

The *Functional Dimension* details the specific functionality of the system, based on services, tasks and goals, as well as the interactions of the system, activated by means of goals or service usage. It allows defining the functionality of the MAS, including services that the organization offers, and goals that entities pursue.

Definition 6. [Task] *A task $ta \in \mathcal{TA}$ is an action that can be carried out by an entity. A task ta is characterized as $ta = \langle Preconditions, Action, Goals, Duration \rangle$ where:*

- *$Preconditions$ is the set of conditions that the system must fulfill prior to the execution of this task. Therefore, a task can be executed iff $Preconditions = true$.*
- *$Action$ refers to the specific action that this task carries out.*
- *$Goals \subseteq \mathcal{G}$ is the set of goals that are achieved with the execution of this task. They can also be seen as the postconditions of the task.*
- *$Duration \in T \times T$ is the time that an action takes to be carried out, represented by two moments in time i.e., start and end of the task.*

Goals achieved by a task. To describe the goals achieved by a task $ta \in \mathcal{TA}$, the function *achieves* is used:

$$achieves : \mathcal{TA} \to 2^{\mathcal{G}}$$

$$achieves(ta) = \bigcup_{\forall g_i (g_i \in Goals \wedge Goals \in ta)} \{g_i\} \tag{3.7}$$

Definition 7. [Service] *A service $s \in \mathcal{S}$ describes a specific set of functionalities being offered by entities playing a specific role to other entities. A service s is characterized as $s = \langle Profile, Process, Performance \rangle$, where:*

- *$Profile = \langle Client, Provider, Input, Output, Preconditions, Postconditions, Obtains, ont \rangle$, where:*
  - *$Client \subseteq \mathcal{R}$ relates a service with the set of roles that use it.*
  - *$Provider \subseteq \mathcal{R}$ relates a service with the set of roles that offer it.*
  - *$Input$ represents the set of inputs (i.e., parameters) requested by the service.*

- *Output represents the set of outputs (e.g., generated data) returned by the service.*
- *Preconditions* $\subseteq \mathcal{AR}$ *represents the set of resources that are required before executing the service.*
- *Postconditions* $\subseteq \mathcal{AR}$ *represents the set of products generated after the execution of the service.*
- *Obtains* $\subseteq \mathcal{G}$ *describes the set of goals that can be achieved by a service (i.e., the effects produced by the service), thus defining the functionality of the system.*
- *ont is the ontology that gives meaning to the different elements of the service.*

• $Process = \langle Steps, Comm \rangle$, *where:*

- $Steps = \{x_i \in \{\mathcal{TA}, \mathcal{S}\} | x_i \preceq x_{i+1}\}_{0<i<N-1}$, *is an ordered list depicting the step-by-step process (each step could be either an invoked service or a task) followed by the service in order to achieve its goal, where* $\preceq$ *is a precedence function between services and tasks that compose the steps followed by the service, and* N *is the length of the list.*
- *Comm defines the communication protocol e.g., FIPA ACL or KQML.*

• $Performance = \langle QoS, CostRes, BenPro \rangle$, *where:*

- $QoS \in [0, 1]$ *refers to the expected quality of this service. It is a rate from 0 to 1, meaning the minimum percentage that the service is assured to be properly provided.*
- $CostRes : \mathcal{AR} \to \mathbb{R}$, *returns the cost of consuming a resource as a precondition of this service. A tuple* $\langle res, cost \rangle$ *means that a resource 'res' has a cost of 'cost' units.*
- $BenPro : \mathcal{AR} \to \mathbb{R}$, *returns the benefit returned by the product which is a postcondition of the service. A tuple* $\langle pro, ben \rangle$ *means that a product 'pro' generates a benefit of 'ben' units.*

In order to facilitate the definition of future aspects of this formalization, the following functions are defined:

Client roles. $client(s)$ is a function that returns the set of roles that are able to use a service $s \in \mathcal{S}$.

$$client : \mathcal{S} \rightarrow 2^{\mathcal{R}}$$

$$client(s) = \bigcup_{\forall r_i (r_i \in Client \wedge Client \in Profile \wedge Profile \in s)} \{r_i\} \tag{3.8}$$

Provider roles. Similarly, $provider(s)$ returns the roles that provide a service $s \in \mathcal{S}$.

$$provider : \mathcal{S} \rightarrow 2^{\mathcal{R}}$$

$$provider(s) = \bigcup_{\forall r_i (r_i \in Provider \wedge Provider \in Profile \wedge Profile \in s)} \{r_i\} \tag{3.9}$$

Goals achieved by a service. $obtains(s)$ returns the set of goals achieved if the service $s \in \mathcal{S}$ is executed.

$$obtains : \mathcal{S} \rightarrow 2^{\mathcal{G}}$$

$$obtains(s) = \bigcup_{\forall g_i (g_i \in Obtains \wedge Obtains \in Profile \wedge Profile \in s)} \{g_i\} \tag{3.10}$$

Preconditions of a service. $preconditions(s)$ returns the set of resources that are preconditions required for the execution of a service $s \in \mathcal{S}$.

$$preconditions : \mathcal{S} \rightarrow 2^{\mathcal{AR}}$$

$$preconditions(s) = \bigcup_{\forall ar_i (ar_i \in Preconditions \wedge Preconditions \in Profile \wedge Profile \in s)} \{ar_i\} \tag{3.11}$$

Postconditions of a service. *postconditions*(s) returns the set of products generated as the postconditions of a service $s \in \mathcal{S}$.

$$postconditions : \mathcal{S} \rightarrow 2^{\mathcal{AR}}$$

$$postconditions(s) = \bigcup_{\forall ar_i (ar_i \in Postconditions \wedge Postconditions \in Profile \wedge Profile \in s)} \{ar_i\} \tag{3.12}$$

Steps of a service. *steps*(s) returns the list of tasks and services that compose the service $s \in \mathcal{S}$.

$$steps : \mathcal{S} \rightarrow 2^{\{\mathcal{TA} \cup \mathcal{S}\}}$$

$$steps(s) = \bigcup_{\forall ts_i (ts_i \in Steps \wedge Steps \in Process \wedge Process \in s)} \{ts_i\} \tag{3.13}$$

Quality of a service. *qos*(s) returns the quality of the service s.

$$qos : \mathcal{S} \rightarrow \mathbb{R}$$

$$qos(s) = qos : qos \in QoS \wedge QoS \in Performance \wedge Performance \in s \tag{3.14}$$

Cost of consuming a resource. The cost of consuming a resource (represented by an artifact ar) by a service s is expressed by means of this function:

$$costResource : \mathcal{AR} \times \mathcal{S} \to \mathbb{R}$$

$$costResource(ar, s) = c | CostRes(ar) = c \wedge CostRes(ar) \in Performance \wedge \\ Performance \in s \quad (3.15)$$

which means that consuming the resource $ar \in \mathit{Preconditions}$ to execute the service s has a cost of c units (money, computational unit, etc.).

Benefit of a product. The benefit of a product (represented by an artifact pr) generated by a service is also expressed by means of the following function:

$$benefit : \mathcal{AR} \times \mathcal{S} \to \mathbb{R}$$

$$benefit(pr, s) = c | BenPro(ar) = c \wedge BenPro(ar) \in Performance \wedge \\ Performance \in s \quad (3.16)$$

which means that the product $pr \in \mathit{Postconditions}$ generated by the service s produces a benefit of c units (that can be money or any other kind of benefit).

Cost of a service. The cost of consuming a service s is expressed by means of the function *costServ*, which sums the costs generated by all the resources consumed by the service s, making use of the function *costResource* (defined in Equation 3.15):

$$costServ : \mathcal{S} \to \mathbb{R}$$

$$costServ(s) = \sum_{\forall ar_i \in preconditions(s)} costResource(ar_i, s) \qquad (3.17)$$

Benefit of a service. The benefit of executing a service s is calculated as the sum of the individual benefits of the products (the benefit of each product is calculated using the function $benefit$ in Equation 3.16) generated by the service, and it is expressed by means of the function $benServ(s)$:

$$benServ : \mathcal{S} \rightarrow \mathbb{R}$$

$$benServ(s) = \sum_{\forall pr_i \in postconditions(s)} benefit(pr_i, s) \qquad (3.18)$$

Definition 8. [Functional Dimension] *The Functional Dimension (FD) of the Organizational Structure of a Virtual Organization $vo \in \mathcal{VO}$ at a given time t is defined as $FD(vo, t) = \langle G, S, GoalDependency \rangle$ where:*

- *$G \subseteq \mathcal{G}$ represents the goals inside the organization, where a goal $g \in G$ is defined as $\langle state(t), deadline \rangle$ where $state(t) \in \{active, achieved, failed\}$ is the current state of the goal (at time t), and $deadline \in T$ is the maximum time at which the goal is expected to be achieved. "active" means that the goal is being pursued by the system and $t \leq deadline$; "achieved" means that the goal has been successfully achieved and $t > deadline$; and "$failed$" means that the goal failed to be achieved and $t > deadline$.*

- *$S \subseteq \mathcal{S}$ is the set of services that the organization offers.*

- *$GoalDependency : G \rightarrow 2^{\mathcal{G}}$ defines dependencies between goals of the organization, i.e., given a tuple $\langle g_1, g_2 \rangle \in GoalDependency$ presents the relationship between goals g_1 and g_2, meaning that g_1 requires that g_2 is achieved before it can be achieved (GMP03).*

Properties of the relations. The *GoalDependency* relation is irreflexive (Table 3.2, Equation 1), asymmetrical (Table 3.2, Equation 2), and transitive (Table 3.2, Equation 3), since a goal cannot be related with itself, neither with its predecessor but it can be related with the successors of its successors.

Properties	
1) $\forall g \in \mathcal{G} : \neg GoalDependency(g, g)$	2) $\forall g_1, g_2 \in \mathcal{G} : GoalDependency(g_1, g_2) \rightarrow \neg GoalDependency(g_2, g_1)$
3) $\forall g_1, g_2, g_3 \in \mathcal{G} : GoalDependency(g_1, g_2) \wedge GoalDependency(g_2, g_3) \rightarrow GoalDependency(g_1, g_3)$	

Table 3.2: Properties of the relations

Organizational services. The function *services*(vo, t) returns the list of services that the Virtual Organization $vo \in \mathcal{VO}$ provides at a time t. Formally,

$$services : \mathcal{VO} \times T \rightarrow 2^{\mathcal{S}}$$

$$services(vo, t) = \bigcup_{\forall s_i (s_i \in S \wedge S \in FD(vo,t))} \{s_i\} \tag{3.19}$$

Organizational goals. The function *goals*(vo, t) returns the list of goals pursued by the Virtual Organization $vo \in \mathcal{VO}$ at a time t. Formally,

$$goals : \mathcal{VO} \times T \rightarrow 2^{\mathcal{G}}$$

$$goals(vo, t) = \bigcup_{\forall g_i (g_i \in G \wedge G \in FD(vo,t))} \{g_i\} \tag{3.20}$$

Expected goals. The function *expectedGoals*(vo, t) returns the set of goals that are expected to be achieved by the organization $vo \in \mathcal{VO}$ at a time t. Formally,

$$expectedGoals : \mathcal{VO} \times T \rightarrow 2^{\mathcal{G}}$$

$$expectedGoals(vo,t) = \bigcup_{\forall g_i (g_i \in goals(vo,t) \wedge g_i = \langle state(t), t_1 \rangle \wedge t_1 \leq t)} \{g_i\} \tag{3.21}$$

Achieved goals. The function $achievedGoals(vo,t)$ returns the set of goals that have been achieved by the organization $vo \in \mathcal{VO}$ at a time t. Formally,

$$achievedGoals : \mathcal{VO} \times T \rightarrow 2^{\mathcal{G}}$$

$$achievedGoals(vo,t) = \bigcup_{\forall g_i (g_i = \langle \text{“}achieved\text{”}, t_1 \rangle)} \{g_i\} \tag{3.22}$$

Active goals. The function $activeGoals(vo,t)$ returns the set of goals whose state is active in the organization $vo \in \mathcal{VO}$ at a time t. Formally,

$$activeGoals : \mathcal{VO} \times T \rightarrow 2^{\mathcal{G}}$$

$$activeGoals(vo,t) = \bigcup_{\forall g_i (g_i = \langle \text{“}active\text{”}, t_1 \rangle)} \{g_i\} \tag{3.23}$$

Failed goals. The function $failedGoals(vo,t)$ returns the set of goals whose state is failed in the organization $vo \in \mathcal{VO}$ at a time t. Formally,

$$failedGoals : \mathcal{VO} \times T \rightarrow 2^{\mathcal{G}}$$

$$failedGoals(vo,t) = \bigcup_{\forall g_i (g_i = \langle \text{“}failed\text{”}, t_1 \rangle)} \{g_i\} \tag{3.24}$$

Role functionality. The function *serviceProviderRole* returns the list of services that are associated with a role r as provider, in a virtual organization vo at a given time t.

$$serviceProviderRole : R \times \mathcal{VO} \times T \rightarrow 2^S$$

$$serviceProviderRole(r, vo, t) = \bigcup_{\forall s_i (s_i \in services(vo,t) \wedge r \in provider(s_i))} \{s_i\} \quad (3.25)$$

where $services(vo, t)$ is defined in Equation 3.19, and $provider(s_i)$ is defined in Equation 3.9.

The function *serviceClientRole* returns the list of services that are associated with a role r as clients, in a virtual organization vo at a given time t.

$$serviceClientRole : R \times \mathcal{VO} \times T \rightarrow 2^S$$

$$serviceClientRole(r, vo, t) = \bigcup_{\forall s_i (s_i \in services(vo,t) \wedge r \in client(s_i))} \{s_i\} \quad (3.26)$$

where $services(vo, t)$ is defined in Equation 3.19, and $client(s_i)$ is defined in Equation 3.8.

The function *serviceRole* returns the list of services that are associated with a role r as client or providers, in a virtual organization vo at a given time t.

$$serviceRole : R \times \mathcal{VO} \times T \rightarrow 2^S$$

$$serviceRole(r, vo, t) = \bigcup_{\forall s_i (s_i \in serviceProviderRole(r,vo,t) \vee s_i \in serviceClientRole(r,vo,t))} \{s_i\} \quad (3.27)$$

Since both clients and providers have different tasks and obligations, it is interesting to be provided with functions that return them separately. Additionally, just in case it is necessary, the function *serviceRole* that returns both types of roles is included.

The function *goalsRole* returns the set of goals that a role r of an organization vo pursues at a time t:

$$goalsRole : R \times \mathcal{VO} \times T \to 2^{G}$$

$$goalsRole(r, vo, t) = \bigcup_{\forall g_i (g_i \in obtains(s_i) \wedge s_i \in serviceRole(r, vo, t))} \{g_i\} \tag{3.28}$$

where $obtains(s_i)$ is defined in Equation 3.10.

Additional restrictions. It must be assured that the provider of a service must be a role contained in the same VO as the service, because the service cannot be provided if there is not an agent playing a provider role. Formally:

$$\forall vo \in \mathcal{VO} \wedge \forall s \in services(vo, t) : provider(s) \subseteq roles(vo, t) \tag{3.29}$$

where $services(vo, t)$ is defined in Equation 3.19, $provider(s)$ is defined in Equation 3.9, and $roles(vo, t)$ is defined in Equation 3.1.

Another key issue for the system designer is to assure that the services located in an organizational unit must achieve any of its goals or a goal of the service need to have a dependency relation with an organizational goal. If not, the service is not providing useful functionality to the organization. Formally, it is described as:

$$\begin{gathered} \forall vo \in \mathcal{VO}, \forall s \in services(vo, t), \exists g_1 \in obtains(s) : \\ g_1 \in goals(vot,) \vee (\exists g_2 \in goals(vo, t) \wedge \exists GoalDependency(g_1, g_2)) \end{gathered} \tag{3.30}$$

where $services(vo, t)$ is defined in Equation 3.19, $obtains(s)$ is defined in Equation 3.10, $goals(vo, t)$ is defined in Equation 3.20, and $GoalDependency(g_1, g_2)$ is defined in Definition 8.

It is possible that a specific goal of a service could not be reached by any of the tasks that compose the service, so this service have to invoke another service which should be able to achieve this desired goal (by means of any of the tasks provided by the invoked service). Formally, it is expressed as:

$$\begin{aligned}\forall s_1 \in services(vo,t) \land \forall g \in obtains(s_1) \rightarrow (\exists ta \in steps(s_1) \land g \in achieves(ta)) \\ \lor(\exists s_2 \in services(vo,t) \land g \in obtains(s_2) \land s_2 \in steps(s_1))\end{aligned} \tag{3.31}$$

where $services(vo,t)$ is defined in Equation 3.19, $obtains(s)$ is defined in Equation 3.10, $steps(s)$ is defined in Equation 3.13, and $achieves(ta)$ is defined in Equation 3.7.

Finally, as pointed out in this section, the $steps(s)$ function (see Equation 3.13) describes the steps (in form of services or tasks) of a service s at a given time t. It must be ensured that the provider role of the invoker service must be a client role of the invoked service. Formally:

$$\forall s_1, s_2 \in services(vo,t), s_2 \in steps(s_1) \land \forall r_1 \in provider(s_1) \rightarrow r_1 \in client(s_2) \tag{3.32}$$

where $services(vo,t)$ is defined in Equation 3.19, $provider(s)$ is defined in Equation 3.9, and $client(s)$ is defined in Equation 3.8.

As summary, the Functional Dimension allows to describe the tasks (including the goals they achieve) and services of the organization (with client and provider roles, the goals it achieves, its preconditions and postconditions, the steps followed by the service, the quality of service, and the cost and benefits of the service and its resources and products). Additionally, the goals of the organization are defined, as well as dependencies between them.

3.3.1.3 Environment Dimension

The Environment Dimension describes the artifacts, i.e., entities that populate the environment of a MAS. This dimension uses the concept of artifact (RVO07), an element introduced by the Agents & Artifacts (A&A) conceptual framework. These elements are employed by agents in order to reach their goals. Additionally, the A&A framework presents the concept of workspace, used to define the topology of the environment of a MAS. The set of all artifacts is represented by $\mathcal{AR}$.

Definition 9. [**Artifact**] *An artifact* $ar \in \mathcal{AR}$ *is defined as* $ar = \langle St, PR, OP, LO, Located\rangle$*, where:*

- *St is a set representing the internal state of an artifact, which contains all the knowledge of the artifact, such as permissions, usage instructions, etc. in the form of a set of statements about the environment. In this case, we consider the internal state of an artifact to be composed of two required relations, plus other values, relations and functions, but it is domain-dependent to determine these complementary elements in order to give a complete specification of the internal state of the artifact. The required relations are* $CheckPr \in St$ *and* $ExecuteOp \in St$*, that are contained into the internal state, and are defined as:*
  - $CheckPr : PR \to 2^{\mathcal{R}}$ *defines the set of roles that can check custom information (i.e., information specifically targeted to a role) by means of the observable properties of the artifact.*
  - $ExecuteOp : OP \to 2^{\mathcal{R}}$ *is the set of roles that are allowed to execute the operations in the artifact.*
- $PR \subseteq St$ *are the observable properties of an artifact that agents can check without executing any operation on it.*
- $OP \subseteq \mathcal{TA}$ *is the set of operations (tasks) that agents can execute when interacting with the artifact.*
- $LO \subseteq \mathcal{TA}$ *refers to the link operations (tasks), which allows the composition and distribution of artifacts.*

- *Located* $\subseteq \mathcal{WS}$ *describes the set of workspaces where an artifact is located.*

Properties of an artifact. The function *properties*(ar) returns the set of properties of the artifact ar. It returns a set $\mathcal{S}t'$, which is a subset of the internal state of the artifact $\mathcal{S}t$ ($\mathcal{S}t' \subseteq \mathcal{S}t$):

$$properties : \mathcal{AR} \to \mathcal{S}t'$$

$$properties(ar) = \bigcup_{\forall pr_i (pr_i \in PR \wedge PR \in ar)} \{pr_i\} \tag{3.33}$$

This is, the properties of the artifact are a subset of its internal state. This means that every type of information contained inside the internal state could be a property of an artifact.

Operations of an artifact. The function *operations*(ar) returns the set of operations of the artifact ar:

$$operations : \mathcal{AR} \to 2^{\mathcal{TA}}$$

$$operations(ar) = \bigcup_{\forall ta_i (ta_i \in OP \wedge OP \in ar)} \{ta_i\} \tag{3.34}$$

Link operations of an artifact. The function *linkOperations*(ar) returns the set of link operations of the artifact ar:

$$linkOperations : \mathcal{AR} \to 2^{\mathcal{TA}}$$

$$linkOperations(ar) = \bigcup_{\forall ta_i (ta_i \in LO \wedge LO \in ar)} \{ta_i\} \tag{3.35}$$

Workspaces of an artifact. The function $locatedArt(ar)$ returns the set of workspaces where an artifact ar is located:

$$locatedArt : \mathcal{AR} \rightarrow 2^{\mathcal{WS}}$$

$$locatedArt(ar) = \bigcup_{\forall ws_i (ws_i \in Located \wedge Located \in ar)} \{ws_i\} \tag{3.36}$$

Definition 10. [Environment Dimension] *The Environment Dimension (ED) of a Virtual Organization $vo \in \mathcal{VO}$ at a given time t is defined as $ED(vo, t) = \langle WSP, WSO, Composition \rangle$ where:*

- *$WSP \subseteq \mathcal{WS}$ is the set of workspaces that build the environment perceived by a virtual organization vo, where a workspace $ws \in WSP$ is defined as $ws = \langle Placed \rangle$, being Placed the value of the absolute location of this workspace inside the environment.*
- *$WSO \subseteq WSP$ is the set of workspaces where a virtual organization vo is located, where a workspace $ws \in WSO$ is defined as $ws = \langle Placed \rangle$, being Placed the value of the absolute location of this workspace inside the environment.*
- *$Composition : WSP \rightarrow 2^{WSP}$ allows defining intersection and nesting relations between workspaces that build the environment. This relation helps building the environment in a similar way as the real world is modeled. A tuple $\langle wsp_1, wsp_2 \rangle \in Composition$ means that the workspace wsp_1 intersects with the workspace wsp_2.*

Properties of the environment. The *Composition* relation is reflexive (Table 3.3, Equation 1) and symmetrical (Table 3.3, Equation 2), since a workspace can intersect with itself and the order of the relation is not relevant and does not affect to the structure of the environment.

Properties
1) $\forall wsp \in WSP : Composition(wsp, wsp)$
2) $\forall wsp_1, wsp_2 \in WSP : Composition(wsp_1, wsp_2) \rightarrow Composition(wsp_2, wsp_1)$

Table 3.3: Properties of the relations

Workspaces of an organization. $workspaces(vo, t)$ returns the set of workspaces where the virtual organization $vo \in \mathcal{VO}$ is located at a time t.

$$workspaces : \mathcal{VO} \times T \rightarrow 2^{\mathcal{WS}}$$

$$workspaces(vo, t) = \bigcup_{\forall wso_i (wso_i \in WSO \wedge WSO \in ED(vo,t))} \{wso_i\} \tag{3.37}$$

Perceived workspaces of an organization. $workspacesPerc(vo, t)$ returns the set of workspaces that the virtual organization $vo \in \mathcal{VO}$ perceives at a time t.

$$workspacesPerc : \mathcal{VO} \times T \rightarrow 2^{\mathcal{WS}}$$

$$workspacesPerc(vo, t) = \bigcup_{\forall wsp_i (wsp_i \in WSP \wedge WSP \in ED(vo,t))} \{wsp_i\} \tag{3.38}$$

Artifacts of an organization. The function $artifactsOrg(vo, t)$ returns the set of artifacts that an organization vo has available (i.e., they are located in the workspaces where the organization is located) at a given time t.

$$artifactsOrg : \mathcal{VO} \times T \rightarrow 2^{\mathcal{AR}}$$

$$artifactsOrg(vo, t) = \bigcup_{\forall ar_i (\exists ws_i (ws_i \in workspaces(vo,t) \wedge ws_i \in locatedArt(ar_i)))} \{ar_i\} \tag{3.39}$$

where $locatedArt(ar)$ was defined in Equation 3.36 and $workspaces(vo, t)$ was defined in Equation 3.37.

Summarizing, the Environment Dimension allows to describe the artifacts located at the environment of the organization (including its properties, operations, and link operations). Also, it describes the workspaces that structure the environment of the organization, the workspaces perceived by the organization, and the artifacts that the organization may use.

3.3.1.4 Normative Dimension

The Normative Dimension describes normative restrictions on the behavior of the entities of the system, including sanctions and rewards, based on the work in (CAB10).

Definition 11. **[Norm]** *A norm $n \in \mathcal{N}$ is defined as $n = \langle D, C, \mathcal{T} \rangle$ where:*

- *$D \in \{\mathcal{O}, \mathcal{F}\}$ is the deontic operator, i.e., obligations ($\mathcal{O}$) and prohibitions ($\mathcal{F}$) that impose restrictions in the behavior of the agents.*
- *$C \in \{\mathcal{TA}, \mathcal{S}\}$ is the task or service that must be carried out in case of obligations ($D = \mathcal{O}$), or has to be avoided in case of prohibitions ($D = \mathcal{F}$).*
- *$\mathcal{T} \in \mathcal{R}$ is the role that is affected by the norm.*

Definition 12. **[Normative Dimension]** *The Normative Dimension (ND) of a Virtual Organization $vo \in \mathcal{VO}$ at a given time t is defined as $ND(vo, t) = \{n_i\}$ where $n_i \in \mathcal{N}$ and $n_i = \langle D, C, \mathcal{T} \rangle$ where:*

- *$D \in \{\mathcal{O}, \mathcal{F}\}$ is the deontic operator.*
- *$C \in \{services(vo, t), \mathcal{TA}\}$*
- *$\mathcal{T} \in roles(vo, t)$*

Organizational norms. The function $norms(vo,t)$ depicts the list of norms followed by the Virtual Organization $vo \in \mathcal{VO}$ at a time t. Formally,

$$norms : \mathcal{VO} \times T \rightarrow 2^{\mathcal{N}}$$

$$norms(vo,t) = \bigcup_{\forall n_i \in ND(vo,t)} \{n_i\} \tag{3.40}$$

Obligation norms. The function $obligationNorms(vo,t)$ returns the set of norms of the organization vo that suppose an obligation.

$$obligationNorms : \mathcal{VO} \times T \rightarrow 2^{\mathcal{N}}$$

$$obligationNorms(vo,t) = \bigcup_{\forall n_i (n_i \in ND(vo,t) \wedge \exists n_i (D \in n_i \wedge D = \mathcal{O})} \{n_i\} \tag{3.41}$$

Prohibition norms. The function $prohibitionNorms(vo,t)$ returns the set of norms of the organization vo that suppose a prohibition.

$$prohibitionNorms : \mathcal{VO} \times T \rightarrow 2^{\mathcal{N}}$$

$$prohibitionNorms(vo,t) = \bigcup_{\forall n_i (n_i \in ND(vo,t) \wedge \exists n_i (D \in n_i \wedge D = \mathcal{F})} \{n_i\} \tag{3.42}$$

Target of a norm. The function $target(n,vo,t)$ returns the role $r \in \mathcal{R}$ that is the target of a norm $n \in ND(vo,t)$.

$$target : \mathcal{N} \times \mathcal{VO} \times T \rightarrow \mathcal{R}$$

$$target(n, vo, t) = r | n \in norms(vo, t) \wedge n = \langle D, C, \mathcal{T} \rangle \wedge \mathcal{T} = r \qquad (3.43)$$

Services associated to a norm. The function $serviceNorm(n, vo, t)$ returns the service $s \in \mathcal{S}$ or task $ta \in \mathcal{TA}$ that is the action to be carried out in case of obligations or to be avoided in case of prohibitions of a norm $n \in ND(vo, t)$.

$$serviceNorm : \mathcal{N} \times \mathcal{VO} \times T \to \{\mathcal{S} \cup \mathcal{R}\}$$

$$serviceNorm(n, vo, t) = a | n \in norms(vo, t) \wedge n = \langle D, C, \mathcal{T} \rangle \wedge C = a \qquad (3.44)$$

As summary, the Normative Dimension describes the norms of the organization, begin possible to distinguish between obligation and prohibition norms, to specify the target of a norm, and the services associated to a norm.

3.3.2 Organizational Entity

The Organizational Entity of a Virtual Organization is the set of entities that populate the system, which can be agents (where the set of all agents is represented by $\mathcal{A}$) or other Virtual Organizations (VO).

Definition 13. [Agent] *An Agent $a \in \mathcal{A}$ that populates a Virtual Organization is characterized in our approach by the elements inside the tuple $a = \langle Goals, Capabilities, Workspaces \rangle$ where:*

- *$Goals \subseteq \mathcal{G}$ is the set of goals that the agent follows.*
- *$Capabilities \subseteq \{\mathcal{S}, \mathcal{TA}\}$ is the set of services and tasks that the agent is able to carry out.*
- *$Workspaces \subseteq \mathcal{WS}$ is the set of workspaces that the agent perceives.*

Goals of an agent. The function *goalsAg* returns the set of goals pursued by an agent a_1:

$$goalsAg : \mathcal{A} \rightarrow 2^{\mathcal{G}}$$

$$goalsAg(a_1) = \bigcup_{\forall g_i (g_i \in Goals \wedge Goals \in a_1)} \{g_i\} \qquad (3.45)$$

Capabilities of an agent. The function *capabilities* returns the set of capabilities (in terms of services and/or tasks) that an agent a_1 is able to carry out:

$$capabilities : \mathcal{A} \rightarrow 2^{\{\mathcal{S}, \mathcal{TA}\}}$$

$$capabilities(a_1) = \bigcup_{\forall ts_i (ts_i \in Capabilities \wedge Capabilities \in a_1)} \{ts_i\} \qquad (3.46)$$

Workspaces of an agent. The function *workspacesAg* returns the set of workspaces that an agent a_1 perceives:

$$workspacesAg : \mathcal{A} \rightarrow 2^{\mathcal{WS}}$$

$$workspacesAg(a_1) = \bigcup_{\forall ws_i (ws_i \in Workspaces \wedge Workspaces \in a_1)} \{ws_i\} \qquad (3.47)$$

It should be noted that an agent could have been defined by more features, but in our proposal we only require the enumerated features.

Definition 14. [Organizational Entity] *The Organizational Entity (OE) of a Virtual Organization vo at a time t is defined as $OE(vo,t) = \{e_i\}$, i.e., the set of entities populating the organization, where $\forall e_i \in OE(vo,t) : e_i \in \mathcal{VO} \vee e_i \in \mathcal{A}$.*

Contained VOs. The function $vos(vo, t)$ returns the set of virtual organizations contained in a virtual organization vo at a given time t. Formally:

$$vos : \mathcal{VO} \times T \rightarrow 2^{\mathcal{VO}}$$

$$vos(vo, t) = \bigcup_{\forall e_i (e_i \in OE(vo,t) \wedge e_i \in \mathcal{VO})} \{e_i\} \tag{3.48}$$

Agents of a VO. The function $agentsVO(vo, t)$ returns the set of agents populating a virtual organization vo at a given time t. It is recursively calculated as the number of agents populating the virtual organization vo, as well as the agents that populate the virtual organizations contained into vo. Formally:

$$agentsVO : \mathcal{VO} \times T \rightarrow 2^{\mathcal{A}}$$

$$agentsVO(vo, t) = \bigcup_{\forall e_i (e_i \in OE(vo,t) \wedge e_i \in \mathcal{A})} \{e_i\} \cup (\forall vo_i \in vos(vo, t) : agentsVO(vo_i, t)) \tag{3.49}$$

Goals of an entity. The function $goalsEn(e_i, t)$ returns the set of goals of the entity e_i at a given time t. They are the goals of the organization, for the goals of a contained VO, or the goals of an agent, depending on the type of entity. Formally,

$$goalsEn : \{\mathcal{VO}, \mathcal{A}\} \times T \rightarrow 2^{\mathcal{G}}$$

$$goalsEn(e_i, t) = \begin{cases} if\ e_i \in \mathcal{A} : goalsAg(e_i) \\ if\ e_i \in \mathcal{VO} : goals(e_i, t) \end{cases} \tag{3.50}$$

where $goalsAg(e_i)$ is defined in Equation 3.45 and $goals(e_i, t)$ is defined in Equation 3.20.

As summary, the Organizational Entity allows to describe the agents and virtual organizations that populate the organization (and the goals they pursue). From each agent, it is possible to get its goals, its capabilities, and the workspaces it perceives.

3.3.3 Organizational Instantiation

The Organizational Instantiation presents the relations between the elements of the Organizational Structure and the Organizational Entity of a Virtual Organization.

Definition 15. [Organizational Instantiation] *The Organizational Instantiation of a Virtual Organization $vo \in \mathcal{VO}$ at a given time t is defined as $\phi(vo,t) = \langle plays, Population, CostPlay, Budget\rangle$ where:*

- *$Plays : \{\mathcal{VO}, \mathcal{A}\} \rightarrow 2^R$, where a tuple $\langle e_i, r_i\rangle$ $(e_i \in OE(vo,t) \wedge r_i \in roles(vo,t))$ is a relationship between an entity of the organization and the set of roles that it plays inside this organization at time t.*
- *$PopulationWS : WSO \rightarrow 2^{\mathcal{A}}$ defines the set of agents that are populating a given workspace, where $WSO \in ED(vo,t)$. A tuple $\langle ws_1, a_1\rangle$ means that the workspace ws_1 is populated by the agent a_1.*
- *$CostPlay : \mathcal{R} \rightarrow \mathbb{R}$ defines the cost of playing a role in the organization.*
- *$Budget : \mathbb{R}$ defines the budget available at the organization.*

Population of a workspace. The function *popWS* returns the agents populating a workspace ws_1 of an organization vo_1 at a given time t:

$$popWS : \mathcal{WS} \times \mathcal{VO} \times T \rightarrow 2^{\mathcal{A}}$$

$$popWS(ws_1, vo, t) = \bigcup_{\forall a_i(\langle ws_1, a_i\rangle \in PopulationWS \wedge PopulationWS \in \phi(vo,t))} \{a_i\} \quad (3.51)$$

Roles played by an entity. The set of roles played by an entity e in the organization vo at a given time t can be obtained by means of the function $playRole(e, vo, t)$:

$$playRole : \{\mathcal{VO}, \mathcal{A}\} \times \mathcal{VO} \times T \rightarrow 2^{\mathcal{R}}$$

$$playRole(e, vo, t) = \bigcup_{\forall r_i(\langle e, r_i\rangle \in Plays \wedge Plays \in \phi(vo,t))} \{r_i\} \tag{3.52}$$

Cost of playing a role. The function $costPlayRole(r_1, vo, t)$ returns the cost of playing a role r_1 in an organization vo at a given time t:

$$costPlayRole : \mathcal{R} \times \mathcal{VO} \times T \rightarrow \mathbb{R}$$

$$costPlayRole(r_1, vo, t) = c | CostPlay(r_1) = c \wedge CostPlay \in \phi(vo, t) \tag{3.53}$$

Entities playing a role. The set of entities that play a role r in the organization vo at a given time t can be obtained by means of the function $rolePlayed(r, vo, t)$:

$$rolePlayed : \mathcal{R} \times \mathcal{VO} \times T \rightarrow 2^{\mathcal{VO} \cup \mathcal{A}}$$

$$rolePlayed(r, vo, t) = \bigcup_{\forall e_i(e_i \in OE(vo,t) \wedge \exists \langle e_i, r_i\rangle \in Plays \wedge Plays \in \phi(vo,t))} \{e_i\} \tag{3.54}$$

Budget of the organization. The function $orgBudget(vo, t)$ returns the budget of the organization vo at a given time t:

$$orgBudget : \mathcal{VO} \times T \rightarrow \mathbb{R}$$

$$orgBudget(vo,t) = b | Budget = b \wedge Budget \in \phi(vo,t) \tag{3.55}$$

Agent offers a service. An agent $a \in \mathcal{A}$ is able to offer a service $s \in \mathcal{S}$ in the organization $vo \in \mathcal{VO}$ at a time t iff it is playing a role $r \in roles(vo,t)$ that is able to provide this service and the agent is capable of offering it. The function *provService* will return the set of services an agent is able to offer.

$$provService : \mathcal{A} \times \mathcal{VO} \times T \rightarrow 2^{\mathcal{S}}$$

$$provService(a,vo,t) = \bigcup_{\forall s_i (s_i \in services(vo,t) : \exists r \in provider(s_i) \wedge r \in playRole(a,vo,t) \wedge s_i \in capabilities(a))} \{s_i\} \tag{3.56}$$

where $services(vo,t)$ is defined in Equation 3.19, $provider(s_i)$ is defined in Equation 3.9, $playRole(a,vo,t)$ is defined in Equation 3.52, and $capabilities(a)$ is defined in Equation 3.46.

Clients of a service. The clients $clientsServ(s,vo,t)$ of a service s are calculated as the set of agents populating the organization vo that play a client role of the service:

$$clientsServ : \mathcal{S} \times \mathcal{VO} \times T \rightarrow 2^{\mathcal{A}}$$

$$clientsServ(s,vo,t) = \bigcup_{\forall a_i (a_i \in agentsVO(vo,t) \wedge \exists r_i \in playRole(a_i,vo,t) : r_i \in client(s) \wedge s \in services(vo,t))} \{a_i\} \tag{3.57}$$

where $agentsVO(vo, t)$ is defined in Equation 3.49, $playRole(a_i, vo, t)$ is defined in Equation 3.52, $client(s)$ is defined in Equation 3.8, and $services(vo, t)$ is defined in Equation 3.19.

Organizational clients. The function $clientOrg(vo, t)$ returns the set of clients of an organization vo at a time t. The clients of an organization are the agents that play a client role of any of the services of an organization.

$$clientOrg : \mathcal{VO} \times T \to 2^{\mathcal{A}}$$

$$clientOrg(vo, t) = \bigcup_{\forall s_i (s_i \in services(vo,t) : \forall a_i \in clientsServ(s_i, vo, t))} \{a_i\} \tag{3.58}$$

where $services(vo, t)$ is defined in Equation 3.19 and $clientsServ(s_i, vo, t)$ is defined in Equation 3.57.

Workspaces populated by an agent. The function $agentWS(a, vo, t)$ returns the set of workspaces populated by an agent a in an organization vo at a given time t.

$$agentWS : \mathcal{A} \times \mathcal{VO} \times T \to 2^{WSO}$$

$$agentWS(a, vo, t) = \bigcup_{\forall ws_i (\exists \langle ws_i, a \rangle \in PopulationWS \wedge PopulationWS \in \phi(vo,t))} \{ws_i\} \tag{3.59}$$

As summary, the Organizational Instantiation allows to define the population of a workspace, the set of roles played by an entity (and the cost of playing each role), the set of entities that play a given role, and the budget that the organization has available. Additionally, it is able to describe the agents that offer or are clients of a service, the workspaces populated by an agent, and the set of clients of the organization.

3.3.4 Organizational Dynamics

In previous sections, the different dimensions and entities that compose the state of a VO at a given time were defined in a formal way. Since a VO can change through time, passing from one state of the organization to another, it is necessary to define all the possible states of the organization as well as the allowed transitions between these states. For this issue, we based our work in the proposal by da Rocha Costa and Dimuro (dRCD08).

To model the states of a VO and their dynamics, let VO be the universe of all possible organizations O. A multi-agent system based on virtual organizations is a structure $MAS = (VO, D)$ where, for every time $t \in T, D^t \subseteq VO \times VO$ defines transitions between different states of the system. In every state of the organization $O \in VO$, at a given time $t \in T$, there is a set of possible next states of the organization, denoted by $D^t(O) \subseteq VO$. Thus, for every $t \in T$, it holds that $O^{t+1} \in D^t(O^t)$, so an organization will only change to another state when it is allowed to reach it from the initial state.

Since the organization is composed by three elements (OS, OE and ϕ), before executing a change of state it is necessary to check that these elements are able to change from the initial state to the possible destination state. Formally:

$$((OS^{t+1}, OE^{t+1}, \phi^{t+1}) \in D^t(OS^t, OE^t, \phi^t)) \leftrightarrow$$

$$((OS^{t+1} \in D^t_{OS}(OS^t)) \wedge (OE^{t+1} \in D^t_{OE}(OE^t)) \wedge (\phi^{t+1} \in D^t_{\phi}(\phi^t)))$$

However, in order to swap from one state to another, it is not necessary to produce a change in all three elements that compose the VO. A change ranges from a very small variation in one or few of the elements building the organization to a big amount of changes in a large amount of entities from the VO. Formally:

$$(O^t \wedge \bigcirc O^{t+1}) \rightarrow \Box((OS^t \wedge \bigcirc OS^{t+1}) \vee (OE^t \wedge \bigcirc OE^{t+1}) \vee (\phi^t \wedge \bigcirc \phi^{t+1})) \quad (3.60)$$

where $\bigcirc$ means that the expression after this operation must be true at the next state, and $\Box$ means that the expression must be true at every state. This means that for moving from a state O^t to a state O^{t+1} a change may happen only in one of the three elements that define a VO, the OS, the OE, or ϕ.

The above formula (that uses LTL (Pnu77)) helps us to formalize how D^t is build.

$$D^t = \bigcup (O^t, O^{t+1}) | O^{t+1} \neq O^t \quad (3.61)$$

D^t is composed of the set of all possible transitions of the MAS, where each new transition is generated every time that in OS, OE or ϕ (or in a combination of these elements) an atomic change is produced.

3.4 Case of study

In this section, we will study the situation of a company that can be modeled as an Adaptive Virtual Organization.

Our example here is the ACME company, which is a family business owned by John and Jane Doe, a couple from Springfield, USA. It manufactures computer processors, and although competence is high, they have signed some important contracts with retailers and computer manufacturers around the world to sell them ACME-branded products. At the top of the organizational hierarchy is Jane, which is the President of ACME (i.e., she is playing the role President), while John is playing the CEO role, the must powerful role of the organization, and the one which has the most power in the organization. After them, Alice, the daughter of the couple, is the supervisor of the marketing department of the company, while Bob (Alice's brother) holds the manager role in the manufacturing department. In lower levels of the hierarchy there are other intermediate managers and supervisors, as well as subordinated agents. This company has been modeled using VOF.

We will include a simplified definition of ACME. Note that if there are elements not depicted in this description is not because they do not exist, but they are not relevant for this example. The ACME company is defined with VOF as: $ACME(t) = \langle OS(\text{“}ACME\text{”}, t), OE(\text{“}ACME\text{”}, t), \phi(\text{“}ACME\text{”}, t)\rangle$, where:

- $OS(\text{“}ACME\text{”}, t) = \langle SD(\text{“}ACME\text{”}, t), FD(\text{“}ACME\text{”}, t), ED(\text{“}ACME\text{”}, t), ND(\text{“}ACME\text{”}, t)\rangle$, where:
  - $SD(\text{“}ACME\text{”}, t) = \langle R, \mathit{Relations}\rangle$, where $R = \langle \mathit{President}, CEO, Manager\rangle$ is the set of roles, and $\mathit{Relations} = \langle \mathit{mon}(\mathit{President}, Manager), \mathit{mon}(CEO, Manager)\rangle$ are the relationships between roles
  - $FD(\text{“}ACME\text{”}, t) = \langle G, S, \mathit{GoalDependency}\rangle$, where $G = \langle MaxProfit, \mathit{MakeProcessors}\rangle$ is the set of goals, and $S = \langle Manufacture, \mathit{Sell}\rangle$ is the set of services, and $\mathit{GoalDependency} = \emptyset$
  - $ED(\text{“}ACME\text{”}, t) = \langle WSP, WSO, \mathit{Composition}\rangle$, where $WSP = \langle ws_1, ws_2\rangle$ is the set of workspaces that the organization perceives, $WSO = \langle ws_1\rangle$ is the set of workspaces where the organization is located, and $Composition = \emptyset$
  - $ND(\text{“}ACME\text{”}, t) = \langle \mathit{ApproveDecision}\rangle$ where $\mathit{ApproveDecision} = \langle \mathcal{O}, ApprTakenDecision, CEO\rangle$, that states that it is an obligation for agents playing the CEO role to carry out the task $ApprTakenDecision$. This is, to approve (or not), any decision taken in the organization.

- $OE(\text{"}ACME\text{"}, t) = \langle John, Jane, Alice, Bob \rangle$ are the entities populating the organization, which are the members of the family
- $\phi(\text{"}ACME\text{"}, t) = \langle Plays, Population, CostPlay, Budget \rangle$, where:
  - $Plays = \langle \{Jane, President\}, \{John, CEO\}, \{Alice, Manager\}, \{Bob, Manager\} \rangle$ shows the roles each agent is playing
  - $Population = \langle \{ws_1, \{John, Jane, Alice, Bob\}\} \rangle$ shows which agents are populating each workspace
  - $CostPlay = \langle \{Manager, 100\}, \{CEO, 500\}, \{President, 300\} \rangle$ are the costs of playing each role of the organization
  - $Budget = 100000$

The $Manufacture$ service is defined as $Manufacture = \langle Profile, Process, Performance \rangle$, where:

- $Profile = \langle Client, Provider, Input, Output, Preconditions, Postconditions \rangle$ where:
  - $Client = Manager$, the role that may request the service.
  - $Provider = CEO$, the role that provides the service.
  - $Input = \text{"}TypeOfMicroprocessor\text{"}$, the input required by the service. In this case, the type of microprocessor to be manufactured.
  - $Output = \emptyset$, since there is no value returned by the service.
  - $Preconditions = \text{"}Silicon\text{"}$, the resource required by the service prior to manufacturing a microprocessor.
  - $Postconditions = \text{"}Microprocessor\text{"}$, the product generated by the service.

- $Process = \langle Steps, Comm \rangle$, where:
  - $Steps = \{ProcessSilicon, AddTransistors, MakeDetails\}$ is an ordered list that contains the tasks of the service.
  - $Comm =$ "$KQML$" is the agent communication language used by this service.
- $Performance = \langle QoS, CostRes, BenPro \rangle$, where:
  - $QoS = 0.9$, assuring that at least the 90% of requests to the service have to be properly executed.
  - $CostRes = 235$ is the number of units spent when this service executes.
  - $BenPro = 342$ is the number of units that the organization gets when executing the service.

The precondition "*Silicon*" is a resource that is defined as a basic artifact as:

$$Ar_{silicon} = \langle \emptyset, SiliconProperties, \emptyset, \emptyset, wsp_1 \rangle$$

where $SiliconProperties$ are the physical properties of this material, and wsp_1 is the workspace where the resource is located.

This example depicts that it is possible to define an organization using VOF with all its elements. It could also be checked that a description using VOF does not lose expressivity from a description using natural language. This formalization helps to structure the information about the organizational description. Having a formal and structured description of the organization facilitates the task of describing, identifying, and solving the forces that drive organizational change in a VO. In the next chapter, the description of these forces can be found, as well as an example on how one of the forces is first identified and then solved in the ACME organization.

3.5 Discussion

The *Virtual Organization Formalization* (VOF) takes inspiration from features taken from some of the most relevant formalization proposals, which were analyzed in Chapter 2. Here, we depict (see Table 3.4) a comparison between VOF and these background proposals.

Organizational concepts					
	OperA	MOISE^{Inst}	PopOrg	POMF	VOF
Structural Dimension					
Roles	✓	✓	✓	✓	✓
Groups	✓	✓			
Agents		✓	✓	✓	✓
Relations	✓	✓	✓		✓
Topology	✓				✓
Functional Dimension					
Capabilities			✓	✓	✓
Abilities			✓		
Services			✓	✓	✓
Objectives	✓	✓		✓	✓
Dynamical Dimension					
Interactions	✓	✓	✓		
Dynamics	✓	✓	✓		✓
Environment Dimension					
Environment		✓			✓
Resources				✓	✓
Normative Dimension					
Norms	✓	✓			✓

Table 3.4: Comparison between different formal representations

Firstly, the organizational temporal evolution proposed by VOF is mainly based on PopOrg (dRCD08), which models the dynamics of the *population* (similar to our OE) and the *organization* (similar to our OS).

Regarding the Structural Dimensions, OperA (Dig03) offers relationships between roles that are similar to those included in VOF. The supervision relationship of VOF is similar to the combination of the power and authorization relationships of OperA, expressing that an agent is able to delegate its objectives to a subordinated agent, like the power relation does (the authorization relation expresses the power relation, but as a temporal situation). Also, in OperA, the objective that a subordinated agent can take from a superior agent is determined by the type of existing relations between roles, which establishes their hierarchy. However, VOF defines this hierarchy using the $RoleHier$ relation.

In MOISE^{Inst} (GBKD05), the structural levels of the organization are split into: (i) *individual level*, built by organizational roles, and presents hierarchy relations between roles (similar to our $RoleHier$ relation); (ii) *social level*, which is built from link relationships between roles, classified as *acq* (*acquaintance*), i.e., having a representation of other agents, *com* (similar to our inf relation), in which agents are able to communicate between them, and *aut* expressing authority over other agents, which can be seen as a combination of *mon* and *sup* relations of VOF; and (iii) *collective level*, which defines *groups* of agents, establishing the *compatibility* between roles and their *cardinalities.*

Regarding the Functional Dimension, PopOrg focuses on the actions developed by the processes and the agents that are carrying them out. POMF (PS06) models concepts that are similar to services, focusing on describing the tasks that compose a given workflow. However, VOF goes beyond (as it follows a Service Oriented Approach (Pap03)) and formalizes a service by means of the roles that provide and consume it, the goals that can be achieved with this service, the invocations between services (allowing a service to call another service to complete its execution), and the tasks that compose each service (as well as the goals that these tasks help to reach).

The Environment Dimension used in VOF is based on the Agents & Artifacts conceptual framework (RVO07), which was included in the SODA metamodel

(NMOD08).

Finally, the Environment Dimension of VOF models norms in a very similar way to the proposal of MOISE^{Inst}, although they use different languages to describe them. VOF is able to relate a norm to a set of VOs, using the target of each norm in the Normative Dimension, limiting its effect only to the target.

Finally, VOF clearly divides the elements that specify the system (i.e. elements in the Organizational Specification, which will produce a structural change if they are modified) and the more dynamic elements of the MAS, represented in the OE.

3.6 Conclusions

This chapter presents a formal specification for Virtual Organizations, named VOF (*Virtual Organization Formalization*), which is composed of: (i) the Organizational Specification (OS), which details the components that specify the system and divides them by means of the organizational dimensions; (ii) the Organizational Entity (OE), which defines the active elements of the system; and (iii) the Organizational Instantiation (ϕ), which details the relationships between elements from OS and OE.

This formalization will help us when dealing with concepts related to Organization-Centered Multi-Agent Systems, because VOF facilitates the process of defining an OCMAS following this formal approach. Additionally, once provided with a formally defined OCMAS it is easier to identify the elements that would change during the life of the organization. As shown in the example of this chapter, an organization can be represented using VOF. In following chapters, the forces that drive organizational change will be defined using VOF, and also a case of study featuring reorganization will be defined using VOF.

4

Design of Forces Driving Adaptation of Agent Organizations

Adaptation is an important feature of human organizations. Being able to adapt allows organizations not only to survive, but to evolve to get new advantages from new situations happening in their environment or from inside the organization. The same way human organizations do, agent organizations should be able to adapt. Even if adaptation is addressed in the MAS literature (Liu01), it lacks the ability to clearly manage the reasons for change. These reasons are known in the social science bibliography as forces that drive the organizational change (Ald99). In this chapter, a set of templates is presented to define these forces at the design time. These templates have been applied in the design of components for detecting the external and internal forces.

4.1 Introduction

As stated in the studies of human organizations by Organizational Theory (OT) (PW71), organizations are dynamic and able to adapt at runtime. The OT is one of the inspirations for Organization-Centered Multi-Agent Systems (OCMAS) (FGM03) developers. Multiple proposals have been presented to design and implement such systems, like (DOM08, DD06, HJST07), most of them focusing on the way the adaptation of the organization is done, or the cost of this adaptation in the organization. However, they do not take completely into account the property of dynamism of human organizations, leaving aside the forces that drive organizational change, which is an important concept when dealing with adaptation. These forces have been widely studied by Social Sciences researchers such as Aldrich (Ald99) and Lewin[1] (Lew51). Aldrich classifies forces into *external* and *internal* forces, depending on where the pressure for change comes from. A force is external if the reason for change comes from the organizational environment, and internal if this reason comes from inside the organization. Forces are characterized by means of the condition factors detecting the action of the force over the organization, and the solutions for reacting to the force that it is affecting the organization.

The objectives of this chapter are: (i) to define a set of guidelines for describing the forces that drive organizational change at design time, including the factors and solutions that characterize them, and (ii) to show how this description facilitates the implementation of forces.

The rest of this chapter is structured as follows: Section 4.2 presents the templates to describe a force at design time. Section 4.3 presents the definition of forces at design time. Section 4.4 describes an example on forces based on the ACME

[1]Lewin states that change is only carried out if the forces supporting the change are stronger than the forces against the change.

example (described in Chapter 3). Finally, Section 4.5 presents the conclusion of this chapter.

4.2 Templates to define forces

Force detection has to be carried out along the organizational life-cycle. For that purpose, guidelines may help developing tools to identify the factors and the solutions of forces. The *factors* express the conditions for stating whether the force is currently active or not. The *solutions* express the actions to execute in the organization in order to either take advantage of the benefits that the force may imply, or minimize the possible damages produced by the force in the organization. In this chapter we propose templates for identifying and describing the factors and solutions of forces. Their contents are formal, closer to an actual implementation. A force is defined by means of a template, where the factors and solutions are pointed out. Factors and solutions are described in separate tables (one for each factor or solution) making possible to reuse a factor or a solution in the definition of another force.

A **force** (Table 4.1) is defined by its name, a textual description, a type stating if the force is internal or external, a set of factors participating on the detection of the action of the force, a force detection condition (which is a boolean expression bearing on the factors), a set of solutions and a selection criteria among the solutions.

Field	**Description**
Name	Name of the force.
Description	Textual description of the force acting over the organization.
Type	*Internal* or *external*.
Factors	Names of factors involved in the detection of the force.
Force detection condition	Logical combination of factors stating that the force is in action.
Solutions	Solutions that can be applied in case the force is active.
Force Solution Condition	Depicts how a solution is chosen.

Table 4.1: Force description template

4.2.1 Factors Stating that a Force is Acting

Table 4.2 defines the components of the factors for expressing the conditions testing that a force is acting over the organization. A factor is implemented as a set of monitoring mechanisms in the organization to detect if the force is acting, and it is characterized by its name, a description, the parameters referring to organizational values that help on the detection of the factor, the condition that states when the factor is active, and the Monitor element of the organization, which can be a role, a specific agent, or an artifact, in charge of monitoring if the detection condition holds.

Factor	
Name	The name of the factor that helps identifying the force.
Description	Textual description of the factor.
Parameters	Organizational elements concerned by the detection of the action of the force, which help on the detection of its action.
Condition	The condition stating the relations among the parameters that help in the detection of the action of the force.
Monitor	The role or entity of the organization responsible of monitoring the force.

Table 4.2: Force factor description template

4.2.2 Solutions to face the Force

Table 4.3 defines the solutions, i.e., the actions that should be carried out in the organization in order to take advantage or to prevent damage from the force that has been detected. Each solution is described by a name, a textual description, a condition that points out the particular factors that need to be satisfied to execute this solution, the parameters involved in the actions of the solution, the actions to be executed to apply the solution, the cost generated when applying this solution, and the roles of the organization that will be in charge of executing the solution.

Field	**Description**
Name	Name of the solution.
Description	Text describing this solution.
Condition	The condition (related to factors) that must hold in order to apply this solution.
Parameters	Describes the elements that have to be known prior to apply the solution.
Action	The set of actions that must be carried out to apply this solution.
Cost	The cost of applying this solution to the organization.
Executor	The responsible roles for applying this solution.

Table 4.3: Force solution description template

4.2.2.1 Selection between solutions

As stated in (AJGF11), to select between different options for change it is necessary to have a utility function that has to express the costs and benefits (both direct and indirect) of both the current and future states of the system, as well as the adaptation costs.

In some situations two forces may apply their solutions to the same organizational elements, thus being necessary to take a decision about which option to take. For this reason, applying the solution for one of these forces may make the organization to solve the effects of both forces. Therefore, in order to exactly choose one of those solutions, the one which maximizes the utility is selected. The utility of a solution *sol* is calculated as:

$$utility_{sol} = benefit_{sol} - cost_{sol} \tag{4.1}$$

where $benefit_{sol}$ is the benefit of the solution when applying the solution sol, and $cost_{sol}$ is the cost generated by the organization after applying the solution sol.

4.3 Description of Forces at the Design Step

Following the templates defined in Section 4.2, forces will be described using a formal language based on VOF, which will connect the description of the forces and the organizational definition. Each element of the organizational model has associated properties or functions. They are accessed using the notation *element.property* (for accessing properties) or *element.function(parameters)* (for accessing functions). Note that both the tables for describing a factor and a solution have an element each to describe the responsible for monitoring the factor and to apply the solution, respectively. Since the definition of forces in this chapter is intended to be generical, in most of the forces the 'Monitor' is a factor referred as 'Monitor element'. This element refers to an entity, which may be an agent of the organization or an artifact, in charge of carrying out the monitoring of the factor. In the case of the 'Executor' of a solution, most forces refer to an 'Organizational Manager', which is an agent playing a management role in the organization. This role has to provide the agent with enough power to develop changes in the organization.

Before analyzing each force, two tables are presented (Tables 4.4 and 4.5), containing the descriptions of elements, operations, etc. that appear during the description of the different forces. In these tables, a relation is carried out among the elements of the Virtual Organization Formalization (described in Chapter 3) and the definition made when designing a force that drives organizational change.

Table 4.4 shows a description of the organizational elements that can be affected by the forces or can be required for detecting them.

Organizational elements		
Attribute	**Attribute VOF**	**Description**
$org_i.Goals$	$goals(org_i, t)$	Set of goals pursued by the organization org_i.
$org_i.Roles$	$roles(org_i, t)$	Set of roles contained by the organization org_i.
$org_i.Services$	$services(org_i, t)$	Set of services of the organization org_i.
$org_i.Environment$	$ED(org_i, t)$	Environment of the organization org_i.
$org_i.Clients$	$clientOrg(org_i, t)$	Clients of the organization org_i.
$org_i.OUs$	$vos(org_i, t)$	Set of organizational units (i.e., virtual organizations) inside org_i.
$org_i.Agents$	$agentsVO(org_i, t)$	Set of agents inside org_i.
$org_i.Norms$	$norms(org_i, t)$	Set of norms that rule org_i.
$org_i.Population$	$\|agentsVO(org_i, t)\|$	Number of agents participating inside org_i.
$org_i.Budget$	$orgBudget(org_i, t)$	Returns the budget of an organization org_i.
$s_i.QoS$	$qos(s_i)$	Quality of the service s_i.
$s_i.Products$	$postconditions(s_i)$	Set of products generated by the service s_i.
$s_i.Goals$	$obtains(s_i)$	Set of goals that the service s_i allows to achieve.
$s_i.Resources$	$preconditions(s_i)$	Set of resources used by the service s_i.
$s_i.Cost$	$costServ(s_i)$	Returns the cost generated by the service s_i.

Table 4.4: Organizational elements

Table 4.5 describes the operations that have an effect on organizational elements and are used for applying the changes caused by the forces. To structure these actions in an adaptation process one may use sequence operator (;), choice operator (|), parallel execution (||), optional execution ([ ]), iteration (a^n) to order the different actions when adapting.

We have employed here a different way to represent the forces (and not directly VOF) because the description of the forces is intended to be closer to the implementation step, whereas VOF is mainly thought for the analysis and design step. However, it can be easily checked that both representations are equivalent.

Organizational functions		
Operation	**Function VOF**	**Description**
$org_i.OUs.Add(ou_i)$	$OE(org_i, t+1) = OE(org_i, t) \cup ou_i$	Organizational unit ou_i added to the organization org_i.
$org_i.OUs.Delete(ou_i)$	$OE(org_i, t+1) = OE(org_i, t) - ou_i$	Organizational unit ou_i deleted from the organization org_i.
$org_i.Agents.Add(a_i)$	$OE(org_i, t+1) = OE(org_i, t) \cup a_i$	Agent a_i added to the organization org_i.
$org_i.Agents.Delete(a_i)$	$OE(org_i, t+1) = OE(org_i, t) - a_i$	Agent a_i deleted from the organization org_i.
$org_i.Agents.Sanction(a_i)$	-	Agent a_i of the organization org_i is sanctioned.
$org_i.Roles.Add(r_i)$	$roles(org_i, t + 1) = roles(org_i, t) \cup r_i$	Role r_i added to the organization org_i.
$org_i.Roles.Delete(r_i)$	$roles(org_i, t + 1) = roles(org_i, t) - r_i$	Role r_i deleted from the organization org_i.
$org_i.Services.Add(s_i)$	$services(org_i, t + 1) = services(org_i, t) \cup s_i$	Service s_i added to the list of services of the organization org_i.
$org_i.Services.Delete(s_i)$	$services(org_i, t + 1) = services(org_i, t) - s_i$	Service s_i deleted from the list of services of the organization org_i.
$org_i.Norms.Add(n_i)$	$norms(org_i, t + 1) = norms(org_i, t) \cup n_i$	Norm n_i added to the set of norms of the organization org_i.
$org_i.Norms.Delete(n_i)$	$norms(org_i, t + 1) = norms(org_i, t) - n_i$	Norm n_i is deleted from the set of norms of the organization org_i.
$org_i.MergeOrgs(org_j, org_k)$	-	A new organization org_i is created after merging two organizations org_j and org_k.
$org_i.Cost(r_i)$	$\sum_{\forall e_i \in rolePlayed(r_i, org_i, t)} costPlayRole(r_i, org_i, t)$	Returns the cost generated by the entities playing the role r_i in an organization org_i.
$s_i.Products.Add(p_i)$	$postconditions(s_i, t + 1) = postconditions(s_i, t) \cup p_i$	Product p_i added as a product generated by the service s_i.
$s_i.Products.Delete(p_i)$	$postconditions(s_i, t + 1) = postconditions(s_i, t) - p_i$	Product p_i removed as a product generated by the service s_i.
$a_i.Roles.Get(r_i)$	$playRole(a_i, org_i, t + 1) = playRole(a_i, org_i, t) \cup r_i$	Agent a_i starts to play the role r_i.
$a_i.Roles.Leave(r_i)$	$playRole(a_i, org_i, t + 1) = playRole(a_i, org_i, t) - r_i$	Agent a_i leaves the role r_i it is playing.
$Monitor.Requests(s_i, t_1, t_2)$	-	Number of requests received in a service s_i during a time interval $[t_1, t_2]$.
$Monitor.Failures(s_i, t_1, t_2)$	-	Number of failures produced when requesting a service s_i during a time interval $[t_1, t_2]$.
$Monitor.Violations(a_i, n_i)$	-	true if the $Monitor$ detects that the agent a_i violated the norm n_i, false otherwise.

Table 4.5: Organizational operations

4.3.1 External forces

This section describes from an OCMAS perspective the set of external forces. They are named in this way because from a human-organization perspective the pressure for change comes from elements outside the organization such as other organizations or the environment. These forces are: (i) obtaining resources; (ii) market; (iii) specialist to generalist; (iv) decay and deterioration; (v) technological changes; (vi) competition; (vii) demographical features; (viii) laws and regulations; and (ix) integration and externalization processes.

4.3.1.1 *Obtaining resources* Force

The execution of a service of an organization fails if it is not able to achieve its preconditions (i.e., resources). In this case, it is necessary to take an action by means of extending the organization to a workspace where it is possible to achieve the preconditions of the services, or to reach an agreement with an agent that may bring the required resource. The Obtaining Resources external force is described in Table 4.6. Its triggering factor is $FailedServiceCallsRate$ and two solutions are defined: $ReachAgreement$ and $ExtendOrganization$. The solution to be applied is the one that would maximize the utility of the resulting organization. Additionally, it is possible that not applying any solution ($noSol$) would result in a better utility than the utilities that other solutions would generate. Therefore, no solution would be applied.

Field	**Description**
Name	ObtainingResources
Description	A resource cannot be achieved by a service of an organization.
Type	External
Factors	$FailedServiceCallsRate$
Force detection condition	$FailedServiceCallsRate.Condition = TRUE$
Solutions	$ReachAgreement$, $ExtendOrganization$
Force solution condition	$argmax_{x \in \{ReachAgreement, ExtendOrganization, noSol\}} utility_x$

Table 4.6: Definition of the *Obtaining resources* force

FailedServiceCallsRate Factor.

This factor (Table 4.7) takes into account the failures of the services in a time interval $[t_1, t_2]$ and is activated if the failure rate is higher than 1 - QoS. The Quality of Service (QoS) (Pap03) defines the expected success rate when calling a service. For example, in the case of having a QoS of 90%, the maximum allowed failure rate is 10%. Each service has a different QoS, so the factor will be differently triggered depending on each service.

To detect whether the factor is active in the organization, the number of requests for a service and the number of failures of such requests by a service are considered. In our application, a request is defined as a tuple $r = \{service, time, status\}$. Let us define the set $R_{t_1,t_2,Service} = \{r\}$ as the set of requests received by the service $Service$ in the period of time $[t_1, t_2]$, and the set $R'_{t_1,t_2,Service} = \{r \in R_{t_1,t_2,Service} | r.status = fail\}$ that records the number of failures on requests to the service $Service$ during the same period of time. The failure rate for the service $Service$ for a time period $[t_1, t_2]$ is calculated as:

$$Monitor.Failures(Service, t_1, t_2) = \begin{cases} \frac{|R'_{t_1,t_2,Service}|}{|R_{t_1,t_2,Service}|} : |R_{t_1,t_2,Service}| \neq 0 \\ 0 \ : \ |R_{t_1,t_2,Service}| = 0 \end{cases} \tag{4.2}$$

where *Service* is a parameter representing a service of the organization. Then, if the failure rate is higher than the expected one, it is possible to apply one of the two possible solutions (or maybe not to apply any solution).

Factor	
Name	FailedServiceCallsRate
Description	If the rate of failures of calls to a service is higher than the allowed failure rate threshold, then the force is considered as acting.
Parameters	$Service \in org_i.Services$, t_1 ,t_2
Condition	$Monitor.Failures(Service, t_1, t_2) > ((1 - Service.QoS) * Monitor.Requests(Service, t_1, t_2))$
Monitor	Monitor element

Table 4.7: Description of the *FailedServiceCallsRate* factor

ReachAgreement solution.

The first solution for solving this force (Table 4.8) is to reach an agreement with an external agent a_i that has to be able to bring the resource to the organization. If the agreement is achieved, this agent a_i will be added into to the organization org_i by using the action $org_i.Agents.Add(a_i)$, and then it has to play a role r_i in the organization. The cost of this solution is the cost of adding the new agent a_i in the organization plus the cost of playing the role r_i by this agent. Finally, the executors of this solution are the organizational managers.

This solution is connected to the joint ventures and outsourcing from integration and externalization processes, but given the nature of OCMAS the solution is considered to be interesting to be used here.

Field	**Description**
Name	ReachAgreement
Description	The organization reaches an agreement with an external agent to bring the required resource to the organization.
Condition	FailedServiceCallsRate
Parameters	$a_i \in \mathcal{A}, r_i \in org_i.Roles$
Action	$org_i.Agents.Add(a_i); a_i.Roles.Get(r_i)$
Cost	$cost(org_i.Agents.Add(a_i)) + cost(a_i.Roles.Get(r_i))$
Executor	Organizational managers

Table 4.8: *ReachAgreement* solution

ExtendOrganization solution.

The second solution proposed by this force (cf. Table 4.9) is to extend the organization to a workspace where it is easier to get the required resources. This change will suppose adding a number of n organizational units (OUs) to the organization org_i in order to physically extend the organization and reaching more workspaces of the environment. For each organizational unit ou_i to be created, first it has to be added to the organization org_i by using $org_i.OUs.Add(ou_i)$. Then a new internal agent $Agent_i$ is added to this organizational unit ($ou_i.Agents.Add(Agent_i)$), and finally the role $r_i \in org_i.Roles$ is assigned to this newly created agent. The cost of this solution is calculated as the cost of creating all the new OUs. Creating an OU implies adding a new organizational unit ($cost(org_i.OUs.Add(ou_i))$, a new internal agent that manages the added organizational unit ($cost(ou_i.Agents.Add(Agent_i))$) to control the added workspace, and also includes the cost of this agent playing a specific role ($cost(Agent_i.Roles.Get(r_i))$). Therefore, this cost is calculated as:

$$\begin{gathered} cost(ExtendOrganization) = n * (cost(org_i.OUs.Add(ou_i)) + \\ cost(org_i.Agents.Add(Agent_i)) + cost(Agent_i.Roles.Get(r_i))) \end{gathered} \tag{4.3}$$

Field	Description
Name	ExtendOrganization
Description	The organization is extended with new organizational units that allow getting the required resources.
Condition	FailedServiceCallsRate
Parameters	org_i, $NewOUs$, $Agent_i$
Action	$\forall ou_i \in NewOUs : org_i.OUs.Add(ou_i); org_i.Agents.Add(Agent_i); Agent_i.Roles.Get(r_i)$
Cost	$n * (cost(org_i.OUs.Add(ou_i)) + cost(org_i.Agents.Add(Agent_i)) + cost(Agent_i.Roles.Get(r_i)))$
Executor	Organizational manager

Table 4.9: *ExtendOrganization* solution

4.3.1.2 *Market* Force

This force appears in the organization when there is one service of the organization that is not receiving enough requests for it, as measured by the entities that monitor the organization. This force is detected by one factor ($ServiceRequests$), and one solution is available ($DeleteService$) as shown in Table 4.10. Therefore, the 'Force Solution Condition' is empty because it is not necessary to choose between solutions because only one solution is possible to be used.

Field	Description
Name	MarketForces
Description	One of the services of the organization has a low number of requests.
Type	External
Factors	$ServiceRequests$
Force detection condition	$ServiceRequests.Condition = True$
Solutions	$DeleteService$
Force solution condition	-

Table 4.10: Definition of the *Market forces*

ServiceRequests factor.

This factor (see Table 4.11) checks that the number of requests received by any organizational service $s_i \in org_i.Services$ is below a specified threshold. Maintaining a service inside an organization generates a cost in this organization because of the agents that have to provide the service, the computational time it requires, the resources required by the service, etc. Therefore, a minimum number of requests per time interval th_{minReq} is defined by organizational managers or designers, and this threshold represents the minimum acceptable value for the number of requests of a service. Requests to this service are monitored by the $Monitor$ entity, which tracks events of the organization, like the number of requests received by a service. Therefore, $Monitor.Requests(s_i, t_1, t_2)$ represents the number of requests received by a service s_i during a time interval $[t_1, t_2]$.

Factor	
Name	ServiceRequests
Description	If the number of requests for a service is lower than a specific value, then this force is acting.
Parameters	th_{minReq}, t_1, t_2, org_i
Condition	$\exists s_i \in org_i.Services : Monitor.Requests(s_i, t_1, t_2) < th_{minReq}$
Monitor	Monitor element

Table 4.11: Description of the *ServiceRequests* factor

DeleteService solution.

Once the factor $ServiceRequests$ has been identified to be active it is necessary to take an appropriate action, as defined in Table 4.12. Since the organizational service s_i is not providing enough utility to the organization so as to be considered as useful, because it is generating an extra cost for the organization, then the service must be deleted from the organization ($org_i.Services.Delete(s_i)$) to prevent more resources to be spent when keeping the service active without receiving enough requests.

Field	**Description**
Name	DeleteService
Description	A service with a low utility is deleted from the organization.
Condition	ServiceRequests
Parameters	$s_i \in org_i.Services$, org_i
Action	$org_i.Services.Delete(s_i)$
Cost	$cost(org_i.Services.Delete(s_i))$
Executor	Organizational manager

Table 4.12: *DeleteService* solution

4.3.1.3 *Specialist to generalist* Force

In some cases, services are not appealing to a big number of clients since the products they are offering are very specialized and specific, so they receive a small number of requests. Differently from the 'Market force', in this case this force takes into account the sum of all the requests received by the whole organizational services. Therefore, a solution for appealing a higher number of clients and for increasing the number of requests received by the organizational services is to add or modify the products of the existing services so as to make them more general, so then being able to attract a wider range of clients. Additionally, there is also another solution that implies adding a new, generic organizational service that has to be able to appeal a wider range of clients. As a result, the factor to detect this force is the number of requests received by the services of the organization ($OrgRequests$), and there are two solutions: modifying the products generated by existing services of the organization ($ModifyProducts$) and adding new services ($AddServices$). The chosen solution would be the one that maximizes the utility of the organization when applied. This force is described in Table 4.13, and its factors and solutions are described as follows.

Field	**Description**
Name	SpecialistToGeneralist
Description	The products offered by the organizational services are so specialized that they cannot get enough clients.
Type	External
Factors	$OrgRequests$
Force detection condition	$OrgRequests.Condition = True$
Solutions	$ModifyProducts$, $AddServices$
Force solution condition	$argmax_{x \in \{ModifyProducts, AddServices\}} utility_x$

Table 4.13: Definition of the *Specialist to generalist* force

OrgRequests factor.

The factor to take into account in this force is the number of requests received by the organizational services (see Table 4.14), because a set of services that is too specialized may not appeal an enough number of requests so as to make it worthy and profitable. As explained in the 'Market forces' description, the Monitor entity tracks the number of requests received by an organizational service s_i during a time interval $[t_1, t_2]$. In the specific case of this force, the sum of the requests to each organizational service is calculated, and then compared to the threshold $th_{minOrgReq}$, which defines the number of minimum acceptable requests received by the organizational services as a whole.

Factor	
Name	OrgRequests
Description	The number of requests received by all organizational services is lower than a defined threshold.
Parameters	$th_{minOrgReq}$, t_1, t_2, org_i
Condition	$(\sum_{\forall s_i \in org_i.Services} Monitor.Requests(s_i, t_1, t_2)) < th_{minOrgReq}$
Monitor	Monitor element

Table 4.14: Description of the *OrgRequests* factor

ModifyProducts solution.

The solution to the *Specialist to generalist* force is to modify the products (that are specific) to more generic products that attract a wider range of agents, thus increasing the number of potential clients of the organization, and assuring the organizational survival (see Table 4.15). Then, it is defined a list of new products (*NewProductsList*) that will replace the existing products offered by the organizational services. The product $pr_i \in NewProductsList$ will replace the products offered by the service s_i. The associated cost of this solution can be calculated as the cost for a service to offer the new product ($cost(pr_i)$), minus the cost of offering the old product of the service ($cost(s_i.Products)$), plus the cost of changing from one product to another ($cost(Change(s_i.Products, pr_i))$).

Field	**Description**
Name	ModifyService
Description	The organizational services are modified so as to offer more generic products.
Condition	OrgRequests
Parameters	org_i, $NewProductsList$
Actions	$\forall s_i \in org_i.Services, \forall pr_i \in s_i.Products : s_i.Products.Delete(pr_i)$ $\forall s_i \in org_i.Services, \forall pr_j \in NewProductsList : s_i.Products.Add(pr_j)$
Cost	$\sum_{\forall s_i \in org_i.Services, \forall pr_i \in s_i.Products, \forall pr_j \in NewProductsList} cost(pr_j) - cost(pr_i) + cost(Change(pr_i, pr_j))$
Executor	Organizational manager

Table 4.15: *ModifyProducts* solution

AddService solution.

Another possibility to make the services of the organization generical enough to appeal more clients is to add totally new organizational services offering generic products. These services have to be similar to the existing ones in their functionalities, but offering more generic products. A list of new organizational services (*NewServiceList*, containing the new and more generical services) is defined so as to be added to the existing organizational service list. This solution is detailed in Table 4.16.

Field	**Description**
Name	AddService
Description	New services that offer more generic products are added.
Condition	OrgService
Parameters	org_i, $NewServiceList$
Action	$\forall s_i \in NewServiceList : org_i.Services.Add(s_i)$
Cost	$\sum_{\forall s_i \in NewServiceList} cost(org_i.Services.Add(s_i))$
Executor	Organizational manager

Table 4.16: *AddService* solution

4.3.1.4 *Decay and deterioration* Force

The *decay and deterioration* force (see Table 4.17) is active in situations where environmental issues make the organizational goals to change without any notice to the organization. In a regular situation, all the organizational goals can be achieved by means of making use of the set of services of the organization. However, when a change in the set of organizational goals happens, the set of services of the organization may not be able to help agents to achieve the new organizational goals, so there would be required to introduce modifications in the set of organizational services so as to comply with the new goals. Then, the factor to detect this force is the set of achievable goals (*AchievableGoals*) of the organization (i.e., the sum of the achievable goals of each organizational service). The solution to this force is to modify the services so then they should help agents to fulfil the required organizational goals ($ModifyServices$).

Field	**Description**
Name	DecayDeterioration
Description	Some organizational goals are not able to be achieved since the organizational services are not able to help agents to achieve them.
Type	External
Factors	$AchievableGoals$
Force detection condition	$AchievableGoals.Condition = True$
Solutions	$ModifyServices$
Force solution condition	-

Table 4.17: Definition of the *Decay and deterioration*

AchievableGoals factor.

The set of organizational services is defined with the idea of being able to help agents to achieve all the organizational goals ($org_i.Goals$). The factor to detect the *Decay and deterioration* force is to check the set of goals that the organization is able to achieve with the current specification of organizational services (see Table 4.18). If not all organizational goals are able to be achieved with the current organizational services ($org_i.Services$), then it is stated that the *Decay and deterioration* force is active, and a reorganization is required.

Factor	
Name	AchievableGoals
Description	The goals that are able to be achieved by an organization are not totally included in the set of organizational goals.
Parameters	org_i
Condition	$org_i.Goals \not\subseteq \bigcup_{\forall s_i \in org_i.Services}\{s_i.Goals\}$
Monitor	Monitor element

Table 4.18: Description of the *AchievableGoals* factor

ModifyServices solution.

The solution for this force (see Table 4.19) is to modify the organizational services so that they help agents to fulfil a set of goals that are part of the organizational goals. Therefore, in this proposed solution, the service s_i is first deleted from the organization ($org_i.Services.Delete(s_i)$), and then the modified service s_i' is added to the organizational definition ($org_i.Services.Add(s_i')$).

Field	Description
Name	ModifyServices
Description	A service is modified in a way that it would help agents to fulfil a different set of goals.
Condition	$AchievableGoals \vee OrgDensity$
Parameters	s_i, s'_i, org_i $\nexists g_i \in s_i.Goals : g_i \in org_i.Goals \wedge \exists g_j \in s'_i.Goals : g_j \in org_i.Goals$
Action	$org_i.Services.Delete(s_i); org_i.Services.Add(s'_i)$
Cost	$cost(org_i.Services.Delete(s_i)) + cost(org_i.Services.Add(s'_i))$
Executor	Organizational manager

Table 4.19: *ModifyServices* solution

4.3.1.5 *Technological changes* Force

New advances in technologies may appear during the life-cycle of an organization. These advances could be useful for the organization to improve its performance. Therefore, it is interesting for an organization to adopt this new technology if the budget allows doing it (Table 4.20).

In an OCMAS domain, the new technology refers to the services provided by the organization. New technology may bring new services, or modify the existing ones, to the organization. The factor of the utility of a service ($UtilityNewService$) is used to detect the force, and the solution $ModifyService$ modifies the existing services so as to introduce the new technology.

Field	Description
Name	TechnologicalChanges
Description	New technology is available to increase the utility of the services.
Type	External
Factors	$UtilityNewService$
Force detection condition	$UtilityNewService.Condition = True$
Solutions	$ModifyService$
Force solution condition	$argmax_{x \in \{ModifyService, noSol\}} utility_x$

Table 4.20: Definition of the *TechnologicalChanges* force

UtilityNewService factor.

During the life of the organization, new equipment will be available in the organizational environment that the organization can adopt and use. The main reason to trigger an adaptation here is the appearance of new equipment that would be adopted as organizational resources. However, the cost of making this change has also to be taken into account so as to decide whether a change is convenient or not. Therefore, as shown in Table 4.21, the utility of the service with the current specification ($utility(s_i, t_1, t_2)$) has to be compared to the utility that would be obtained with the new service specification ($utility(s'_i, t_1, t_2)$) and the cost of changing from one specification to another ($cost(org_i.Services.Delete(s_i)) + cost(org_i.Services.Add(s'_i))$).

The *utility of a service* s_i at a given time interval $[t_1, t_2]$ is calculated as:

$$\begin{gathered} utility(s_i, t_1, t_2) = profit(s_i.Products, t_1, t_2) - \\ (cost(s_i.Resources, t_1, t_2) + cost(s_i.Agents, t_1, t_2)) \end{gathered} \tag{4.4}$$

where $profit(s_i.Products, t_1, t_2)$ is the profit that the products of the service s_i generate during a given time interval $[t_1, t_2]$, $cost(s_i.Resources, t_1, t_2)$ is the cost of using the resources of the service s_i during a given time interval $[t_1, t_2]$; and $cost(s_i.Agents, t_1, t_2)$ is the cost of the agents (in terms of computational cost) required to provide the service s_i during an interval $[t_1, t_2]$.

In VOF (see Chapter 3), the cost of consuming a resource or the profits of generating a product where defined in the description of a service (see Definition 7) (*CostRes* and *BenPro*, respectively). The function that returns the cost of consuming all the resources of a service s_i is $costServ(s_i)$ (see Chapter 3, Equation 3.17), while the function that returns the profit obtained by all products of a service s_i is $benServ(s_i)$ (see Chapter 3, Equation 3.18). Equivalencies between VOF and the description of this force are defined as:

- $benServ(s_i) \equiv profit(s_i.Products, t_1, t_2)$

- $costServ(s_i) \equiv (cost(s_i.Resources, t_1, t_2) + cost(s_i.Agents, t_1, t_2))$

Factor	
Name	UtilityNewService
Description	The utility of the current service specification is compared to the utility of the new service specification and the cost of switching services.
Parameters	s_i, s'_i,t_1, t_2
Condition	$utility(s_i, t_1, t_2) < (utility(s'_i, t_1, t_2) - (cost(org_i.Services.Delete(s_i)) + cost(org_i.Services.Add(s'_i))))$
Monitor	Monitor element

Table 4.21: Description of the *UtilityNewService* factor

ModifyService solution.

After determining that an adaptation is required to introduce the new technology to the organization it is then necessary to modify the service specification to deploy this change (Table 4.22). In the same way as it was carried out at the *Decay and deterioration* force, to introduce modifications into a service, this service has first to be deleted from the organization ($org_i.Services.Delete(s_i)$) and then added again with the new technology ($org_i.Services.Add(s'_i)$).

Field	**Description**
Name	ChangeEquipment
Description	The new equipment is added to a service by modifying its specification.
Condition	$UtilityNewService$
Parameters	org_i, s_i, s'_i $s_i.Resources \neq s'_i.Resources$
Action	$org_i.Services.Delete(s_i); org_i.Services.Add(s'_i)$
Cost	$cost(org_i.Services.Delete(s_i)) + cost(org_i.Services.Add(s'_i))$
Executor	Organizational manager

Table 4.22: *ModifyService* solution

4.3.1.6 *Competition* Force

An organization might not be the only one populating the organizational environment. There can be other organizations working in the same environment that may offer similar products or services that start a competition situation, thus provoking a decrease on the number of clients of the organization (so less requests for services would be received). This *Competition* force (see Table 4.23) is detected by calculating the organizational density (*OrgDensity*). This is, it has to be calculated the number of organizations working in the same environment offering similar products and services. To solve this situation, it is possible to associate the organization with another (*AssociateOrg*), or to modify the organizational services ($ModifyServices$) to make them different to other organizations so as to appeal more clients.

Field	**Description**
Name	Competition
Description	Organizations populating the environment offer similar products and services representing competence for our organization.
Type	External
Factors	$OrgDensity$
Force detection condition	$OrgDensity.Condition = True$
Solutions	$AssociateOrg$, $ModifyServices$
Force solution condition	$argmax_{x \in \{AssociateOrg, ModifyServices\}} utility_x$

Table 4.23: Definition of the *Competition* force

OrgDensity factor.

The *Competition* force is active when this factor appears in the organization (see Table 4.24). If the number of organizations populating the same environment of the organization is higher than an expected value ($th_{allowedComp}$ threshold), and the number of clients is decreasing, then this force is active. Therefore, the number of organizations has to be checked, as well as the number of clients of the organization at two different moments in time.

This factor is active if the equation $competition(org_i, t_1, t_2)$ holds:

$$\begin{aligned} competition(org_i, t_1, t_2) = (| \bigcup_{o_i \in org_i.Environment} \{o_i\} : \exists s_i \in o_i.Services \\ \wedge \exists s_j \in org_i.Services \wedge s_i.Products = s_j.Products| > th_{allowedComp}) \wedge \\ org_i.Clients(t_1) > org_i.Clients(t_2) \wedge t_2 < t_1 \end{aligned} \tag{4.5}$$

where $org_i.Clients(t_1)$ represents the number of clients of the organization org_i at a time t_1, and $org_i.Clients(t_2)$ represents the number of clients of the organization org_i at a time t_2. These expressions help on the process of checking if the number of clients has been reduced.

Factor	
Name	OrgDensity
Description	Other organizations in the organizational environment are offering similar products and services, thus making the number of clients of our organization to decrease.
Parameters	org_i, t_1, t_2, $th_{allowedComp}$
Condition	$competition(org_i, t_1, t_2) = True$
Monitor	Monitor element

Table 4.24: Description of the *OrgDensity* factor

AssociateOrg solution.

One of the possible solutions for this force is to associate the organization org_i with another organization, so they could join forces and represent a more powerful organization against their competitors (Table 4.25). This association does not imply a merge between the two organizations, but a collaboration between them in other ways, like using the products of one organization as resources for another to offer new products. This is achieved by adding (or modifying) two services (one on each organization), in a way that one of them produces new products that the other service uses them as resources. The new products generated by this collaboration will put the organization into a new privilege position.

The situation where two organizations (org_1 and org_2) can be associated for offering new products is represented as:

$$\begin{gathered}\exists s_i \in org_1.Services, \exists s_j \in org_2.Services : \\ (s_i.Resources \subseteq s_j.Products) \vee (s_j.Resources \subseteq s_i.Products)\end{gathered} \tag{4.6}$$

Associating two organizations means that one organization is able to offer not only its own organizational services, but also the ones from another organization. Equations 4.7 and 4.8 represent this situation.

$$org'_1.Services = org_1.Services \bigcup org_2.Services \tag{4.7}$$

$$org'_2.Services = org_1.Services \bigcup org_2.Services \tag{4.8}$$

Field	**Description**
Name	AssociateOrg
Description	The organization associates with another organization to improve its performance.
Condition	$OrgDensity$
Parameters	org_1, org_2
Action	$\forall s_i \in org_1.Services : org_2.Services.Add(s_i)$ $\forall s_j \in org_2.Services : org_1.Services.Add(s_j)$
Cost	$\sum_{\forall s_j \in org_2.Services} cost(org_1.Services.Add(s_j))$
Executor	Organizational manager

Table 4.25: *AssociateOrg* solution

ModifyServices solution.

The second possible solution to the *Competition* force consists on modifying the organizational services (by changing the products they generate) to make them different from other organizations, and trying to attract more clients. This solution is the same as the one previously presented in Table 4.19. As stated, a factor or a solution can be a factor or solution of more than one force.

4.3.1.7 *Demographical features* Force

External agents may behave in unexpected ways (e.g., violating norms). Therefore, if an entity is acting in an unappropriated manner for the organization, it has to be sanctioned by the organization. Table 4.26 presents a template for the force, including the factor to detect it ($ViolatedNorms$) and its solution ($SanctionAgent$).

In an open system, in which external agents coming from the environment of the organization may join the organization, these agents may play a 'Client' role (or similar), which has associated norms that they have to follow.

The solution to this force may be considered more a regulation process rather than an adaptation one. Since there is no modification of any of the structural elements of the organization, the solution for the force cannot be considered as adaptation or organizational change, but a regulation. However, this force is here described because it is considered as a force that drives organizational change by the researchers of social sciences, and it is a situation that can normally appear in an agent organization.

Field	**Description**
Name	DemographicalFeatures
Description	Due to the openness of the organization, some agents could not comply with the organizational rules and regulations.
Type	External
Factors	$ViolatedNorms$
Force detection condition	$ViolatedNorms.Condition = True$
Solutions	$SanctionAgent$
Force solution condition	-

Table 4.26: Definition of the *Demographical features* force

ViolatedNorms factor.

Detecting this force implies to take attention to external agents (see Table 4.27) e.g., playing the *Client* role. External agents join the organization to consume the services offered by the organization. Client agents have to follow the norms of the organization they are populating. The agents are monitored by a Monitor that tracks the violations of the norms of the system made by agents.

Factor	
Name	ViolatedNorms
Description	A self-interested agent violates a norm of the system.
Parameters	a_i, n_i, org_i
Condition	$\exists n_i \in org_i.Norms, \exists a_i \in org_i.Agents : Monitor.Violations(a_i, n_i)$
Monitor	Monitor element

Table 4.27: Description of the *ViolatedNorms* factor

SanctionAgent solution.

In the case that it is detected that an external agent a_i is not following the norms associated to the role it is playing, it is necessary to take an action to avoid that this agent may affect the normal performance of the organization. Thus, the agent is sanctioned by the organization to avoid further problems by using the action $org_i.Agents.Sanction(a_i)$ (see Table 4.28).

Field	**Description**
Name	SanctionAgent
Description	The agent that is not following its assigned norms is sanctioned to avoid damage to the organization.
Condition	$ViolatedNorms$
Parameters	org_i, a_i
Action	$org_i.Agents.Sanction(a_i)$
Cost	$cost(org_i.Agents.Sanction(a_i))$
Executor	Organizational manager

Table 4.28: *SanctionAgent* solution

4.3.1.8 *Laws and regulations* Force

An organization has its own norms governing its behavior. However, the organization is sometimes nested inside another superior organization with its own norms and rules (i.e., the local, regional, or national governments, etc.) that have to be respected by the organization. Therefore, if there are conflicting norms between the organization org_i and a superior organization, the organization org_i has to adapt its norms so as to comply with the norms of that superior organization (see Table 4.29).

Field	**Description**
Name	LawsRegulations
Description	Norms coming from a superior organization of the organization are in conflict with its own norms.
Type	External
Factors	$ConflictingNorms$
Force detection condition	$ConflictingNorms.Condition = True$
Solutions	$ModifyNorms$
Force solution condition	-

Table 4.29: Definition of the *Laws and regulations* force

ConflictingNorms factor.

The *Laws and regulations* external force is triggered when at least there is a norm n_2 from a superior organization org_2 that is in conflict with another norm n_1 of the organization org_1 itself (Table 4.30). It must be decided when a norm n_2 from the superior organization org_2 is in conflict with another norm n_1 of the organization org_1 itself. In our approach, two norms are in conflict when both are referring to the same action and the same roles, but they have a different deontic operator. For example, when there is a norm n_1 belonging to the organization org_1 with an 'obligation' deontic operator, and a norm n_2 from the superior organization org_2, but with a 'prohibition' operator and referring to the same action and roles.

The conflict between two norms is defined by means of the following function:

$$conflict(n_1, n_2) = \begin{cases} true \ : \ n_1 \in org_i.Norms \wedge n_1 = \langle D, C, \mathfrak{T} \rangle \wedge \\ n_2 \in org_2.Norms \wedge org_1 \in org_2.OUs \wedge n_2 = \langle D', C, \mathfrak{T} \rangle \wedge \\ D \neq D' \\ false \ : \ otherwise \end{cases} \tag{4.9}$$

A detailed description of the norm concept can be found in Chapter 3, Definition 11.

Factor	
Name	ConflictingNorms
Description	There are norms coming from the environment that are in conflict with the organizational goals.
Parameters	n_1, n_2, org_1, org_2
Condition	$\exists n_1 \in org_1.Norms, n_2 \in org_2.Norms, org_1 \in org_2.OUs : conflict(n_1, n_2) = True$
Monitor	Monitor element

Table 4.30: Description of the *ConflictingNorms* factor

ModifyNorms solution.

The solution for this force is to modify the organizational norms of org_i that are in conflict with the norms of the superior organization (see Table 4.31). Therefore, a conflicted norm n_1 has to be modified in a way that it is not in conflict anymore with the norm n_2 from the superior organization. Thus, the norm may be deleted from the organization, and it could also be added after carrying out modifications of the norm. This kind of changes might focus on the actions to take on each norm (C) or on the roles affected by this norm ($\mathcal{T}$).

Field	**Description**
Name	ModifyNorms
Description	The norms of the organization are modified to comply with the environmental goals.
Condition	$ConflictingNorms$
Parameters	org_i, $n_1 = \langle D, C, \mathcal{T} \rangle$, $n'_1 = \langle D, C', \mathcal{T}' \rangle$
Action	$org_i.Norms.Delete(n_1)$; $[org_i.Norms.Add(n'_1)]$
Cost	$cost(org_i.Norms.Delete(n_1)) + cost(org_i.Norms.Add(n'_1))$
Executor	Organizational manager

Table 4.31: *ModifyNorms* solution

4.3.1.9 *Integration and externalization processes* Force

If the utilities of two organizations are not as higher as expected, they may take the decision of merging or associating with another organization to join forces and reduce costs, thus increasing their utility. The reason for the change is that the utility of a standalone organization is low, and it is checked whether the joint utility with another organization is higher and beneficial for both organizations. There are two factors to detect this force ($OrgUtilities$ and $OrgDensity$) and two solutions to apply ($MergeOrgs$ and $AssociateOrg$, being the one which maximizes the utility the chosen solution), as depicted in Table 4.32.

Field	**Description**
Name	Merging
Description	Two organizations join to improve their situation inside their environment.
Type	Internal
Factors	$OrgUtilities$, $OrgDensity$
Force detection condition	$OrgUtilities.Condition = True \vee OrgDensity.Condition = True$
Solutions	$MergeOrgs$, $AssociateOrg$
Force solution condition	$argmax_{x \in \{MergeOrgs, AssociateOrg\}} utility_x$

Table 4.32: Definition of the *Integration and externalization processes* force

OrgUtilities factor.

This factor calculates the value of the utility of the two organizations (see Table 4.33). The condition to trigger this force is to check that the standalone utilities of the two organizations are lower than the utility of the joint organization. The utility of an organization is calculated as the average of the utilities of all the services of this organization. So the utility of an organization org_i during a time interval $[t_1, t_2]$ is calculated as:

$$utility(org_i, t_1, t_2) = \frac{\sum_{\forall s_i \in org_i.Services} utility(s_i, t_1, t_2)}{|org_i.Services|} \tag{4.10}$$

where $utility(org_i, t_1, t_2)$ is the utility of an organization org_i, $utility(s_i, t_1, t_2)$ is the utility of the service s_i (see Equation 4.4), and $org_i.Services$ is the set of services of the organization org_i, whose relation with VOF elements can be checked in Table 4.4.

Let us suppose that $org_j \in \mathcal{VO}$ and $org_k \in \mathcal{VO}$ are two independent organizations, and org_i is the organization that would result from joining org_j and org_k. Then, this factor will be active if this expression holds:

$$(utility(org_j, t_1, t_2) < utility(org_i, t_1, t_2)) \wedge (utility(org_k, t_1, t_2) < utility(org_i, t_1, t_2)) \tag{4.11}$$

In order to calculate the utility of the organization org_i that appears when joining the organizations org_j and org_k, it is necessary to know the set of services of this new organization. This set may be calculated depending on a specific scenario. However, in a generic situation, when two organizations join, they join their sets of services. Therefore, the set of services of the organization org_i is defined as:

$$org_i.Services = org_j.Services \cup org_k.Services \tag{4.12}$$

Notice that in this case, org_i is calculated with a predictive intentions. The organization may be then merged or just associated to another depending on the chosen solution.

Factor	
Name	OrgUtilities
Description	The utility of the organizations by their own is lower than the utility in the case they were joined.
Parameters	org_j, org_k, t_1, t_2 org_i where $org_i.MergeOrgs(org_j, org_k)$
Condition	$(utility(org_j, t_1, t_2) < utility(org_i, t_1, t_2)) \wedge (utility(org_k, t_1, t_2) < utility(org_i, t_1, t_2))$
Monitor	Monitor element

Table 4.33: Description of the *OrgUtilities* factor

OrgDensity factor.

It could be also possible that two organizations merge in a situation where the number of organizations of the same environment that offer similar services is high. This factor takes into account the organizational density, which represents the number of organizations that populate the same environment providing similar products or services. This factor also appeared at the *Competition* force (see Table 4.24).

MergeUnits solution.

The first solution for this force is to merge the two organizations whose utility is not as higher as expected (see Table 4.34). In order to carry the change out, it has to be analyzed the structure of the organization, and then it has to be decided how the structure is mixed, which roles are redundant, etc. The function $org_i.MergeOrgs(org_j, org_k)$ defines that an organization org_i is created when the organizations org_j and org_k merge.

In a general situation, in order to give birth to a new organization, this supposes to merge the organizational elements (services, roles, agents, norms, goals, etc.). However, some of these elements might be redundant, so only one of them is kept in the new organization. Additionally, it may happen that elements from both organizations are not compatible (e.g., contradictory norms or goals) so the organizational management would be in charge of deciding about the elements that will prevail in the new, merged organization. We define the function $\mathit{compatible}(e_1, e_2)$ that returns $True$ if the elements e_1 and e_2 (may be goals, norms, services, or another organizational element) are compatible between them.

Therefore, the function $org_i.MergeOrgs(org_j, org_k)$ is supposed to execute the following operations in a generic situation:

- $org_i.Services = org_j.Services \cup org_k.Services$
- $org_i.Goals = org_j.Goals \cup org_k.Goals$, where $\forall g_m, g_n \in org_i.Goals$: $\mathit{compatible}(g_m, g_n) = True$
- $org_i.Roles = org_j.Roles \cup org_k.Roles$, where $\forall r_m, r_n \in org_i.Roles$: $\mathit{compatible}(r_m, r_n) = True$
- $org_i.Norms = org_j.Norms \cup org_k.Norms$, where $\forall n_m, n_n \in org_i.Norms$: $\mathit{compatible}(n_m, n_n) = True$
- $org_i.Environment = org_j.Environment \cup org_k.Environment$
- $OE(org_i, t) = OE(org_j, t{-}1) \cup OE(org_k, t{-}1)$, where $\forall e_m, e_n \in OE(org_i, t)$: $\mathit{compatible}(e_m, e_n) = True$

As explained, these operations are defined only for a generic situation. On each specific domain, the organizational management of both organizations has the responsibility of carrying out the type of merge that is more convenient for the new organization.

Then, the merge is deployed in the two organizations, resulting in a new organization that combines elements from the two of them. The main difference between merging and associating two organizations is that merging implies transforming two organizations into a different organization, while associating two organizations implies to combine and share between them the same set of services, but keeping their existence as standalone and differentiated organizations.

Field	**Description**
Name	MergeUnits
Description	The two organizations merge to improve their utility.
Condition	$OrgUtilities \lor OrgDensity$
Parameters	org_i, org_j, org_k
Action	$org_i.MergeOrgs(org_j, org_k)$
Cost	$cost(org_i.MergeOrgs(org_j, org_k))$
Executor	Organizational manager

Table 4.34: *MergeUnits* solution

AssociateOrg solution.

One of the possible solutions for this force is to associate the organization with another organization, so they could join forces and represent a more powerful organization against their competitors. This association does not imply a merge between the two organizations, but a collaboration between them in other ways, like using the products of one organization as resources for another to offer new products. This is achieved by adding (or modifying) two services (one on each organization), in a way that one of them produces new products that the other service uses them as resources. The new products generated by this collaboration will put the organization into a new privilege position. This solution also appeared in the Competition force (see Table 4.25).

4.3.2 Internal forces

In this section, the internal forces are described using the templates, as the external forces have been described. In these forces, the organizational theory states that the pressure for change comes from inside the organization (Ald99) (from its own services, norms, goals, members, etc.). The internal forces are: (i) growth; (ii) economical restrictions; (iii) crisis; and (iv) goal succession.

4.3.2.1 *Growth* Force

In the regular organizational life-cycle, an organization starts containing a small number of members. Thus, a simple, basic structure is normally enough to correctly manage the organization. This simple structure is normally built of one organizational unit and a manager role which is in charge of the subordinate agents. However, if the organization is successful enough, the number of agents inside the organization will increase (because it appeals a higher number of external agents e.g., clients), and also the budget may rise.

This growth inside the organization will force an organizational change that will consist on transforming the organization into a hierarchy (Ald99) by introducing new organizational units and intermediate manager roles that will improve the way the organization is managed, distributing this management around the new organizational units, thus balancing the load of the different entities of the system. A hierarchy is chosen instead of a flat structure because a flat structure will make that the newly added organizational units would not be under control of the original organizational unit.

Table 4.35 defines the *growth* force with the two factors, the number of agents ($NumberAgents$), and the budget the organization has available ($HighBudget$); and one solution ($MakeHierarchy$). The force is triggered when either the number of agents inside an organizational unit, or the organizational budget is higher than the specified threshold.

Guideline for detecting a force	
Field	**Description**
Name	Growth
Description	The number of clients or budget of the organization increase
Type	Internal
Factors	$NumberAgents$, $HighBudget$
Force detection condition	$NumberAgents.Condition = True \vee HighBudget.Condition = True$
Solutions	$MakeHierarchy$
Force solution condition	-

Table 4.35: Definition of the *Growth* force

NumberAgents factor.

The *Growth* internal force can be detected by taking into account the number of agents that populate the organization (Table 4.36). The number of agents populating an organization is calculated by using the function $org_i.Population$, that returns the number of agents in the organization org_i, and it is equivalent to the function $|agentsVO(org_i, t)|$ from the Organizational Entity of VOF (see Chapter 3, Equation 3.49; and Table 4.4). This number has to be compared to the maximum number of agents allowed for the organization org_i, which is the threshold $th_{MaxPopulation}$ defined by the organizational designer. If the population is bigger than the maximum capacity, then the factor is active.

Name	NumberAgents
Description	The number of agents populating an organization is higher than the maximum allowed population
Parameters	org_i, $th_{MaxPopulation}$
Condition	$org_i.Population > th_{MaxPopulation}$
Monitor	Monitor element

Table 4.36: *NumberAgents* factor.

HighBudget factor.

When the budget of the organization is higher than expected it is necessary to make a change in the organization (Table 4.37), because a rise on the budget will enable the organization to increase its number of services and functionalities, thus attracting more clients. Therefore, a change in the organization would be required to properly manage the new situation of the organization. The budget of an organization org_i is represented by means of $org_i.Budget$, and the maximum budget it may have without being required a modification is represented by the threshold $th_{MaxBudget}$.

Name	HighBudget
Description	If the budget of the organization increases above a threshold, then a change is necessary.
Parameters	org_i, $th_{MaxBudget}$
Condition	$org_i.Budget > th_{MaxBudget}$
Monitor	Monitor element

Table 4.37: *HighBudget* factor.

MakeHierarchy solution.

The *Growth* force is solved by transforming the organization into a hierarchy (see Table 4.38). In the business field, a hierarchy allows to distribute employees, clients, services, etc. among different organizational units, thus facilitating the management of the organization by means of the newly created organizational units.

In the case of the OCMAS domain, it is necessary to add a number $NewOUs$ of organizational units (OUs) (this number depends on each domain), where an organizational unit ou_i is added to an organization org_i as $org_i.OUs.Add(ou_i)$ as well as an agent a_i for each new OU to manage it ($ou_i.Agents.Add(a_i)$). These agents have to play a role $r_i \in org_i.Roles$ that allows them to manage an OU ($a_i.Roles.Get(r_i)$).

In this specific organizational definition, this solution would be equivalent to split the organization. In order to achieve that, it is required that the original organizational unit remains in the organization, but not providing any specific functionality. It would be a container for organizational goals and roles that the newly created organizational units will inherit. Thus, the actual behavior of the organization would be the behavior of two independent organizations that will allow to achieve the same global goal.

Solution for preventing damage or taking advantage of a force	
Field	**Description**
Name	MakeHierarchy
Description	Make a hierarchy in the organization, add organizational units and agents to improve the organizational management.
Factor	$NumberAgents \vee HighBudget$
Parameters	org_i, $NewOUs$, r_i
Actions	$\forall ou_i \in NewOUs : org_i.OUs.Add(ou_i); ou_i.Agents.Add(a_i); a_i.Roles.Get(r_i)$
Cost	$NewOUs * (cost(org_i.OUs.Add(ou_i)) + cost(ou_i.Agents.Add(a_i)) + cost(a_i.Roles.Get(r_i)))$
Executor	Organizational Manager

Table 4.38: *MakeHierarchy* solution

4.3.2.2 *Economical restrictions* Force

Agent organizations have a finite budget to develop their activities. They are limited to specific computational resources. Therefore, an organization is required to change when it is surpassing the limit of the budget it has been assigned (see Table 4.39). The factor to detect the force is the cost associated to the organization ($CostOrg$), while the solution is to reduce the cost generated by the organization ($ModifyServices$).

Field	**Description**
Name	EconomicalRestrictions
Description	The cost of the organization is higher than its associated budget.
Type	Internal
Factors	$CostOrg$
Force detection condition	$CostOrg.Condition = True$
Solutions	$ModifyServices$
Force solution condition	-

Table 4.39: Definition of the *EconomicalRestrictions*

CostOrg factor.

In the case that the cost generated by an organization org_i is higher than its budget $org_i.Budget$ (see Table 4.4 to check its relation with VOF), it is stated that the force is active and a solution has to be carried out (see Table 4.40). The cost of an organization is calculated as the cost generated by its organizational services and the cost generated by the roles being played by entities:

$$cost(org_i) = \sum_{\forall s_i \in org_i.Services} s_i.Cost + \sum_{\forall r_i \in org_i.Roles} org_i.Cost(r_i) \qquad (4.13)$$

where $s_i.Cost$ refers to the cost generated by a service s_i (see Table 4.4), and $org_i.Cost(r_i)$ (see Table 4.5) returns the cost generated by the entities playing the role r_i in an organization org_i.

Factor	
Name	CostOrg
Description	The cost of the organization is higher than the budget it has assigned.
Parameters	org_i
Condition	$cost(org_i) > org_i.Budget$
Monitor	Monitor element

Table 4.40: Description of the *CostOrg* factor

ReduceCost solution.

Reducing the cost generated by the organization can be done by reducing the cost generated by its organizational services (see Table 4.41). The cost of a service is mainly generated by the cost of consuming the resources that the service requires for its execution. Therefore, a service s_i would have to be modified in a way that the resources it consumes are less costly than the previously consumed ones, i.e., $cost(s_i.Resources) > cost(s'_i.Resources)$, being s'_i the modified version of s_i. It is possible to only delete the service (i.e., without adding it again after modifying it) if it is considered that a modified service would not give the organization significant profits.

The cost of the resources consumed by a service s_i is calculated as:

$$cost(s_i.Resources) = \sum_{\forall res_i \in s_i.Resources} cost(res_i) \tag{4.14}$$

Field	Description
Name	ReduceCost
Description	The cost generated by the organization is reduced by modifying its services.
Condition	$CostOrg$
Parameters	org_i, s_i, s_i' where $cost(s_i.Resources) > cost(s_i'.Resources)$
Action	$\forall s_i \in org_i.Services : org_i.Services.delete(s_i); [org_i.Services.add(s_i')]$
Cost	$\sum_{\forall s_i \in org_i.Services} cost(org_i.Services.delete(s_i)) + cost(org_i.Services.add(s_i'))$
Executor	Organizational manager

Table 4.41: *ReduceCost* solution

4.3.2.3 *Goal succession* Force

There are organizations that disappear after fulfilling their organizational goals. However, other organizations look for new organizational goals to achieve. Therefore, these organizations will continue with their existence, after changing their organizational goals with others that are similar to the former ones (or at least, achievable with the current organizational specification), trying to take profit of the capabilities and abilities of the organizational members. Then, this force (see Table 4.42) can be detected by means of one factor, the one stating if all the organizational goals have been achieved or not (*AchievedGoals*), and one solution, modifying the pursued organizational goals (*ChangeGoals*).

For example, think about an organization fighting for the right to vote for women during the 19th and 20th centuries. After achieving this goal, this organization could disappear, or it might change its objectives so as to look for other goals such as the equality of opportunities and salary for women.

Field	**Description**
Name	GoalSuccession
Description	The organization achieves all its goals and loses its meaning.
Type	Internal
Factors	$AchievedGoals$
Force detection condition	$AchievedGoals.Condition = True$
Solutions	$ChangeGoals$
Force solution condition	-

Table 4.42: Definition of the *GoalSuccession* force

AchievedGoals factor.

This force is acting over the organization if all the current organizational goals are achieved (Table 4.43). It is necessary to know whether an organizational goal is being achieved or not at a given time, so we use the set $achievedGoals(org_i, t)$ (defined in Chapter 3, Equation 3.22) that defines the organizational goals that the organization org_i has achieved at a time t.

Therefore, the *Goal succession* force is acting inside the organization $org_i \in \mathcal{VO}$ if all its goals are being achieved at a time t:

$$org_i.Goals \subseteq achievedGoals(org_i, t) \tag{4.15}$$

where $org_i.Goals$ is the set of organizational goals that the organization org_i is pursuing.

Factor	
Name	$AchievedGoals$
Description	The goals of an organization have been completely achieved.
Parameters	org_i, t
Condition	$org_i.Goals \subseteq achievedGoals(org_i, t)$
Monitor	Monitor element

Table 4.43: Description of the *AchievedGoals* factor

ChangeGoals solution.

The solution for keeping an organization org_i alive is to look for new organizational goals to be achieved by the organization (see Table 4.44). These goals are chosen by the organizational managers, which are the ones in charge of deciding the future of the organization. The new set of goals is $NewGoals$, where $\forall g_j \in NewGoals : g_j \in \mathcal{G}$. Also, it may be assured that these goals are different from the current, achieved goals ($NewGoals \notin org_i.Goals$).

It may happen that the new goals of the organization are not able to be achieved by the current set of organizational services. Therefore, the *Decay and deterioration* force (Section 4.3.1.4) would be active, so then an action for also solving this force would be taken.

Field	Description
Name	ChangeGoals
Description	New goals are added to the organization.
Condition	$AchievedGoals$
Parameters	org_i, $NewGoals$ $NewGoals \notin org_i.Goals$
Action	$\forall g_i \in org_i.Goals : org_i.Goals.Delete(g_i)$ $\forall g_j \in NewGoals : org_i.Goals.Add(g_j)$
Cost	$\sum_{\forall g_i \in org_i.Goals} cost(org_i.Goals.Delete(g_i))+$ $\sum_{\forall g_j \in NewGoals} cost(org_i.Goals.Add(g_j))$
Executor	Organizational manager

Table 4.44: *ChangeGoals* solution

4.3.2.4 *Crisis* Force

When the utility of the organization highly decreases and it does not lately return to a normal value, thus causing a misbehavior of the organization, it is necessary to deploy changes in the organization to assure that it may not disappear. These changes have to be deep, affecting to all the elements of the organization.

Crisis (Table 4.45) is a force detected by the factor of the low organizational utility (*OrgUtilityCrisis*) that requires a solution (*ChangeOrg*) consisting on deep changes in the organizational structure, functionalities, etc. because if no solution is deployed, the organization would be headed towards its dissolution.

Field	**Description**
Name	Crisis
Description	The organizational utility decreases to a very low value, so deep changes are required.
Type	Internal
Factors	$OrgUtilityCrisis$
Force detection condition	$OrgUtilityCrisis.Condition = True$
Solutions	$ChangeOrg$
Force solution condition	-

Table 4.45: Definition of the *Crisis* force

OrgUtilityCrisis factor.

The organizational utility ($utility(org, t_1, t_2)$), calculated in the *Merging* force (see Equation 4.10), is compared to a special threshold for crisis situations (th_{Crisis}). This threshold has a low value, because if the organization is inside a crisis its organizational utility would be low (normally, lower than 0.5, or even lower). See Table 4.46 for a description of this factor.

Factor	
Name	OrgUtilityCrisis
Description	The utility of the organization decreases below a specified value, so it is stated that the organization is in a crisis.
Parameters	org, th_{Crisis}, t_1, t_2
Condition	$utility(org, t_1, t_2) < th_{Crisis}$
Monitor	Monitor element

Table 4.46: Description of the *OrgUtilityCrisis* factor

ChangeOrg solution.

The solution to a crisis is to carry out deep changes in the organization (see Table 4.47). These changes have to mainly focus on the organizational services, since the organizational utility is mainly calculated as the utility of the services. However, not only the organizational services have to be adapted, but also other organizational elements that may allow the organization to improve its behavior if they are modified or deleted. For example, roles may be deleted from the organization if they suppose a high cost. Also, the organizational goals may be modified so as to adapt them to the new scenario of the organization.

More specifically, a subset of solutions also employed in other forces can be applied here. The specific solutions will depend on the domain, and they will focus on increasing the organizational utility to make it higher than the th_{Crisis} value. These previously presented solutions (one or a combination of them, depending on the domain) are appropriate to be deployed in an organization to solve a crisis situation: *DeleteService* (Table 4.12), *ModifyProducts* (Table 4.15), *ModifyServices* (Table 4.19), *AssociateOrg* (Table 4.25), *MergeUnits* (Table 4.34), *ReduceCost* (Table 4.41), and *ChangeGoals* (Table 4.44). We state that these are not the only ones which are appropriate. The final decision about which solution to apply on each domain is responsibility of the designer of that domain.

Field	**Description**
Name	ChangeOrg
Description	Deep changes in the organization are carried out to assure the organizational survival. These changes mainly focus on the organizational services.
Condition	$ChangeOrg$
Parameters	-
Actions	See the table of the specific applied solution
Cost	Depends on the specific solution
Executor	Organizational manager

Table 4.47: *ChangeOrg* solution

4.3.3 Summary of thresholds

During the definition of the forces, some thresholds (see Table 4.48) have been defined to help in the detection of the forces that drive the organizational change. The value of these thresholds has to be set by the organizational designers at the design time, and then they are checked by the implemented monitoring mechanisms (e.g., agents, artifact, etc.) during the runtime of the organization for detecting whether a factor is active or not.

Thresholds		
Thresholds	**Description**	**Factor**
$th_{allowedComp}$	Maximum number of organizations that can populate the same environment as our organization and offer similar products or services without being harmful for our organization.	OrgDensity
th_{minReq}	Minimum acceptable requests received by an organizational service.	ServiceRequests
$th_{minOrgReq}$	Minimum acceptable requests received by the organizational services as a whole.	OrgRequests
th_{Crisis}	Minimum organizational utility value allowed. It is used to detect whether an organization is in crisis.	OrgUtilityCrisis
$th_{MaxPopulation}$	Maximum number of agents that may populate the organization.	NumberAgents
$th_{MaxBudget}$	Maximum budget that the organization can manage without a reorganization.	HighBudget

Table 4.48: Thresholds

4.4 Example: ACME

In this section we will continue presenting the example started in Chapter 3 about the ACME company. Here, an example describing how a force (in this case, the *Obtaining resources* external force) is first detected, and then solved, is depicted. Since this chapter is intended to present the forces from a theoretical point of view we only include how one force is detected and then a solution for this force is applied. However, Chapter 6 presents a case of study based on a smart virtual building, where some forces are instantiated for that case of study.

4.4.1 Identifying when a Force is Acting

The ‘Obtaining resources’ force for ACME can affect it in the case this company is not able to get silicon to manufacture processors. When ACME wants to manufacture new processors, the service $Manufacture$ is called, which fails if it is not able to get the required silicon. The 'Obtaining resources' force is defined by means of Table 4.49, which instantiates the Table 4.6 that provided a generic definition of the force.

Field	**Description**
Name	ObtainingResources for ACME
Description	The execution of the service to manufacture microprocessor fails because it cannot get silicon.
Type	External
Factors	$FailedServiceCallsRate$
Force Detection Condition	$FailedServiceCallsRate.Condition = True$
Solutions	$ReachAgreement$, $ExtendOrganization$
Force Solution Condition	$argmax_{x \in \{ReachAgreement, ExtendOrganization, noSol\}} utility_x$

Table 4.49: Definition of the *Obtaining resources* force for ACME

The 'Obtaining resources' force is acting over an organization for the service $Manufacture$ (from a period of time between t_1 and t_2) if this equation holds (see Table 4.50):

$$Monitor.Failures(Manufacture, t_1, t_2) > \\ ((1 - Manufacture.QoS) * Monitor.Requests(Manufacture, t_1, t_2))$$

where $Manufacture.QoS$ is the quality of the service $Manufacture$ as stated in its service definition, $Monitor.Requests(Manufacture, t_1, t_2)$ are the requests of the service $Manufacture$ from time t_1 to time t_2 as counted by the $Monitor$, and $Monitor.Failures(Manufacture, t_1, t_2)$ is the number of failures of a service $Manufacture$ during the same period of time, and it is calculated (according to

Equation 4.3.1.1) as:

$$Monitor.Failures(Manufacture, t_1, t_2) = \begin{cases} \frac{|R'_{t_1,t_2,Manufacture}|}{|R_{t_1,t_2,Manufacture}|} : \\ |R_{t_1,t_2,Manufacture}| \neq 0 \\ 0 \ : \ |R_{t_1,t_2,Manufacture}| = 0 \end{cases} \tag{4.16}$$

whew the set $R_{t_1,t_2,Manufacture} = \{r\}$ is the set of requests received by the service $Manufacture$ in the period of time $[t_1, t_2]$, and the set $R'_{t_1,t_2,Manufacture} = \{r \in R_{t_1,t_2,Manufacture} | r.status = fail\}$ records the number of failures on requests to the service $Manufacture$ during the same period of time. In our application, a request is defined as a tuple $r = \{service, time, status\}$.

Let us suppose that the quality of service was established to 0.5, meaning that at least the 50% of the requests to the service have to be successful. The number of calls to the service $Manufacture$ during the studied time lapse was 6, i.e. $Monitor.Requests(Manufacture, t_1, t_2) = 6$, while the function $Monitor.Failures(Manufacture, t_1, t_2)$ returned 4 as the number of failures during the same time lapse. Therefore, as it can be checked in Equation 4.4.1 we can state that this force is acting over ACME, and a solution for this force must be carried out.

$$\begin{aligned} Monitor.Failures(Manufacture, t_1, t_2) > \\ ((1 - Manufacture.QoS) * Monitor.Requests(Manufacture, t_1, t_2)) \equiv \\ 4 > (1 - 0.5) * 6 \end{aligned}$$

Factor	
Name	FailedServiceCallsRate for ACME
Description	If the failed service calls rate (i.e., manufacturing microprocessors) is higher than the allowed failure rate threshold, then this force is considered as acting.
Parameters	t_1,t_2,$Manufacture$
Condition	$Monitor.Failures(Manufacture, t_1, t_2) > ((1 - Manufacture.QoS) * Monitor.Requests(Manufacture, t_1, t_2))$
Monitor	Monitor element

Table 4.50: Description of the FailedServiceCallsRate factor for ACME

4.4.2 Facing the Force

Once the force has been detected to be acting over the organization, it is then necessary to face it, trying to get benefit from it or to avoid damage that the force could produce, by following the guideline detailed in Table 4.51 (which is an instantiation of Table 4.8 for the ACME example). In this case, for the 'Obtaining resources' force, it consists on accepting an external agent into the organization which should be able to bring the desired resource to the organization, because it is the solution that maximizes the utility of the organization.

This solution is the one that maximizes the utility because is the one that minimizes the cost, while the benefits obtained by the organization (i.e., the number of correctly fulfilled requests) would be the same with any of the two available solutions. Let us suppose that the cost of adding an agent is 5 units, adding a role has a cost of 4 units, and the cost of making an agent play the new role is 3 units (thus having a total cost of 12 units), while the cost of adding an organizational unit is set to 15 (plus the cost of adding agents that manage the new OU). So, the solution of reaching an agreement with an external agent is chosen because of its lower cost.

Therefore, let us suppose that an external agent named Claire, which should provide ACME with the required amount of silicon, is accepted to join the organization. In order to do so, it is necessary to 'move' the agent Claire to ACME, by adding her to the list of agents of the organization ($ACME.Agents.Add(Claire)$). As previously stated, the proposed guideline for each force is generical, so extra changes to the organization can be carried out if it is considered as required or beneficial for the organization. In the ACME example, since Claire is playing a new role in the organization (which is named as $Provider$) it is necessary to modify the list of organizational roles ($ACME.Roles.Add(Provider)$). Additionally, a new service *GetSilicon* is defined so as to request Claire more silicon to manufacture processors ($ACME.Services.Add(GetSilicon)$).

Field	**Description**
Name	ReachAgreement for ACME
Description	ACME reaches an agreement with an agent that will join the organization and bring the required resource.
Condition	FailedServiceCallsRate
Parameters	$Claire \in \mathcal{A}, Provider \in \mathcal{R}$
Action	$ACME.Agents.Add(Claire)$; $ACME.Roles.Add(Provider)$; $Claire.Roles.Get(Provider)$
Cost	$cost(ACME.Agents.Add(Claire)) + cost(Claire.Roles.Get(Provider)) + cost(ACME.Roles.Add(Provider))$
Executor	Organizational managers

Table 4.51: *ReachAgreement* solution for ACME

4.5 Conclusions

In this chapter, templates for the design of the forces that drive organizational change, including the factors that help to detect if they are active or not, and the solutions that will take advantage of the forces or to prevent damage to the organization, have been presented. This proposal is the main contribution of this work. These forces, which are classified into external and internal forces depending on where the reason for adaptation comes from, are expressed not only with natural language (for the descriptions) but with a formal language (based on VOF) that helps designers and developers of OCMAS on the identification and solution of these forces in a computational domain. Then, having the possibility of detecting and solving the forces that drive organizational change, this gives the opportunity of developing Adaptive Virtual Organizations featuring not only abstract reasons for change, but forces which have been widely studied first on a human domain, and now in an OCMAS domain. Finally, an ACME-based case of study is presented so as to check how a force could be detected and solved in an OCMAS.

5

Artifacting and Regulating the Environment of a Virtual Organization

Designing the forces that drive organizational change in the computational domain is not a trivial work. It is required to be provided with different tools that help developers to design and implement them. Therefore, this chapter presents a tool that is useful to design and implement such forces, named the artifacts for organizational mechanisms. This proposal combines two previous works: the organizational mechanisms (CBHO09) (previously explained in section 2.1.5), which are introduced into a Multi-Agent System with the aim of influencing the behavior of agents populating it to achieve their goals in a proper way; and the artifacts, which were presented within the Agents & Artifacts conceptual framework (RVO07), whose features such as properties and operations present advantages for implementing organizational concepts such as roles, services, norms, etc. In this chapter, first a formal model that defines how organizational mechanisms can be designed by using artifacts theory is presented.

Additionally, an extension of the Environment Dimension of the Virtual Organization Model (ABFF+11) is detailed. This extension allows this model to regulate the environment by supporting *artifacts for organizational mechanisms*. The three main entities of this framework are agents, artifacts, and workspaces, which have been integrated in this work inside the Virtual Organization Model. Finally, this chapter presents a case of study related to a health-care domain and a discussion on this topic.

5.1 Introduction

As explained in Chapter 2, there are different approaches for developing Multi-Agent Systems (MAS), ranging from closed, agent-centered systems to open, organization-oriented systems (AGV+04). When developing this last kind of systems, it can be noticed that the environment surrounding MAS is mainly considered as heterogeneous, unpredictable, distributed, and dynamic (Omi01). Being such a complex environment, it must include mechanisms and tools that help managing and controlling it.

When defining software systems, designers can use metamodels (VG91), which are a mechanism that allows defining modeling languages in a formal way, establishing the primitives and the syntactic-semantic properties of a model. Organizational Modeling Languages (OML) (ABFF+11) are metamodels used by OCMAS designers to define the elements that the system will contain at runtime. These OMLs extend existing modeling languages to include organizational elements. OMLs enable modeling agent coordination inside open systems and establishing mechanisms that control the organization in a social level, i.e. interactions between agents, organizational goals, norms, etc. (DMWD02). These models include both individual and global perspectives, and few of these proposals, such as the one in (ABFF^{+}11), try to provide models capable of representing organizational change, in order to give response to changes in the environment or in the organizational structure.

Nowadays, the environment of a MAS (PW07) is being modeled as a first class abstraction of the system. Different approaches presented new concepts that help developers to model the environment. One of the most recognized is the **Agents & Artifacts** (A&A) conceptual framework (RVO07), which is based on human cooperative elements.

Some proposals rely on the statement that the environment can be used to modify the behavior of a MAS from both a macro perspective (from the system's point of view) and a micro perspective (from agents' point of view). **Organizational mechanisms** (CBHO09) (previously presented in Chapter 2) are a valid method to provide coordination into organizations. These mechanisms can provide additional information to agents which may persuade them to behave in a certain way; or they can produce changes in the environment that may impose certain behavior to agents. Thus, it is very useful to use these mechanisms in an open system where external agents are located, so then being able to promote coordination.

Seeing that artifacts are located into the MAS environment and they can also improve coordination between agents, one of the objectives of this work is to model organizational mechanisms as artifacts, in order to facilitate system designers its usage and implementation. A generic idea of every mechanism will be given, so that MAS developers will be allowed to create the most effective artifact for their system. We will only define the minimum features, properties and operations, that the artifacts must provide to be considered as artifacts for organizational mechanisms.

The second objective of this chapter is to extend a previously validated OML to design OCMAS by adding first-class entities that allow modeling the regulation of the environment, since there are no OMLs that provide these mechanisms. As explained in Chapter 2, most of the current approaches do not take into account the design of environmental regulation at design time. They usually focus on the design of entities that are part of the system but they do not pay much attention on how those entities should be regulated in order to better achieve the goals of the system. Thus, allowing the design of mechanisms for regulating the environment becomes a non-straightforward task. Many models have specific abstractions to allow certain kind of regulations, such as norms, policies, etc., but they lack of more generic and reusable design-time entities that facilitate the regulation of the environment.

Therefore, the reasons for adding these first-class abstract entities for designing OCMAS are: (i) artifacts provide concepts (e.g., workspaces) that help designers to model the environment; (ii) we intend to enable regulation of the environment by using organizational mechanisms; and (iii) to also provide reusable first-class abstractions to build regulation mechanisms at design time.

We have chosen the existing Virtual Organization Model (VOM) (AJB09) as a modeling language to be extended since it was developed inside our research group, named *Grupo de Tecnología Informática - Inteligencia Artificial*. VOM is also providing support to the Agent-Oriented Software Engineering methodology GORMAS (ABJ11), which provides a set of guidelines and patterns to develop virtual organizations. Therefore, we have such high level of knowledge about VOM, so introducing modifications inside it would be easier than modifying a different OML. VOM will be modified so as to support concepts such as artifacts and organizational mechanisms that help describing and operating with the environment.

Finally, a case of study of an application of this type of artifacts will be given, based on a real problem from the health care domain.

The rest of this chapter is structured as follows: Section 5.2 presents the concept of artifact, as well as different proposals based on this concept. Section 5.3 presents the Artifacts for Organizational Mechanisms. Section 5.4 presents the new environment dimension for the Virtual Organization Model that integrates Artifacts for Organizational Mechanisms. Section 5.5 presents a case of study based on a health-care domain. Section 5.6 compares some of the existing artifacts with our proposal. Finally, Section 5.7 gives our conclusions on this proposal.

5.2 Artifacts

Artifacts (RVO07) are non-proactive, but reactive entities that agents employ to achieve their goals. As artifacts do not have assigned goals, they are associated to the goals of the agent that uses the artifact. To accomplish these goals, artifacts provide a *functionality*, which is partitioned into some *operations* that agents can execute when interacting with them. These operations are part of the *usage interface* of the artifact, which is completed with the *observable properties* that agents can check without invoking any operation in it. Artifacts provide a second group of operations, called *link operations* (accessible through a link interface) that enables composition of artifacts and load distribution, since different artifacts may be located at the same or different workspaces (RVO07), which is the portion of environment that is perceived by an agent, who is able to interact with. Every workspace contains a set of artifacts; and the set of workspaces composing the environment is used to define its topology. Finally, artifacts are enhanced with a *function description* (which acts as a manual) and a set of *operating instructions*, an essential feature when dealing with open systems, since external agents can discover artifacts and evaluate whether they could be useful to reach their goals.

Since artifacts are very malleable components from the environment of a MAS, designers can develop new types of artifacts according to actual system needs. The Agent Oriented Software Engineering community has already developed different types of artifacts (EA10): (i) *basic artifacts*, which comprises artifacts that give information of very general world features (for example, clocks, calendars and timetables); (ii) *coordination artifacts* (ORV$^+$04) which improve the coordination between agents in a MAS; (iii) *reputation artifacts* (HBV10) that manage reputation values of agents in an organization; (iv) *cognitive stigmergy artifacts* (ROV$^+$07) which provide information about an agent or a society of agents that can be useful to other agents or groups; (v) *organizational artifacts* (HBKR10) which are used to manage an organization; and (vi) *argumentation artifacts* (OMO08) which manage arguments between agents. Moreover, it is possible to use the *CArtAgO* framework (RVO06) to implement artifacts. This framework is engineered upon the principles of the Agents & Artifacts (A&A) conceptual framework (RVO07).

5.3 Artifacting the Organizational Mechanisms

This section describes how both types of organizational mechanisms (presented in Section 2.1.5), named informative and regulative, can be modeled as artifacts. Artifacts allow an easy merging of the organizational mechanisms into the environment of a MAS.

Firstly, we formalize an artifact in a generic way as follows:

Definition 16. *An **Artifact** is a tuple $\langle PR, OP, LO, St\rangle$ where:*

- *PR are the observable properties of the artifact that agents can directly check without invoking an operation;*
- *OP is the set of operations that agents can execute when interacting with it;*
- *LO stands for link operations, which can be called by other artifacts. This type of operations enables artifact composition and functionality distribution by linking artifacts. In some cases, these operations may be used to help the initialization of another artifact;*
- *St is the internal state of an artifact, which is not accessible by the agents populating the system.*

The result of this modeling is a set of three types of artifacts. The *informative artifacts* are based on the informative mechanisms; the *incentive artifacts* are based on the incentive mechanisms; while the *coercive artifacts* are based on the coercive mechanisms.

5.3.1 Informative artifacts

Informative mechanisms return information about actions to an agent, given a partial description of its internal state and taking into account the partial view of the environment that the mechanism has. The informative mechanism has been modeled as an artifact, named *informative artifact*, being a passive entity that it is used by agents in order to help them in their deliberative process.

Definition 17. *An* ***Informative Artifact*** *is defined as an artifact $Ar_{inf} = \langle PR, OP, LO, St \rangle$ where:*

- *$PR \subseteq \{St \cup \emptyset\}$ are the observable properties of the informative artifact, which are a subset of the information contained into the artifact, or an empty set (in this case it has no observable properties).*
- *$OP : \mathcal{S}' \to \mathcal{I}$ are the operations of the artifact, where:*
  - *$\mathcal{S}'$ represents a partial description of an agent's internal state.*

  - *$\mathcal{I}$ represents the information returned by the artifact, based on the internal state of the artifact and the partial description of the agent's internal state (semantically, $\mathcal{S}' \times St \to \mathcal{I}$).*

- *$LO : \Theta \to \mathcal{I}$ is a link operation that is used by an artifact Ar_1 to obtain information from the Ar_{inf} artifact, where:*
  - *$\Theta \subseteq (\Sigma \cup \mathcal{S}')$ is the information sent by Ar_1 to Ar_{inf};*
  - *$\Sigma \subseteq \{St_1 \cup \emptyset\}$ is a partial state of Ar_1, being St_1 the internal information of Ar_1;*
  - *$\mathcal{S}'$ represents a partial description of the internal state of the agent that is requesting information to the artifact Ar_1;*
  - *$\mathcal{I}$ represents the information returned by the artifact Ar_{inf} to the artifact Ar_1 (previously requested), based on the partial description of Ar_1 (Σ), the partial description of the agent's internal state who is requesting Ar_1 ($\mathcal{S}'$) and the internal state of the artifact Ar_{inf} (St). Semantically: $(\Sigma \cup \mathcal{S}') \times St \to \mathcal{I}$.*

- *St represents the internal state of the artifact, i.e., the information contained into the artifact, which is not directly accessible by agents or other artifacts.*

Informative artifacts are not required to provide link operations, so they might be only accessible by agents in their same workspace. When they offer a link operation, artifacts located in their same workspace or in any other nested or intersected workspace can obtain relevant information from this informative artifact by means of its link operations.

Figure 5.1 shows a graphic representation of an informative artifact. As explained before, this type of artifact needs, at least, one operation: `requestInformation`(St). Giving a partial description of the internal state of the agent, which can contain the roles, believes, facts and other features associated with the agent and its environment, this operation returns a package of information that contains: (i) the type of the information, which can be a recommendation or an advice about

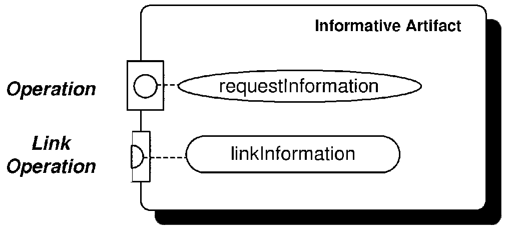

Figure 5.1: *Informative artifact for an informative mechanism*

actions, or information about the consequences of executing an action; (ii) the description of this information, and (iii) a set of actions that are related to this information (they can be services that an agent could take, recommended services, services that have consequences to the agent, etc.). Moreover, as explained before, an informative artifact could need to make requests to other artifacts in order to obtain information or update its internal state. For this reason, the informative artifacts are enhanced with a link operation, `linkInformation`(S', St), which is executed as a consequence of an operation (OP) invoked in another artifact. The requester artifact, Ar_1, sends the partial state of the agent (agent's internal state) that requested information to Ar_1, as well as a subset of its own internal information. As a result, the requested artifact, Ar_{inf}, will return some piece of information also based on its own internal state.

To exemplify how this type of artifacts work, we define an artifact that publicly provides norms currently active in the system. This artifact would contain operations shown in Figure 5.2.

Let $Ar_{inf}^{norms} = \langle \emptyset, \{requestNorms\}, \{linkInfo\} \rangle$ be an artifact that aims to provide agents (on demand) with information about norms, such as the specification of norms that rule a role, active or deactivated norms, etc. The operation $linkInfo$ may be used by other artifacts in order to gather information related to norms that could improve their usage. Thus, the artifact encapsulates functionality for both, agents that request information for their personal purposes, and other artifacts that could also be interested in some information that the artifact manages about norms.

We can observe that, in this example, it does not exist any observable property, since norms cannot be directly accessed by agents, but they may be requested by using the operation *requestNorms*. Consider that, since it is an informative artifact, the agent requesting for norms must send a part of its mental state in order to allow the artifact to give back some useful information.

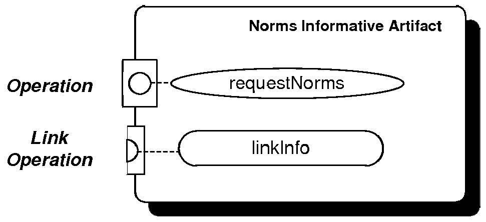

Figure 5.2: *Informative artifact for informing about norms*

Notice that this artifact is not a mere repository of norms, since it allows to be tuned to distinguish among different types of information that should be provided to agents. Thus, the designer probably does not want that any agent could know all the norms at any time, but it could probably prefer to give the precise information to the agent that is interested in, in such a way that it does not disclose any sensitive information.

A typical scenario would consist on an agent requesting for the set of norms that rule a specific role that the agent wants to play. Responsibilities, duties, and rights that roles specify for its enactment should make the artifact to provide suitable information on demand.

5.3.2 Incentive artifacts

Incentive mechanisms are mechanisms that are able to produce changes in the system environment from a global view, modifying the rewards and penalties that are active in the system. These mechanisms rely on the belief that a possibly little change in the incentive system (that can be affecting only to a small number of agents) affects the entire system. In this subsection, the incentive mechanisms are modeled as artifacts, named *incentive artifacts*. These artifacts will execute organizational changes, which bring the possibility of implementing an adaptive system, by varying elements from the system (e.g., adding or deleting norms). After a change in the incentive system of the MAS is produced, transition probabilities between different states of the system are affected. In order to carry out these changes, it is necessary to have an agent or a human playing a special role that we call "system adapter", which is able to manage organizational changes when necessary to promote the adaptiveness of the MAS. The system adapter is the only agent that has privileges to execute the operations of an incentive artifact.

Definition 18. *An **Incentive Artifact** is defined as an artifact* $Ar_{inc} = \langle PR, OP, LO, St\rangle$ *where:*

- $PR \subseteq \{St \cup \emptyset\}$ *are its observable properties;*
- $OP : \Delta$ *is the operation that allows the system adapter to introduce or remove incentives in the system;*
- $LO = \emptyset$*, since this type of artifacts has no predefined link operations;*
- St *represents the internal state of the artifact.*

The operation of the artifact (OP) modifies the transition probability between different states of the system. This operation is defined as:

$$\Phi = St \to [\mathcal{X} \times \mathcal{A}^{|\mathcal{A}g|} \times \mathcal{X} \to [0..1]], \text{ where:}$$

- Φ is the MAS transition probability distribution, describing how the environment evolves as a result of agents' actions.
- $\mathcal{X}$ is the environmental state space.
- $\mathcal{A}^{|\mathcal{A}g|}$ is the set of actions executed by agents between two states of the MAS.

This operation works as follows: the agent provides some piece of information to the artifact, which might change its internal state (St). Given this new internal state, the transition probability between two states of the system is modified, so the behavior of the MAS changes in a global perspective.

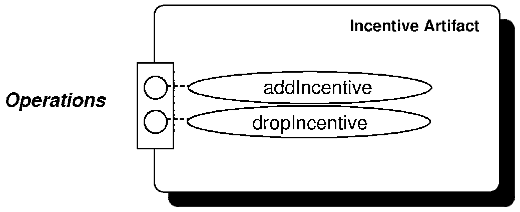

Figure 5.3: *Incentive artifact*

Figure 5.3 shows how an incentive artifact is modeled, including its minimum required features. The OP set contains two different operations: `addIncentive`($\{ty, ro, ac\}$), with $\{ty, ro, ac\} \in St$, which adds an incentive to the incentive system of the MAS; and `dropIncentive`($\{ty, ro, ac\}$), with $\{ty, ro, ac\} \in St$, which drops an incentive from the MAS. In these functions ty stands for the type of incentive (reward or penalty), ro refers to the role or set of roles affected by this incentive and ac represents a set of actions that are related with this incentive.

Sometimes, it could be useful to send information about changes in the incentive system to the agents populating the MAS. In order to execute this task it is necessary to provide the environment with an informative artifact, modeled as explained in the previous subsection.

To exemplify the incentive artifacts, we have chosen an organizational environment related to norms again.

Let $Ar_{inc}^{norms} = \langle \emptyset, \{addNormIncentive, dropNormIncentive\}, \emptyset \rangle$ be an incentive artifact that allows introducing positive incentives (*rewards*) and negative incentives (*penalties*) into an organization. These incentives consist of a set of possible consequences that norm fulfilment or violation, respectively, may entail. As aforementioned, the incentive mechanisms aim to improve the system performance by introducing changes in the environment that somehow influence the agents' reasoning. For this example, we consider that the artifact does not contain any observable property and that it does not offer any minimum link operation to be requested by other artifacts. The usage interface (OP) should not be available for every agent participating in the system. That is, this kind of artifacts does not provide information, but changes the environment, so only agents with sufficient permissions to do it should use operations in OP, depending on the domain. In our case agents capable of playing role "system adapter" can employ $addNormIncentive$ operation, so then attaching a penalty to a norm in case of violation; or introducing rewards for norm fulfilments. Incentives may also be updated through the time, by using $dropIncentive$ operation to remove the former and then updating with the new one by using $addNormIncentive$ operation.

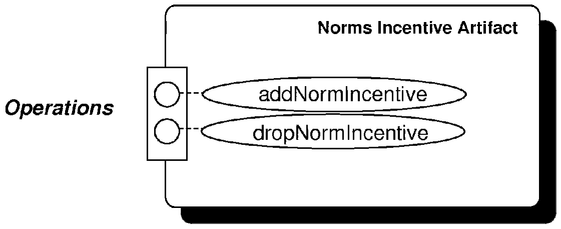

Figure 5.4: *Incentive artifact for norms*

5.3.3 Coercive artifacts

Coercive mechanisms are aimed to produce changes in the environment of the system by producing changes in the agents' capability functions, given a possibly partial description of MAS. As it occurs with incentive mechanisms, coercive mechanisms are also relying on the existence of the "system adapter" role, which is able to promote organizational changes.

Since in an open system it is not possible to modify the capabilities of the agents, there are alternative ways to do so. The common ones are to introduce norms that modify these capabilities, and to modify the role of the agents whose capabilities require to be changed to another role that gives the agent the desired capabilities.

Formally, a coercive artifact is defined as:

Definition 19. *A* ***Coercive Artifact*** *is an artifact* $Ar_{coe} = \langle PR, OP, LO, St \rangle$ *where:*

- $PR \subseteq \{St \cup \emptyset\}$ *are its observable properties;*
- $OP : St \rightarrow [\mathcal{A}g \times \mathcal{X} \times \mathcal{A} \rightarrow \{0,1\}]$ *is the operation carried out by the coercive artifact, where:*
  - $\mathcal{A}g$ *is an agent of the MAS;*
  - $\mathcal{A}$ *is the action space that includes all possible actions that can be performed in the system.*
- $LO = \emptyset$*, since this type of artifacts has no predefined link operations;*
- St *represents the internal state of the artifact.*

The operation $St \rightarrow [\mathcal{A}g \times \mathcal{X} \times \mathcal{A} \rightarrow \{0,1\}]$, given the artifact's internal state, returns the capability for executing an action or not, 1 and 0 respectively. Internally, this operation works as follows: the artifact needs its internal state (St) as well as the information provided by the system adapter ($\mathcal{A}g$ and $\mathcal{A}$) in order to execute this operation. After compiling all this information, the artifact calculates the new action space of the agent. This change can be seen as a local change but, since agents are related between them, changes in a single agent might produce changes in a set of agents, i.e. in the global state of the MAS.

Similarly to the informative and incentive artifacts, the coercive artifacts do not have mandatory observable properties. In case of having them, they are a subset of the internal knowledge of the artifact. The number of the available observable properties will depend on the purpose of each artifact. Additionally, they are not required to have link operations.

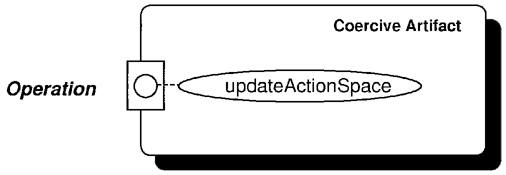

Figure 5.5: *Coercive artifact*

Figure 5.5 shows how a coercive artifact is modeled, including their minimum requirements. The only operation defined in this artifact is `updateActionSpace`($\{ag, a\}$), where $\{ag, a\} \in St$ and $ag \in \mathcal{A}g, a \in \mathcal{A}$, which receives an agent and and action from the system adapter and returns the capability for the given agent to perform this action.

As done with incentive artifacts, to show an example of coercive artifacts an organizational environment related to norms is taken:

Let $Ar_{coe}^{norms} = \langle \emptyset\ ,\ \{updateActionSpace\}\ ,\ \emptyset \rangle$ be a coercive artifact that aims to update agents' action spaces through time. As we stated in Chapter 2, Section 2.1.5, coercive mechanisms directly modify agents' action spaces to keep the former from undesirable behaviors. Thus, this artifact will be in charge of modifying those action spaces on demand of some special agents that have the permission to introduce these changes in the environment. Therefore, if one of the agents with sufficient permissions (i.e. "system adapter") observes that, for instance, the violation of a norm occurred, he could take the decision of banning some actions to the agent that did not fulfil that norm, trying to avoid that behavior in the future. In the same way that the artifact may remove actions from an agents' action space, it might also add actions to it, if agent's behavior is being acceptable. For instance, the system could test participants with a trial period to ensure that they behave accordingly to system's objectives, allowing them to perform more and more actions progressively.

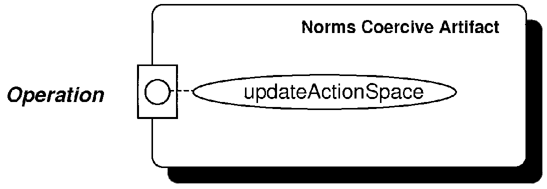

Figure 5.6: *Coercive artifact for norms*

Some examples of mechanisms that could be designed as incentive or regulative artifacts are: *normative manager*, that is, encapsulating dynamic consequences that fulfilment or violation of norms may entail; or *traffic sanctions manager*, where different sanctions may be applied about driving rules, even introducing constraints in the environment (roads can be closed, driver licenses could be taken away, etc.).

5.4 Artifacting the Environment

This section presents an extension of the Environment Dimension of the Virtual Organization Model (VOM) (AJB09) (see Figure 5.7) in order to improve the way that the environment is regulated in a MAS defined following the Virtual Organizational Model.

The Virtual Organization Model is an OML that follows a UML approach, and it is aimed to model Organization-Centered MAS, identifying the common elements that are present in an organization. As most of the metamodels, VOM also gives support to a software development methodology by upholding the development of the Virtual Organizations defined in GORMAS (ABJ11) methodology. VOM is composed of five *organizational dimensions*: structural, functional, dynamical, environment and normative, explained in Chapter 2. Systems defined by VOM are structured by means of the elements identified in these dimensions. VOM is related to VOF in the way that they are based in the same development methodology. However, they present differences due to the singularities of representing the an organization by means of a graphical notation or a formalization.

Figure 5.7 depicts the Environment Dimension of VOM. The main entities represented in this model are Resources and Applications. *Resources* are environmental objects that do not provide a specific functionality, but they are essential for the task execution. *Applications* are employed to model passive services i.e., a set of operations that do not require agent interaction to be provided and executed. Applications and resources are contained in organizational units (i.e., they can be seen by members of the organization) or they belong to agents. Access to applications and resources is carried out by means of *Environment Ports*, whereas *Service Ports* are employed for publishing services and controlling their access. Thus, a system designer can specify who is responsible for the management of each port, and also who will have reading and writing permissions over the resource, application or service that each port is responsible of.

This Environment Dimension is mainly focused on describing the elements of the environment (e.g., resources, agents and organizational units), the operations

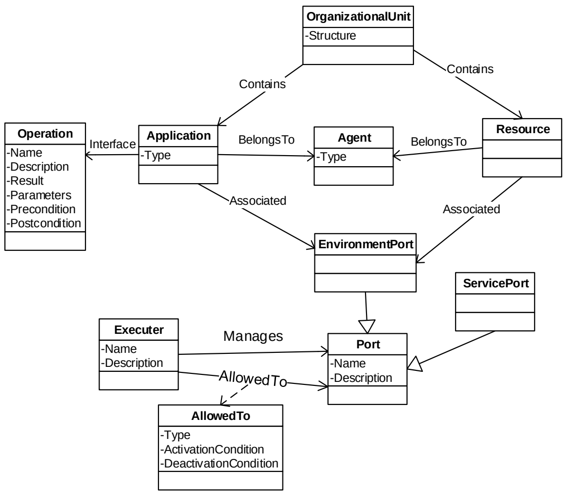

Figure 5.7: Environment Dimension of the VOM metamodel

that can be made over those elements (e.g., applications and services) and who is enabled to make use of those elements (e.g., by using ports). However, no mechanisms for regulating the environment can be easily defined, such as incentive or coercive mechanisms that modify the action space of agents. If needed, systems designers should try to define them by means of applications, norms, services or other elements not directly related with the environment, being completely lacking of guidelines from the metamodel.

Therefore, as previously explained, the main objective of this work is to integrate Artifacts for Organizational Mechanisms into the Virtual Organization Model so as to better assist the system designer when modeling the environment.

The extension of the Virtual Organization Model has been carried out in two different activities:

1. **Integrating applications, resources, and ports by means of arti-**

facts. The main objective of adding artifacts in VOM is to have available all the operations and observable properties of the three types of Artifacts for Organizational Mechanisms (informative, incentive, and coercive artifacts). Additionally, artifacts present features (e.g., observable properties) that encapsulate and extend functionalities provided by the resources, applications, and ports of the Environment Dimension.

2. **Clearly representing where environmental elements are located by using the concept of workspace.** Workspaces will help designers to physically structure the environment, so each entity inside the environment will have a specific location.

In the first activity, resources, applications, and ports have been integrated inside the artifact concept as follows: (i) **Resources** are entities that are provided with a set of values that can be checked, reduced and/or increased by agents. Since artifacts provide observable properties that can represent these values, resources have been replaced by artifacts (whose observable properties would be the set of checkable values of the resources) in the new proposal of the Environment Dimension. Note that to modify these properties, a set of operations (to increase or reduce a value) have to be established inside an artifact. (ii) **Applications**, which are interfaces for operations, can be implemented by means of artifacts since their operations can be directly translated into the artifact's operations. (iii) The access management of resources and applications carried out by the **Environment ports** can be handled now by artifacts, which are provided with an *internal state* and a *function description* (that acts as an operating manual) where the information about permissions is stored. For example, permissions may follow a Role-Based Access Control (RBAC) model (SCFY96), so the information related to the different roles would be stored in the artifact. For **Service ports**, whose functionality is to control the access to services and publicize them, the internal state and function description of an artifact are also employed to control permissions (again, a RBAC policy may be used), whereas services are publicized by means of the observable properties of the artifact.

In the second activity, the Environment Dimension of VOM is provided with a physical description of the environment by means of the concept of **workspace**, where artifacts are located. For each workspace, an absolute position (named location) is given. Workspaces of the system can be intersected and nested, features that allow artifacts to be placed in different workspaces, so they are able to be perceived and used from various locations. Agents and Organizational Units are related to entities from the Environment Dimension. In this case, an agent is able to perceive a set of workspaces, and use the artifacts that are located inside this set. Additionally, an Organizational Unit can be placed inside a workspace, so this organizational unit has a specific location inside the organizational environment.

It should be noted that the organization is represented as the whole set of organizational units that make it up. Moreover, the environment of a system is the aggregation of all the workspaces of the system, where the organization to be designed will be located. The addition of workspaces makes the designer able to model the environment in a similar way than the real-world environment would be designed. The previous version of VOM was not able to explicitly define the visibility or the particular location of each element inside the environment. Using our proposed extension, designers can now clearly define these features.

As depicted in Figure 5.8, the new version of the Environment Dimension is now structured around the *Artifact* entity. There are three different types of artifacts, representing the three types of Artifacts for Organizational Mechanisms, inheriting from the main artifact entity. Each type of artifact is represented in the metamodel by means of its particular operations and observable properties.

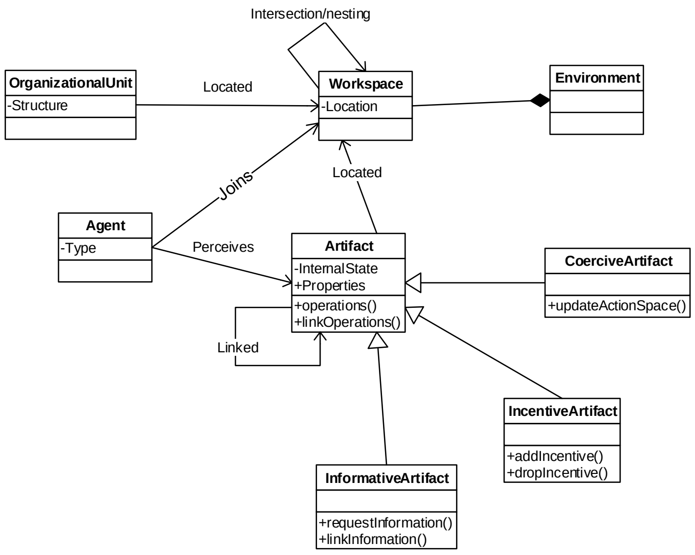

Figure 5.8: *New Environment Dimension including artifacts and workspaces*

Informative artifacts are provided with operations that allow agents (and other artifacts) to request information. Incentive artifacts, whose goal is to modify the reward system of the MAS, are enhanced with operations for adding and deleting incentives from this reward system. Finally, coercive artifacts are able to modify the action space of an agent by means of its particular operation.

These modifications enhance the metamodel by making it a proper coordination system for a Virtual Organization, and improving the way how the environment is regulated. As explained before, in the previous version of VOM, the environment could have been regulated by means of applications, services, or norms, but since these elements were not defined with this objective in mind, introducing mechanisms for regulating a system might imply a rather hard work for designers. With this new addition, environmental regulations can be explicitly developed using Artifacts for Organizational Mechanisms (whose access permissions are controlled by the artifacts themselves, being not necessary to use ports). By including Artifacts for Organizational Mechanisms inside the Environment Dimension of VOM, the means for introducing them inside an agent-based system has been provided here.

5.5 Case Study: Inter-Hospital Transfer Coordination

In this section we focus on a real problem in the health care domain. The domain of medical assistance, in general, includes many tasks that require flexible on-demand negotiation, initiation, coordination, information exchange and supervision among different involved entities (e.g., ambulances, emergency centers, hospitals, patients, physicians, etc.). In particular, we focus on coordination of inter-hospital transfers. This task is performed by the SUMMA112, which is the emergency center in charge of providing sanitary assistance to urgencies, emergencies, catastrophes and special situations in the Autonomous Region of Madrid, in Spain. The aim of this task is to coordinate the transfer of patients among different hospitals, for example, in order to provide a specific treatment.

Let us show an example of this problem. Juan Domínguez, 23 years old, is waiting for assistance in the *Hospital de El Escorial*,[1] where he has been detected to have appendicitis. In this case, laparoscopic surgery is required. Juan would need to stay for another two days in observation, and would have to come again two weeks later for cure. After the diagnosis, the physicians decided that he should be treated in the next 24 hours. However, because of an excess of operations at this time in that hospital, there is no means to provide the appropriate treatment

[1]It is a hospital located in the north of the region of Madrid

in this hospital. Thus, Juan should be transferred to another hospital that has sufficient capacities to treat his case.

The administrative staff at *Hospital de El Escorial* calls the SUMMA112 coordination center to request the transfer. Therefore, the center is in charge of providing a solution to such a problem by assigning a destination hospital to that patient.

The solution adopted by the SUMMA112 is currently based on an agreement process between the SUMMA112 and the hospitals that have enough resources to treat the patient and, at the same time, they want to treat the patient. That is, sometimes hospitals have the resources to treat the patient but for any reason, they are not interested in taking this kind of patient e.g., because they are carrying out a clinical trial regarding patients suffering another kind of disease.

Once the problem definition has been presented, we model different solutions to such a problem based on Artifacts for Organizational Mechanisms. Following an agent-based approach, each hospital could be modeled as an autonomous agent participating in the system whose objective is to plan successfully the inter-hospital transfer. Besides, the system's administrator (in our case it could be the own SUMMA112 coordination center) is able to endow the environment with different organizational artifacts, aiming to fulfill the global objective.

5.5.1 Solution based on an informative artifact

The first adopted solution relies on an informative artifact, $Ar^{arg}_{inf} = \langle \emptyset, \{newPatient, writeArguments, getArguments, getCommonSets\}, \emptyset \rangle$. In particular, such a mechanism encapsulates an argumentation algorithm able to propose a solution for a problem, based on arguments sent by different agents. Adapting it to our particular example, hospitals, represented in the system by autonomous agents, could argue about who of them would be in charge of attending a particular patient. They send their arguments, and the artifact calculates a solution over the shared arguments. Finally, the proposed solution is offered by the own artifact, and the coordination center by querying it, adopts such a solution.

As argumentation algorithm, we use the one presented by Oliva *et al* (OMO08). The objective of that algorithm is to achieve a solution based on the arguments sent by agents. This process is divided in three steps: (i) agents share their arguments by sending them to the artifact (using operation `writeArguments`); (ii) the artifact executes the algorithm and calculates the conflict free and preferred extension over the shared arguments; and finally (iii) agents can query the artifact for getting the common sets calculated (operation `getCommonSets`). Besides, agents can also consult the artifact so as to understand and know other agents' arguments (by means of operation `getArguments`).

Therefore, in our example, when a new patient has to be transferred, the coordination center informs both, agents and artifact, about it, providing its particular characteristics and requirements. After that, agents send their arguments so as to (or not to) accept that patient. Then, the informative artifact calculates which agent (hospital) wins the argumentation process, in other words, which hospital has to admit such a patient. Figure 5.9 depicts the informative artifact based on argumentation in charge of providing the solution.

Let us see a particular example showing how it works (see Figure 5.10). The *"Hospital de El Escorial"* calls the SUMMA112 coordination center to request the transfer of the patient called Juan. Then, all hospitals represented by their agents join the system with the global objective of assigning a destination to Juan. Let us suppose that only two hospitals join the system: *"Hospital de Móstoles"* and *"Hospital Reina Sofia"* represented by *agent-1* and *agent-2*, respectively. Once they are in the system, the coordination center, represented

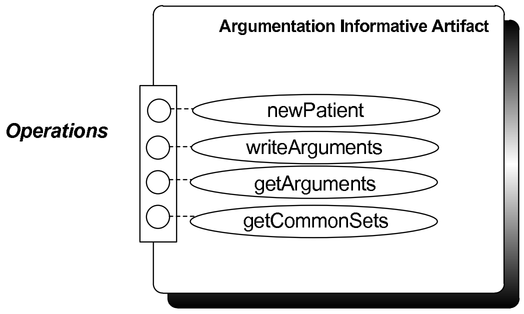

Figure 5.9: *Informative artifact encapsulating an argumentation algorithm*

by *agent-cc*, introduces the record of the new patient (`[Juan,23,‘‘Hospital de El Escorial",appendicitis]`, i.e. name, age, origin hospital and diagnosis) in the informative artifact, by using the `newPatient` operation. After that, *agent-cc* asks hospitals for sending their arguments with regards to accept the new patient. In this case, both agents representing hospitals send their arguments to the informative artifact; *agent-2* sends the argument `[not enough bed capacities]`, while *agent-1* sends `[carrying out clinical trial regarding appendicitis]`. When both arguments are sent, the informative artifact calculates the result based on the argumentation algorithm and both hospitals and coordination center ask the informative artifact for the identifier of the patient and the artifact returns the destination hospital. In this example, the agent that clearly wins the argumentation is the *agent-1* since it is interested in admitting the patient while *agent-2* is not. Therefore, the informative artifact returns `[agent-1, ‘‘Hospital de Móstoles"]` when it is queried with `[patient, Juan]`. Finally, the coordination center is able to plan the transfer preparing the necessary resources (ambulance, medical record, etc.).

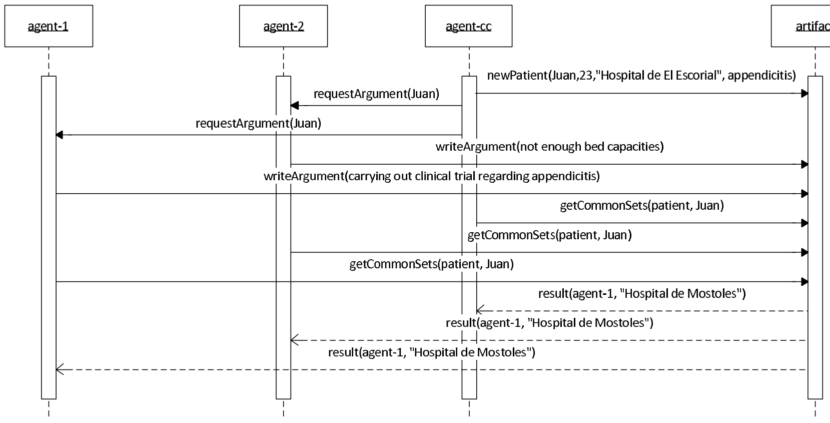
agent-1
agent-2
agent-cc
artifact
newPatient(Juan,23,"Hospital de El Escorial", appendicitis)
requestArgument(Juan)
requestArgument(Juan)
writeArgument(not enough bed capacities)
writeArgument(carrying out clinical trial regarding appendicitis)
getCommonSets(patient, Juan)
getCommonSets(patient, Juan)
getCommonSets(patient, Juan)
result(agent-1, "Hospital de Mostoles")
result(agent-1, "Hospital de Mostoles")
result(agent-1, "Hospital de Mostoles")

Figure 5.10: *Sequence diagram for the solution based on argumentation*

5.5.2 Solution based on an incentive artifact

The solution presented in section 5.5.1, actually, is the solution adopted nowadays by the SUMMA112 coordination center, in the region of Madrid.[1] However, such a solution suffers from some potential problems that, in fact, are currently happening. For instance, hospitals sometimes use false arguments so as to avoid to admit certain patients that could require a long or complicated treatment.

Dealing with these problems we propose another solution relying on an incentive artifact coupled with an informative one. The objective of the incentive artifact is to modify the consequence of some actions so as to agents have incentives to perform (or not) such actions. In this case, the coordination center is interested in agents to perform the action `admit a patient`. Thus, since the coordination center knows that hospitals are interested in having a high budget, the consequence of such an action will be modified such that the hospital's budget will be increased when it performs that action.[2] Besides, in order to allow agents

[1]The solution is not agent-based but human-based.

[2]Actually, lots of fluctuations in a hospital's budget do not make any sense, thus, the new

to reason about the new consequences, this information will be provided to agents by querying an informative artifact (see Figure 5.11).

Therefore, when the coordination center receives a new request to transfer a patient, it joins the system, as well as all hospitals, with the aim of assigning a destination to such patient. Since the coordination center (*agent-cc*) is the administrator of the system, it has enough permission to directly operate with the incentive artifact. Thus, when it joins the system it employs the operation `addIncentive(reward, hospital, admit patient)` (or `dropIncentive(reward, hospital, admit patient)`) so as to add (or drop) the incentive that gives a *reward* to agents playing the role *hospital* when they perform the action *admit patient*. Moreover, in order to allow agents to reason about the reward added to the system, there exists an informative artifact that provides the new consequences of that action. In this case, the artifact will provide the consequences (reward) of the action *admit patient*. This artifact, by means of a link operation with the incentive artifact (`addRewardInformation`), receives the consequences and the action it has to inform about. Therefore, agents are able to know about consequences of different actions before carrying them out.

Let us see how our example is solved now by using the proposed solution (see Figure 5.12). Again, Juan needs to be transferred to another hospital. Thus, *"Hospital de El Escorial"* calls the SUMMA112 coordination center (*agent-cc*) and joins the system together with *"Hospital de Mostoles"* (*agent-1*) and *"Hospital Reina Sofia"* (*agent-2*). Once all of them are in the system, *agent-cc* employs the operation `addIncentive(budget+1000, hospital, admit patient)` provided by the incentive artifact. At the same time, the incentive artifact employs the operation

consequences should be modified when a number of patients is admitted over a period of time, however, for the sake of simplicity and clarity, we assume that it is always modified when a hospital admits a patient.

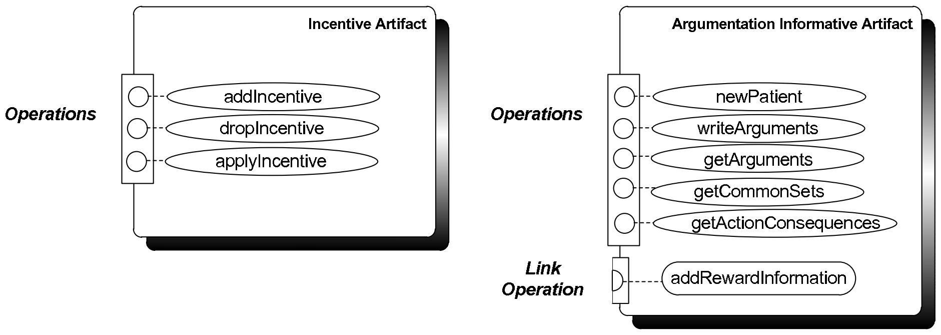

Figure 5.11: *Incentive and informative artifacts used in this solution*

`addRewardInformation(admit patient, budget+1000)`, which is a link operation, over the informative artifact. Then, *agent-cc* informs hospitals, as well as the informative artifact, about the new patient, providing them with his information (`[Juan, 23, ‘‘Hospital de El Escorial", appendicitis]`). Once agents receive a message about a new patient, they have to deliberate on admitting it or not. Since agents already know about an informative artifact that informs about consequences of actions[1] they query it (using operation `getActionConsequences`) providing two actions: `admit patient` and `reject patient`. In this case, let us suppose that *agent-2*'s argument about not having enough bed capacities was false, and now, knowing the new consequences for admitting a patient, it decides to admit Juan. Therefore, both hospitals are now interested in admitting Juan, and inform the *agent-cc* about it. So, the coordination center has to make a decision about which of both hospitals will be the destination for Juan. This decision could be taken based on some algorithm running in the own coordination center, or even the argumentation artifact presented in section 5.5.1 could be used, com-

[1]How informative artifacts are discovered is out of the scope of this work. It could be done either by announcing them in a meta-informative artifact belonging to all systems, or because agents know about the existence of artifacts located at the same workspace as they are.

bining both solutions. Finally, once the *agent-cc* makes the decision, it informs hospitals about the patient's destination and it is the *agent-cc* itself who invokes the operation *applyIncentive* in the incentive artifact so as to give the winner the promised reward.

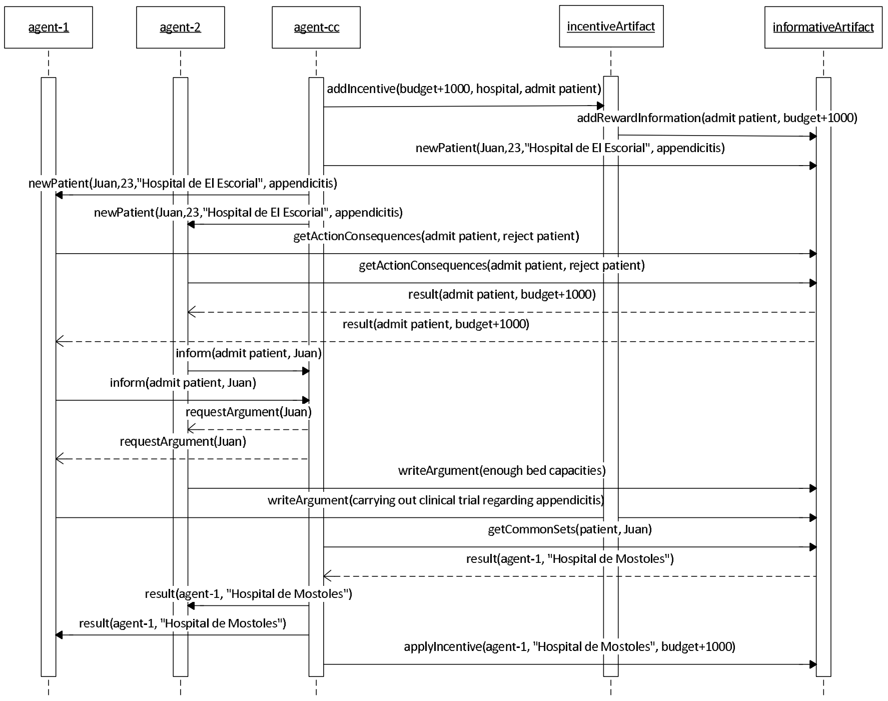
agent-1
agent-2
agent-cc
incentiveArtifact
informativeArtifact
addIncentive(budget+1000, hospital, admit patient)
addRewardInformation(admit patient, budget+1000)
newPatient(Juan,23,"Hospital de El Escorial", appendicitis)
newPatient(Juan,23,"Hospital de El Escorial", appendicitis)
newPatient(Juan,23,"Hospital de El Escorial", appendicitis)
getActionConsequences(admit patient, reject patient)
getActionConsequences(admit patient, reject patient)
result(admit patient, budget+1000)
result(admit patient, budget+1000)
inform(admit patient, Juan)
inform(admit patient, Juan)
requestArgument(Juan)
requestArgument(Juan)
writeArgument(enough bed capacities)
writeArgument(carrying out clinical trial regarding appendicitis)
getCommonSets(patient, Juan)
result(agent-1, "Hospital de Mostoles")
result(agent-1, "Hospital de Mostoles")
result(agent-1, "Hospital de Mostoles")
applyIncentive(agent-1, "Hospital de Mostoles", budget+1000)

Figure 5.12: *Sequence diagram for the solution based on an incentive artifact*

5.5.3 Solution based on a coercive artifact

So far we have put forward two different solutions for the inter-hospital transfer of patients, based on informative and incentive artifacts, respectively. However, there are still some cases in which outcomes resulting from the negotiation for the transfer are not as they should be expected. The solution presented above attempted to incentive hospitals to admit patients by rewarding them for their positive attitude. Nevertheless, this could bring a drawback on the results. That solution fosters the positive behavior of hospitals but does not keep them from not accepting patients they are not interested in. A solution for avoiding this behavior could be to add an incentive artifact that tries to maximize the number of positive bids for accepting a patient when a transfer is needed. It would be designed in exactly the same manner than the one proposed in section 5.5.2 but imposing negative sanctions for not accepting patients, e.g. by reducing budget in some medical services. Even though, this solution entails some problems for any transfer negotiation process. In the end of any transfer negotiation process there is only one “winner”, what means that only one hospital admits the patient. Facing this fact, a question raises: among the remainder hospitals that have not ”won” the patient, which of them should be punished? It seems clear that a difference should be made between those which bade for the patient and those who did not. Even if the hospital really wants to accept the patient but it does not get it, it would not be considered the same as if a hospital is not interested in the patient and does not bid for him/her. Some other non-desirable cases could happen, such as, for instance, a hospital that bids for a patient, but with a rather low level of interest because it previously knows that the patient will not be assigned to it. This last case cannot be avoided by using incentive artifacts.

For the reasons exposed above we point out another solution, based on the addition to the system of a coercive artifact (see Figure 5.13) to keep hospitals from not accepting patients in which they are not interested in. The objective of coercive artifacts lies in the need of producing changes in the environment of the system by producing changes in the agents’ capability functions. In this case, we suggest allowing the artifact to close certain medical services in hospitals that do not bid for patients that could be potentially treated by those same services.

Closing services is equivalent to ban the action of accepting patients for that service in that hospital. Nonetheless, it is reasonable to think that a medical service cannot be removed suddenly just because the hospital that it belongs to rejected a patient. Thus, an intermediate solution would be to close a service only if the number of rejections of patients to be treated in it excesses a given threshold.

As we have previously mentioned, the coordination center (*agent-cc*) acts as a system administrator, so it is supposed to have enough permissions to be able to modify the environment, that is, it has permissions to modify the artifact. Then, *agent-cc* will use the operation `updateActionSpace(constraint, hospital, admitPatient)`, which means that a new constraint is introduced into the system affecting agents playing role *hospital* regarding the action *admitPatient*. Moreover, as it occurred in the case of adding an incentive artifact, an informative artifact would be needed in order to show agents the consequences of not fulfilling new constraints. In our case, this artifact will provide the consequences of not admitting new patients. This artifact will contain a link operation that will be invoked by the coercive artifact so as to inform agents about new consequences about admitting a patient.

Following, we explain how this solution fits in our case study, as depicted in Figure 5.14. *"Hospital de El Escorial"* needs to transfer Juan to another hospital. With this purpose it contacts SUMMA112 coordination center, represented by *agent-cc*, joining the system as well as *"Hospital de Móstoles"*, represented by *agent-1*, and *"Hospital Reina Sofia"*, represented by *agent-2*. Once this situation is reached, *agent-cc* uses the operation `updateActionSpace(closeMedicalService, hospital, rejectPatient)` of the coercive artifact explained above. Then, this artifact uses link operation `addConstraint(rejectPatient, closeService)` of the informative artifact to show consequences of the action *rejectPatient*. Even if

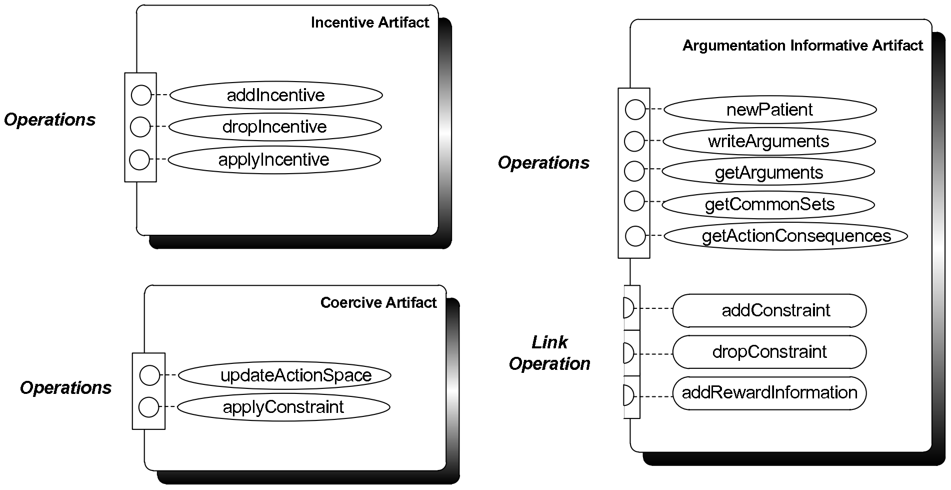

Figure 5.13: *Coercive, incentive and informative artifacts used in this solution*

the consequences do not specify the threshold of non-admitted patients for closing a service, the information about the possibility of closing a service should be enough to the hospitals for being concerned.

Then, as occurred in the other solutions the *agent-cc* informs hospitals about the new patient waiting for being transferred, providing them with this information (`[Juan, 23, ‘‘Hospital de El Escorial", appendicitis]`). Let us suppose that this coercive artifact is put in the environment together with the incentive and informative artifacts explained in the previous subsections. Then, agents representing hospitals will send queries to the informative artifact in order to know consequences of admitting or rejecting new patients (by means of `getActionConsequences`). Later, agents have to decide if their arguments, given in the first solution, are strong enough or they might be changed. If several hospitals bid for admitting the patient, the final choice of deciding the winner will correspond to the *agent-cc*, possibly based on the arguments given by an argumentation informative artifact (as proposed in the solution presented in section 5.5.1, if applied). After selecting the "winner" hospital, then *agent-cc* informs all participants and it invokes operation `applyConstraint(agent, rejectPatient)` for every hospital that rejected to accept Juan. Thus, the artifact reviews the rejections record of those agents that refused to admit the patient and decides if they should be penalized by closing one of their medical services, for instance, the one with less productivity.

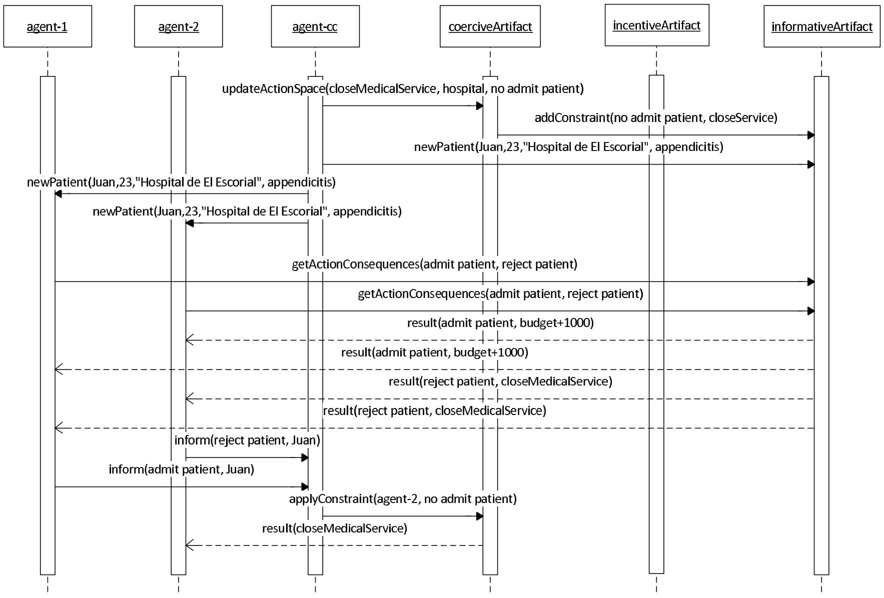
agent-1
agent-2
agent-cc
coerciveArtifact
incentiveArtifact
informativeArtifact
updateActionSpace(closeMedicalService, hospital, no admit patient)
addConstraint(no admit patient, closeService)
newPatient(Juan,23,"Hospital de El Escorial", appendicitis)
newPatient(Juan,23,"Hospital de El Escorial", appendicitis)
newPatient(Juan,23,"Hospital de El Escorial", appendicitis)
getActionConsequences(admit patient, reject patient)
getActionConsequences(admit patient, reject patient)
result(admit patient, budget+1000)
result(admit patient, budget+1000)
result(reject patient, closeMedicalService)
result(reject patient, closeMedicalService)
inform(reject patient, Juan)
inform(admit patient, Juan)
applyConstraint(agent-2, no admit patient)
result(closeMedicalService)

Figure 5.14: *Sequence diagram for the solution based on a coercive artifact*

Applying an incentive artifact as the one presented in Section 5.5.2 we do not avoid a non-collaborative behavior of the hospitals, but we just give rewards to those which are collaborative. With the solution proposed in current section we try to constraint that potential lack of collaboration of some hospitals, so trying to assure that when a hospital rejects an admission is because it cannot treat the patient properly.

5.5.4 Modeling the solution with VOM

VOM is used here to model the solution, representing all the participating entities. VOM uses GOPPR (KLR96) notation to represent its entities. Figure 5.15 depicts a caption including all the entities that will be used to model this case of study.

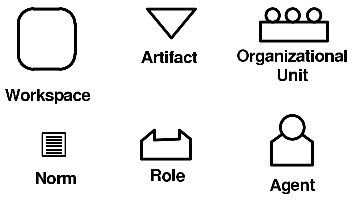

Figure 5.15: Caption for some of the VOM entities

Prior to the modeling of the environment, "Plays" relationships between roles, and agents and organizational units should be represented inside the Dynamical Dimension of VOM (see Figure 5.16). In this dimension, features such as interactions between agents or the role enactment process are described. In our scenario, hospitals, the negotiation table, Madrid's Health Care System and SUMMA112 are modeled by means of organizational units.

Using these entities from VOM, we are able to model the environment of this case of study. As it can be seen in Figure 5.17, the main entity of this diagram is an organizational unit representing the health care system of the region of Madrid. Furthermore, let us suppose that this Organizational Unit (OU) is containing other OUs representing the hospitals that participate in this scenario (*Hospital de El Escorial*, *Hospital de Móstoles*, and *Hospital Reina Sofía*), the SUMMA112 service and the negotiation table. Each hospital, and the SUMMA112 service, are

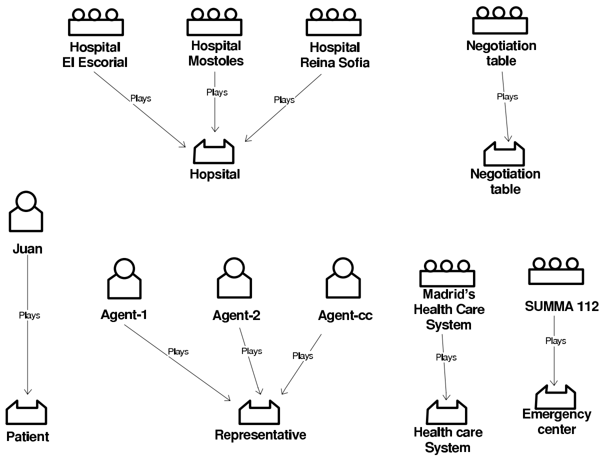

Figure 5.16: Dynamical Dimension, depicting organizational units, agents and roles

provided with a representative agent. Also Juan, the patient, which is the main reason for creating this scenario, is represented by means of an agent entity.

The required artifacts for organizational mechanisms are located inside the same workspace as the OU that represents the health care system (see Figure 5.17). Thus, agents belonging to the Madrid's Health Care System can make use of these artifacts. More specifically, representative agents are allowed to use informative artifacts to obtain the information they require during this process. Since the coordination center is the administrator of the system, it has enough permissions to operate directly with the incentive and coercive artifacts, in order to introduce incentives and sanctions into the negotiation process. Additionally, incentive and coercive artifacts are linked to the informative artifact since they can send information that will be incorporated into the internal state of the informative artifact, making this information available to representative agents.

It should be noted that this model only represents the interactions at de-

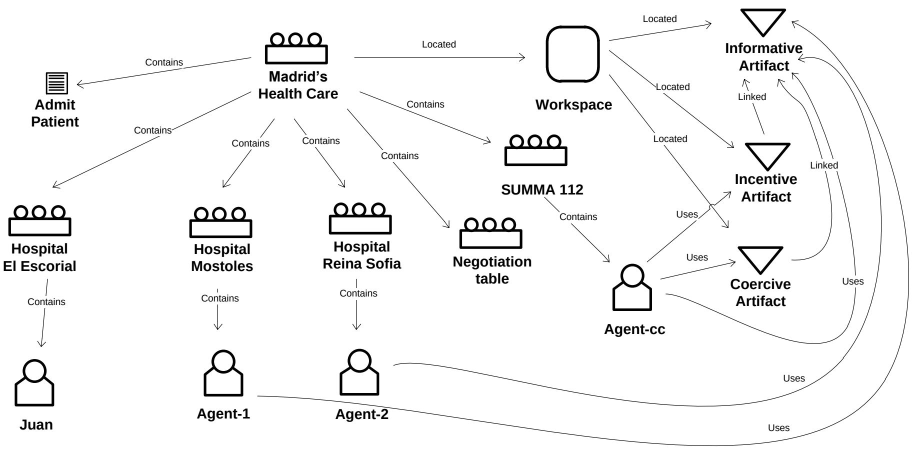

Figure 5.17: Environment Dimension of our case of study

sign time, and it does not represent changes at runtime. Thus, in this example, agents *Agent-1* and *Agent-2* (representatives of the hospitals) and *Agent-cc* from SUMMA112 will have a meeting inside the Negotiation Table OU, thus they will have to move from their own OUs to the Negotiation Table.

Table 5.1 gives a description of the different functions implemented inside the artifacts that are employed in this example.

The proposed solution points out some advantages of using the VOM model extended with Artifacts for Organizational Mechanisms. In order to develop the same domain (and scenario) with the previous Environment Dimension of VOM (i.e., without the extension), regulation would have had to be achieved by using *application* and *resource* entities to model regulation processes and *ports* in order to enable access to them. With the proposed model extension, oriented to regulation design, the artifacts for organizational mechanisms allow an easier design, since they permit to deploy any kind of regulating mechanisms for the system from a common interface for different types of regulation mechanisms (informative,

Functions of the Artifacts for Organizational Mechanisms	
Name	**Description**
Informative Artifact	
newPatient	Inserts the information about a new patient in the internal state of the artifact
addRewardInformation	A link operation that inserts the information about a new reward in the internal state of the artifact
writeArguments	Inserts an argument from an agent in the internal state of the artifact
getArguments	An operation that returns the arguments provided by hospitals
getActionConsequences	Returns the information about carrying out an action
getCommonSets	Returns the common sets computed after an argumentation process
addConstraint	Inserts the information about a new constraint being applied in the internal state of the informative artifact
dropConstraint	Removes the information about a constraint being applied in the internal state of the informative artifact
Incentive artifact	
applyIncentive	Inserts the information about a new incentive being applied over an artifact in the internal state of the informative artifact
addIncentive	Introduces a new incentive in the reward system of the organization
dropIncentive	Removes an incentive from the reward system of the organization
Coercive artifact	
updateActionSpace	Updates the action space of an agent by allowing or forbidding him to carry out actions
applyConstraint	Applies a constraint over a hospital of the organization

Table 5.1: Operations that are implemented in the Artifacts for Organizational Mechanisms of the case study

incentive, or coercive).

5.6 Discussion

The aim of this section is twofold. On the one hand, it is aimed to compare our proposal with other available artifacts presented by different authors, in order to determine whether existing artifacts have features of informative, incentive or coercive artifacts (see section 5.6.1). On the other hand, a discussion about OMLs is presented in terms of how they model the environment (in section 5.6.2), in order to present a perspective on the design of OCMAS.

5.6.1 Comparison with different artifacts

Some of the artifacts presented by the community of researchers provide information to agents after receiving information about a partial view of their internal state, so they can be seen as informative artifacts. Table 5.2 depicts the different types of artifacts identified (EA10) related with the artifacts for organizational mechanisms that they can be designed with.

For example, the *Role Evolution Coordination Artifact* (HBO10), which is aimed to build and evolve a role specialization taxonomy, consisting on a set of roles with a specific order, over time; and make this information available to the agents. This artifact contains three operations: (i) *getBestRolesForInteraction*, which provides the most specialized roles for a given service type interaction; (ii) *getAgentsForRoles*, which provides the set of agents that play at least one of the roles in a given set of roles; and (iii) *getRolesForAgent*, which provides the set of roles that a given agent plays in the system.

Previously, in section 5.3.1, we have formally defined the operation of an informative artifact as $\mathcal{S}' \times St \to \mathcal{I}$. A correspondence between the operations of the *Role Evolution Coordination Artifact* and the operation of an informative artifact can be established. In this way, the *getBestRolesForInteraction* operation function can be described as follows:

- $\mathcal{S}' = Serv$, where $Serv$ is a service type interaction.
- $St = R$, where R is the complete set of roles of the MAS.
- $\mathcal{I} = \mathcal{P}(R)$, where $\mathcal{P}(R)$ are the most specialized roles for S.

Similarly, the function *getAgentsForRoles* has the following correspondence:

- $\mathcal{S}' = \mathcal{P}(R)$, where $\mathcal{P}(R)$ is a set of roles.

- $St = \mathcal{A}g$, where $\mathcal{A}g$ is the complete set of agents of the MAS.
- $\mathcal{I} = \mathcal{P}(\mathcal{A}g)$, where $\mathcal{P}(\mathcal{A}g)$ is the set of agents that play at least one of the roles in $\mathcal{P}(R)$.

Finally, the function *getRolesForAgent* presents the following correspondence with an operation of an informative artifact:

- $\mathcal{S}' = \mathcal{A}g$, where $\mathcal{A}g$ is an agent of the system.
- $St = R$, where R is the complete set of roles of the system.
- $\mathcal{I} = \mathcal{P}(R)$[1], where $\mathcal{P}(R)$ is the set of roles that $\mathcal{A}g$ plays in the system.

Another example of artifacts that can be considered as informative artifacts are the argumentation artifacts (OMO08), being the *Co-Argumentation Artifact* (CAA) (OMO08) the most recognized one. This artifact gives assistance to argumentation processes. Participating agents share their arguments (i.e. a partial view of their internal state) with the artifact, which collects this information. Then, the artifact evaluates the arguments provided by all agents and calculates both the "social acceptability" (the acceptability of the arguments of a specific agent) and the "social behavior" (the acceptability of the arguments from a global perspective). Using both values, agents take a final decision in their argumentative process, so their respective behaviors change.

The CAA implementation provides two observable properties (*Social Behavior*, *Social Acceptability*) and one operation (*writeArguments*), which allows agents to store their arguments in the artifact.

Following the formalization of artifacts for organizational mechanisms, this CAA can be modeled as both an informative artifact and an incentive artifact. In

[1]$\mathcal{P}(R)$ stands for the power set of R.

this case, this artifact can be implemented with two different operations: *getSocialValues* and *writeArguments*.

It is possible to establish a correspondence between the operation *getSocialValues* of a CAA and the required operation of an informative artifact. The partial description of an agent's internal state ($\mathcal{S}'$) is represented in a CAA as the argument that the agent will use during the argumentation process.

$$\mathcal{S}' = Arg_t \tag{5.1}$$

where Arg_t is an argument provided by the agent t.

In this example, the internal state of the artifact (St) is the set of arguments that it has stored up to this moment.

$$St = \bigcup_{i=1}^{n} Arg_i \tag{5.2}$$

where St is the compilation of n arguments.

Finally, the information ($\mathcal{I}$) returned by the artifact are the values of the Social Acceptability and Social Behavior:

$$\mathcal{I} = \{SocAcc, SocBeh\} \tag{5.3}$$

where $SocAcc$ is the Social Acceptability and $SocBeh$ is the Social Behavior.

In a similar way, other types of artifacts, such as *Coordination artifacts* (ORV$^+$04), or *Organizational artifacts* (HBKR10) can also be described using artifacts for organizational mechanisms. Therefore, since coordination artifacts encapsulate a coordination service, this coordination service can be implemented by means of an informative artifact (providing useful information to the agents), an incentive artifact (modifying the transition probability between different states of the system), or a coercive artifact (allowing or banning agents from developing different actions). For example, the *Follow Me* coordination artifact (ORV$^+$04) is considered as an informative artifact, since its provided information is useful for agents in order to know their next action to execute (using *get* operation). Agents are also able to request the artifact to execute an action using the $do(a)$ operation. Formally, $do(a)$ operation from the *Follow Me* example can be described as:

- $\mathcal{S}' = \emptyset$, since an agent does not need to provide information from his internal state.
- $St = \bigcup_{i=1}^{n} \mathcal{A}_i$, the set of available actions, where n is the total number of actions contained in the artifact.
- $\mathcal{I} \in \{true, false\}$ confirms whether the action is executed or not.

The second operation from the *Follow Me* Artifact, (i.e., *get*) can be formally defined as:

- $\mathcal{S}' = \mathcal{A}g$, being $\mathcal{A}g$ the identifier of the agent that requests the operation.
- $St = \bigcup_{i=1}^{n} \mathcal{A}_i$, the set of available actions, where n is the total number of actions contained in the artifact.
- $\mathcal{I} = \mathcal{A}_i$ is the next action that the requester agent has to execute.

The observable property of this artifact is $act(a)$, that represents whether an action a is completed or not.

Regarding *organizational artifacts* (HBKR10), they are used to manage an agent organization in order to help the organization to reach its goals from a global, social level. A clear example of this type of artifacts is an artifact that helps informing or managing norms that, as it has been previously explained along Section 5.3, it can be modeled as an informative artifact (providing norms currently active in the system), an incentive artifact (introducing positive or negative incentives into an organization) or a coercive artifact (removing actions from agent's action space or including new possible actions, i.e. when an agent takes or leaves a role).

The examples of organizational artifacts proposed in ORA4MAS (HBKR10) include four artifacts that can be identified with the different types of artifacts for organizational mechanisms. These four artifacts are: OrgBoard, GroupBoard, SchemeBoard, and NormativeBorad. *OrgBoard* is an artifact that returns information about different aspects of the organization such as the agents playing a role, so it is clearly an informative artifact. But its link operation, used to add new elements inside an OrgBoard artifact, changes the system elements, so this link operation features properties from incentive artifacts. *GroupBoard* artifact is an incentive and coercive artifact because of its operations. It also has properties from informative artifacts, given by its $isMember$ link operation, which given an agent returns whether an agent is a member of a group or not. This artifact is focused on the role enactment process of the system. Therefore, if an agent adopts or leaves a role he will be affected by a different reward system, according to the adopted/left role. Also, taking or leaving a role in different proposals such as Electronic Institutions (e-Institutions) (ERAS$^+$01) or THOMAS (GJR$^+$09) changes the available actions that an agent is able to perform, so it is a feature of a coercive mechanism. As an example, *AdoptRole*(ρ) operation from GroupBoard artifact, used by an agent to adopt a new role, being ρ the role to be adopted, is formalized using two types of operations, related to incentive and coercive artifacts. The operation related to incentive artifacts refers to the norms that affect the role being taken. Then, the reward system changes, thus modifying the probability of changing the system from one state to another. On the other hand, taking a role in the beforehand mentioned proposals also modifies the set of actions that an agent can perform, so this is a feature related to coercive artifacts. Formally, as explained before (in section 5.3.3) an operation from a coercive artifact is defined as $St \rightarrow [\mathcal{A}g \times \mathcal{X} \times \mathcal{A} \rightarrow \{0, 1\}]$.

In this particular case:

- St is the internal knowledge of the *GroupBoard* artifact.
- $\mathcal{A}g$ is the agent taking the role ρ.
- $\mathcal{X}$ is the current environmental state.
- $\mathcal{A}$ is an action from the action's space.

Therefore, after executing the *adoptRole* operation, tuples containing the information about the agent's action space are added to the internal state of the artifact.

ORA4MAS's *SchemeBoard* is an incentive artifact, since it is focused on missions of the system. An agent is able to commit or leave a specific mission, and missions are enhanced with a set of obligations, thus this implies changing the reasoning of an agent. Finally, the *NormativeBoard* artifact presents features from informative and incentive artifacts. It is informative because it provides information about norms. A norm is able to change the reward system that an agent is affected by, thus also defining an incentive artifact.

Reputation artifacts (HBV10) encapsulate the collection of norm violations of the participants in a system and then aggregate them allowing agents to request information from a specific agent, which is able to change the behavior of an agent. Therefore, a reputation artifact presents the properties of an informative artifact. For example, the reputation artifact (HBV10) that helps Alice, a MSc. student, to select the best partner to write a paper with, it uses experiences previously collected and evaluated by the artifact, so Alice will be able to build a reputation for each partner based on the computed evaluation.

Artifacts for Cognitive Stigmergy (ROV$^+$07) are implemented to promote awareness inside a system. Agents are aware of activities of other agents and some other different events inside the system using artifacts. These artifacts provide information about annotations made during the execution of a MAS. Therefore, Artifacts for Cognitive Stigmergy are considered as informative artifacts.

Finally, there are some basic artifacts, like a counter or a database (RP09), that are also considered to be informative artifacts.

We state that ORA4MAS is very similar to our proposal since both proposals can be considered as complementary. Both of them are focused on regulating and controlling the behavior of an OCMAS but our proposal has a more general point of view than ORA4MAS, which focuses on open normative multi-agent systems. Due to this normative approach, authors of ORA4MAS pay attention to mechanisms to instrument norms. They use two types of norm instrumentation mechanisms: regimentation and enforcement (GAD07). Regimentation is a mechanism that prevents agents to perform an action that violates a norm. This operation can be also carried out by a coercive artifact, which removes an action that can violate a norm from the action's space of an agent. On the other hand, ORA4MAS also applies enforcement, a reactive mechanism executed after the violation of a norm. Then, the artifact selects the sanction or reward that should be applied. An enforcement mechanism can be implemented by using an incentive artifact that is able to modify the reward system of a MAS.

As shown in this section, features of existing artifacts can be modeled following the proposed formalization of artifacts for organizational mechanisms. In most cases, current operations offered by existing artifacts make them to be easily modeled as informative artifacts. Moreover, the proposed formalization can also be useful to extend operations of current artifacts so as to apply incentive and/or coercive mechanisms into the environment. Also, an artifact can be also modeled as the combination of different types of artifacts for organizational mechanisms.

5.6.2 Environment definitions from other Organizational Modeling Languages

As explained in this chapter, we have included artifacts in an OML to facilitate the designer the task of defining, designing, and implementing mechanisms that regulate the environment. In this section, we will analyze how other OMLs model the environment, and we will compare them with VOM, and the extension presented in this chapter. More specifically, this section will focus on the following proposals: GAIA (ZJW03), OMACS (DeL09), SODA (MDO08), CArtAgO (RVO06) and MEnSA (ABC$^+$08).

GAIA (ZJW03) is the first OML that explicitly took into account social concepts. The environment of a MAS designed by GAIA is defined by means of the resources that roles have available. Each resource has a set of associated actions that is able to execute. The portion of the environment that agents can sense and effect is determined by their specific role and current status. The previous version of VOM was provided with the resource concept taken from Gaia, but in this new proposed version resources and their properties are embedded into the concept of artifact, thus facilitating the task of managing and using them.

The **OMACS** (DeL09) metamodel provides suitable mechanisms that allow the system to reorganize itself at runtime. OMACS domain model consists on a set of *environmental objects* that describe the elements of the system and a set of *properties* that specify the principles and processes that govern the environment. The main lack of this metamodel when modeling the environment is that it is unable to establish a particular location for each element of the environment just like the new Environment Dimension of VOM makes by using workspaces.

Artifacts and workspaces were firstly introduced in an OML by the **SODA** metamodel (MDO08), whose most recent version is compliant with the A&A (Agents & Artifacts) conceptual framework. The environment is thus modeled and structured by means of artifacts and workspaces.

CArtAgO (RVO06) is a framework built following the principles of the A&A conceptual framework. Its metamodel is composed of three main entities: (i) *agent bodies*, which make it possible to situate agents inside the environment; (ii) *artifacts* as the basic building blocks for structuring the environment; and (iii) *workspaces* aimed at defining the topology of the environment. This metamodel can be considered as a 'light' version of the SODA metamodel, but focusing on implementation details. Similarly to SODA, this metamodel enables describing the components of the environment but not how to regulate them.

MEnSA metamodel (ABC$^+$08) integrates fragments from well-known metamodels: Gaia, PASSI (Cos05), SODA and Tropos (BGG$^+$04). It is divided into requirements, design and implementation steps. In the *requirements* and *design* steps the topology is defined through *workspaces* that build the environment, in which *artifacts* are located (like in SODA). Artifacts can also be grouped by means of compositions.

Neither of these OMLs provide mechanisms nor facilities for regulating the environment. We could have selected any of them as a basis for our extension, specially those ones that employ the artifact concept; but we decided to extend VOM since we have a great knowledge of this OML and it gives support to GORMAS (ABJ11) methodology, which provides a set of guidelines that helps designers to model organizations and employs a software engineering method combined with an engineering process for designing human organizations based on the Organizational Theory (MLP98).

5.7 Conclusions

Organizational Mechanisms are aimed to improve coordination between agents in a MAS, trying to change this coordination from a micro perspective (i.e., the perspective of individual agents), providing useful information to the agents (by means of informative mechanisms); and a macro perspective (i.e., the perspective of the whole MAS), by modifying either the consequences of the actions of the agents (by using incentive mechanisms) or the capability functions of the agents (by means of coercive mechanisms).

In this chapter, these mechanisms have been modeled as artifacts to facilitate developers to better deploy and implement them, as well as adding functionality in MAS environments. Three types of Artifacts for Organizational Mechanisms have been defined: (i) *Informative Artifacts*, which provide information to an agent based on the internal state of this agent and the partial view of the environment that the artifact has; (ii) *Incentive Artifacts*, which modify the global behavior of the system by changing the incentive system of the MAS; and (iii) *Coercive Artifacts*, which update the action space of an agent. All these artifacts make use of the environment of a MAS, so they can explode all knowledge they have about entities populating the system.

Informative artifacts are passive entities used by agents to provide them with some information in order to help them in their deliberative processes. Incentive artifacts are introduced with the aim of modifying the consequences of actions such that agents have incentives (sanctions and rewards) to perform a particular action. Finally, coercive artifacts are aimed to produce changes in the environment of the system by producing changes in the agents' capability functions, adding or removing actions to the agents' action space.

Additionally, these artifacts for Organizational Mechanisms have been integrated inside an Organizational Modeling Language, so as to provide designers with a modeling tool for representing the environment. In this way, the Environment Dimension model of the Virtual Organization Model (AJB09) has been extended with the informative, incentive and coercive artifacts.

With the addition of the Artifacts for Organizational Mechanisms, the resulting extended metamodel becomes a useful coordination tool for an OCMAS since these artifacts introduce regulations into the environment that will positively affect the global utility of an OCMAS. Both artifacts and organizational mechanisms provide features that can help agents to coordinate themselves and organize a system without the need of other methods and tools. Additionally, using the concept of workspaces (also included in the metamodel), it is possible to physically define the environment, thus giving an absolute location to each entity populating this environment.

Since these artifacts present properties that make them suitable for representing organizational knowledge (and to promote coordination inside an OCMAS) as well as to define the environment, they are a tool that will help designers and developers of OCMAS with the definition and implementation of the forces that drive organizational change, which are presented in the next chapter. For example, mechanisms able for detecting the forces (and others to apply a solution to this force) may be encapsulating into artifacts for organizational mechanisms.

Relations between Artifacts and Organizational Mechanisms				
Artifact	**Org. Mechanisms**	**Observable properties**	**Operations**	**Link operations**
Coordination Artifacts				
Follow Me (ORV$^+$04)	Informative	`act`(a)	`do`(a), `get`	-
Argumentation Artifact				
Argument Acceptance Artifact (OMO08)	Informative	`Social Acceptability`, `Social Behavior`	`Write Arguments`	-
Artifacts for Cognitive Stigmergy				
Dashboard (ROV$^+$07)	Informative	`Annotations`	`takeAnnotations`, `manageAnnotations`	`traceOperations`
Log (ROV$^+$07)	Informative	`Events`	`inspectEvents`, `orderEvents`	`traceOperations`
Diary (ROV$^+$07)	Informative	`Annotations`	`makeAnnotation`	-
Note-board (ROV$^+$07)	Informative	`Annotations`	`makeAgentAnnotation`	`makeArtifactAnnotation`
Organizational Artifacts				
OrgBoard (HBKR10)	Informative, incentive	`OrgSpecification`, `GroupBoard`, `SchemeBoard`, `NormativeBoard`	`getOrgAgents`, `getMemberAgents`	`registerOrgArt`
GroupBoard (HBKR10)	Informative, incentive, coercive	`OrgBoard`, `SchemeBoards`, `Type`, `PlayableRoles`, `PlayersOfRole`	`adoptRole`, `leaveRole`	`addScheme`, `removeScheme`, `isMember`
SchemeBoard (HBKR10)	Incentive	`OrgBoard`, `GoalsState`, `NormativeBoard`, `RespGroupBoards`, `Type`, `PlayableMissions`, `PlayersOfMissions`	`commitMission`, `leaveMission`, `setGoalAchieved`	-
NormativeBoard (HBKR10)	Informative, incentive	`OrgBoard`, `NormStatus`	-	`updateAgentStatus`
Role Evolution Coordination Artifact (HBO10)	Informative	`Taxonomy`	`getBestRolesForInteract`, `getAgentsForRoles`, `getRolesForAgent`, `communicateTrust`	-
Reputation Artifact				
Alice's MSc course (HBV10)	Informative	`Evaluation`	-	`informAboutOrganization`
Basic Artifacts				
Counter (RP09)	Informative	`count`	`inc`, `reset`	-
Database (RP09)	Informative	`n_records`, `table_names`	`createTable`, `addRecord`, `query`	-

Table 5.2: Examples of artifacts grouped by their type

6

Case of study: Management of activities in a smart building

In this chapter we will depict how the templates for detecting and solving the forces that drive organizational change (see Chapter 4) are used in a case of study. Additionally, it will be depicted how to define Artifacts for Organizational Mechanisms to allocate organizational knowledge and also to structure the environment of the organization, as it has been presented in Chapter 5. This case of study is related to the management of activities assigned to the different rooms of a smart virtual building in a university.[1]

The building case of study has been chosen here because it presents a scenario where its features make possible that a varied set of forces may appear during the organizational life. Moreover, since the organization is located in an environment that influences its behavior, not only internal, but also external forces may affect

[1]This case of study is inspired in the work proposed by Sorici *et al.* (SBPS11).

the building. Therefore, reorganization would be carried out as the solution of a bad organizational performance. Its solutions will imply to make changes in different organizational elements, such as roles, organizational units, services, etc. so a variety of solutions is also shown in this chapter. This case of study is also interesting because it presents an example that comes from the human world and it is translated into the agent perspective, depicting that OCMAS are useful to make simulations of real-world organizations. However, the forces are also able to be triggered in a domain originally defined for computational agents. Another reason for choosing this case of study is that it has already been used as a testbed for reorganization, so its reliability has been proven (SBPS11). All these reasons make this case of study appropriate for exemplifying the proposal of this thesis.

6.1 Description of the case of study

This section describes the case of study, including the description of the scenario by means of a graphical notation, and in a formal way using VOF. Additionally, there are described the costs and thresholds used in this case of study.

6.1.1 Description of the scenario

The building of this case of study is located in a university. It has a set of rooms, where each room has assigned a type of activity. The three types of activities that a room can carry out are: teaching, meeting, and brainstorming. Each room has also a room manager, who is in charge of managing the room, checking its equipment, etc. The activity of a room is assigned by means of the role the room manager is playing (e.g., if the room manager is playing a teaching manager role, the room will be able to host teaching activities), and the role of a room manager may be switched during the runtime of the organization. There is a set of client agents (e.g., teachers) that send their requests for a specific type of activity, and the scheduler of the building will be responsible of allocating the requests among the rooms. The scheduler is also responsible of modifying the roles of the room managers.

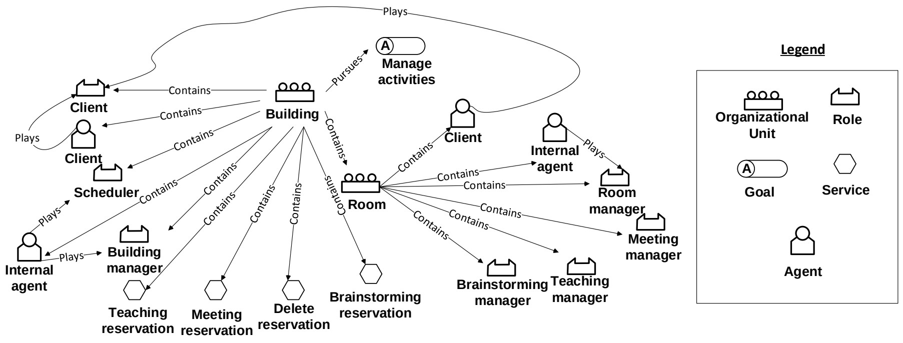

Figure 6.1: Structural dimension of the building represented with the GORMAS notation.

Figure 6.1 represents the organizational model issued from the analysis phase. This model is based on the graphical notation used by the GORMAS methodology (ABJ11) to describe OCMAS. As it can be seen in this figure, the organization is composed of a **Building** organizational unit, which contains **Room** units, as many as needed. Each Room unit represents the way agents and services governing the room will be organized. It contains an Internal agent that plays the **Room Manager** role and one of the following roles: **Teaching Manager**, **Meeting Manager**, or **Brainstorming Manager**. At the system initialization it only plays the Room Manager role, which is in charge of managing utilities, equipment, and other tasks related to the room management. With the objective of specifying the activity to be carried out in a specific Room at a specific time, the Internal Agent also plays one of any of the other three roles, which are exclusive between them, since a Room can only develop one type of activity at the same time. In each Room, there can be **Client** agents, in charge of requesting activities. The type of activities of each Room is changed dynamically depending on the requests issued by the Client agents. All Rooms are normally equipped to be able to develop any of the three types of activities at any time.

Additionally, the Building unit also contains the role **Client** (played by the Client agents), and the roles **Scheduler** and **Building Manager**, played by Internal Agents. The Building Manager specifies the type of the activities to be carried out in each Room by assigning a specific role (Brainstorming, Meeting, or Teaching Manager) to each Internal Agent playing the Room Manager role. The internal agents populating the Building unit are in charge of achieving the **Manage Activities** goal to assure a correct organizational performance. Finally, the Building Unit is composed of four services: one for deleting existing reservations and three services for reservation: *Teaching Reservation*, *Meeting Reservation*, and *Brainstorming Reservation.*

The scheduling of activities that are carried out in the Rooms of the Building are controlled by the **Scheduler**. The Clients of the Building (external agents) send their petitions (that include the type of room, the day, the start time and the duration of the activity expressed in number of hours) to the Scheduler, who is responsible of assigning a specific activity to a room at the required time. In this application, the organization may be subject to many sources of change. For example, an important decrease in the number of requests received by the organization or in the number of clients populating the organization are two of these sources.

In the next section, the organization is described by means of the Virtual Organization Formalization (VOF).

6.1.2 Description using VOF

Using the VOF formal representation, the building is defined as a tuple $building(t) = \langle OS(building,t), OE(building,t), \phi(building,t)\rangle$, where:

The organizational structure is defined as: $OS(building,t) = \langle SD(building,t), FD(building,t), ED(building,t), ND(building,t)\rangle$.

The structural dimension is defined as: $SD(building,t) = \langle R, Relations\rangle$, where:

- $R = \{Client, BuildingManager, RoomManager, TeachingManager, MeetingManager, BrainstormingManager, Scheduler\}$
- $Relations = \{mon(BuildingManager, RoomManager), mon(BuildingManager, Scheduler), inf(Client, Scheduler), roleHier(RoomManager, TeachingManager), roleHier(RoomManager, MeetingManager), roleHier(RoomManager, BrainstormingManager)\}$

The building manager monitors the activities developed by the room manager and the scheduler (because agents playing the roles of room manager and scheduler are subordinates of the building manager, which is in charge of the organization). The client and scheduler roles have an information relation between them because they have to exchange messages regarding the request of an activity. Finally, the roles teaching manager, meeting manager, and brainstorming manager are specializations of the room manager role, because the specialized roles have all the features of the room manager, plus the specific required features for each type of room. For example, the teaching manager has the capability of working with the projector.

The functional dimension is defined as: $FD = \langle G, S, GoalDependency\rangle$, where:

- $G = \{ManageActivities\}$
- $S = \{TeachingReservation, MeetingReservation,$
  $BrainstormingReservation,\ DeleteReservation\}$
- $GoalDependency = \emptyset$

The goal of the building, as an organizational unit, is to manage the activities being placed there. In order to do so, it offers several services to make a teaching reservation, a meeting reservation, and a brainstorming reservation. Additionally, it offers the service to delete a reservation that already exists.

As an example, the teaching reservation service is defined[1] as $TeachingReservation = \langle ProfileTeachRes, ProcessTeachRes, PerfTeachRes\rangle$, where:

- $ProfileTeachRes = \langle Client, Scheduler, \{Date, time\},$
  $ActivityConfirmation, PrecTeachRes, \emptyset\rangle$ where:
  - $PrecTeachRes = \exists Room \in OE(building, t)$
- $ProcessTeachRes = \langle StepTeachRes,$ "$KQML$"[2]$\rangle$, where:
  - $StepTeachRes = \{CheckSchedule.ConfirmReservation\}$ is the set of tasks that are carried out during the execution of this service.
- $PerfTeachRes = \langle 0.9, 800, 350\rangle$, are the values of quality, cost, and benefit of the service, respectively.

This service receives as input the desired date and time for the activity, and returns as output the confirmation whether the activity is carried out or not. As

[1]The service definition is found in Chapter 3, Definition 7.

[2]KQML is the agent communication language used in this service.

preconditions (*PrecTeachRes*), it is necessary that a room exists in the building to request the service. Moreover, the process followed by the service consists of two tasks. First, the schedule is checked so as to look for a room with a gap in its schedule for allocating the activity. Second, the activity is confirmed or not depending on the existence of an open space for the activity. Regarding the performance values for this service, the quality of the teaching reservation service is set to be a high value, 90%, because the teaching activities are considered to be crucial inside the university. Finally, the building management has calculated the cost and benefits produced when this service is executed. The cost of the service (i.e., the cost of having an agent playing the provider role plus the cost of executing the tasks of the service) is set to 800 units, as depicted in Table 6.1. Let us suppose that the net benefit of correctly allocating an activity into a room is 350 units.

The other services (Meeting reservation and Brainstorming reservation) are defined in a similar way, with the only difference of the quality of service, because the organizational management intends to assure different qualities depending on the importance of the service. In the case of a Meeting reservation, this value is set to 60%, and in the case of the Brainstorming reservation it is set to 50%, because these activities are not as critical as the teaching activities.[1] However, it is important to assure that an appropriate number of requests for these services is fulfilled to guarantee a good organizational performance.

At the start of a week, the type of activities the rooms are able to carry out can be randomly distributed, or they can follow the distribution of the week before. In this example, they are randomly distributed. The cost of the services is the same

[1]Such QoS means that the building manager will only accept a maximum of 10%, 40%, and 50% of failures for teaching, meeting, and brainstorming requests for activities, respectively.

for the three of them because they require the same resources, agents, and carry out the same tasks.

The environment dimension is defined as $ED(building, t) = \langle WSP, WSO, Composition\rangle$, where:

- $WSP = \{CSSch, AIDep, SEDep\}$
- $WSO = \{CSSch\}$
- $Composition = \{\{AIDep, SEDep\}\}$

where $CSSch$ is the School of Computer Science, $AIDep$ is the Department of Artificial Intelligence, and $SEDep$ is the Department of Software Engineering, the three worskpaces that the building is able to perceive since they are related to the same topic. The building is located in the workspace of the School of Computer Science, and the workspaces of the departments intersect because they share the same physical location. A visual description of the environment dimension can be found in Figure 6.2.

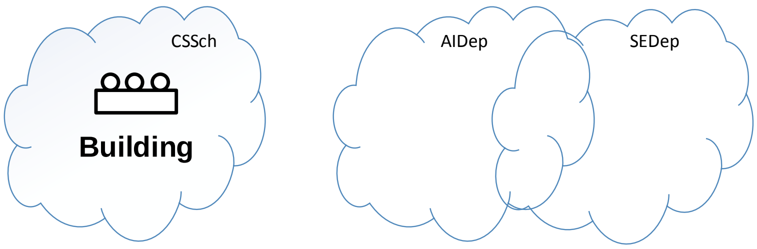

Figure 6.2: *Environment dimension of the case of study*

The normative dimension is defined as $ND(building, t) = \{ClosingTime, ReservationBeforeActivity\}$. The first norm states that the building has to close at 21:00. The second norm states that it is necessary to make a reservation prior to carry an activity out.

The norm $ClosingTime$ is defined[1] as $ClosingTime = \langle \mathcal{O}, CloseAt21, BuildingManager \rangle$, meaning that the agent playing the $BuildingManager$ role has the obligation of executing the task $CloseAt21$ that means that the building closes at 21:00. The norm $ReservationBeforeActivity$ is defined as $ReservationBeforeActivity = \langle \mathcal{F}, ActivityWithoutReservation, Client \rangle$, that states that it is forbidden for agents playing the $Client$ role to carry out the task $ActivityWithoutReservation$. This is, to start an activity without making a reservation using the appropriate service.

The organizational entity is defined as:

$$OE(building, t) = \{room_1, room_2, room_3, c_1, ..., c_{30}, bm_1, Scheduler_1, rm_1, rm_2, rm_3\}$$

The building is populated at a time t by three rooms (organizational units) with its corresponding room manager agents (rm_i), as well as the building manager (bm_1), the scheduler, and thirty client agents (c_i).

Finally, the organizational instantiation is defined as $\phi(building, t) = \langle Plays, Population, CostPlay, Budget \rangle$, where:

- $Plays = \{\langle c_1, Client \rangle, ..., \langle c_{30}, Client \rangle, \langle bm_1, BuildingManager \rangle, \langle Scheduler_1, Scheduler \rangle, \langle rm_1, RoomManager \rangle, ..., \langle rm_3, RoomManager \rangle\}$
- $Population = \{\langle CSSch, \{c_1, ..., c_{10}, bm_1, Scheduler_1, rm_1, rm_2, rm_3\} \rangle, \langle AIDep, \{c_{11}, ..., c_{20}\} \rangle, \langle SEDep, \{c_{21}, ..., c_{30}\} \rangle\}$
- $CostPlay = \langle \{Client, 100\}, \{BuildingManager, 500\}, \{Scheduler, 300\}, \{RoomManager, 200\} \rangle$

[1]The description of the definition of a norm can be found in Chapter 3, Definition 11.

- $Budget = 7000$

In this example, each agent plays its related role, so then client agents play client role, room manager agents play room manager role, and so on. Agents playing building manager, scheduler, and room manager roles, as well as ten clients, are located in the same workspace as the building, i.e. $CSSch$. Other ten clients are located in the $AIDep$ workspaces, and ten clients more are located in the $SEDep$ workspace. Moreover, the costs generated by agents playing each role and the budget of the organization are defined at the design time by the organizational designer. In this example, we have decided that being a Building Manager is much more costly than being a client because a building manager requires more knowledge and capabilities to develop its functionalities than a client agent. Having 7000 units as a budget allows the organization to have the required agents (a building manager, a scheduler, and a room manager for each room), as well as to host an acceptable amount of client, and have the services properly running.

6.1.3 Description of costs and thresholds

This section presents the costs of the organization (see Table 6.1) and also the thresholds (see Table 6.2), defined by organizational designers, used in this case of study. Notice that since this is a case of study to depict the features of the forces, the exact values presented here are not significant, but it is important to note the relations and proportions between these values. As an example, the cost of getting the room manager role is the 20% of the cost of playing it. Also, playing the role $BuildingManager$ is the most costly because the mental state and the functions that an agent playing this role has are more complex than the ones from other roles, thus generating higher computational cost, required storage space, etc.

Organizational costs		
Action	**Value**	**Comments**
$org_i.Agents.Add(a_i)$	50	This value is independent from the type of agent to add.
$a_i.Roles.Get(BuildingManager)$	100	20% of the cost of playing building manager role.
$a_i.Roles.Get(RoomManager)$	60	20% of the cost of playing this role.
$a_i.Roles.Get(TeachingManager)$	60	20% of the cost of playing this role.
$a_i.Roles.Get(MeetingManager)$	60	20% of the cost of playing this role.
$a_i.Roles.Get(BrainstormingManager)$	60	20% of the cost of playing this role.
$a_i.Roles.Leave(TeachingManager)$	30	10% of the cost of playing this role.
$a_i.Roles.Leave(MeetingManager)$	30	10% of the cost of playing this role.
$a_i.Roles.Leave(BrainstormingManager)$	30	10% of the cost of playing this role.
$org_i.OUs.Add(ou_i)$	400	8 times more costly that adding an agent. This value is independent from the type of OU to be added.
$building.Services.Add(s_i)$	160	20% of the cost of a reservation service.
$building.Services.Delete(s_i)$	80	10% of the cost of a reservation service.
$building.Norms.Delete(n_i)$	30	Its cost is related to the changes in the knowledge of the agents affected by this norm.
$building.Agents.Sanction(a_i)$	Variable	Depends on the specific sanction.
$TeachingReservation.Cost$	800	Cost generated by the agents participating in the service, plus the cost generated by the tasks of the service.
$MeetingReservation.Cost$	800	Cost generated by the agents participating in the service, plus the cost generated by the tasks of the service.
$BrainstormingReservation.Cost$	800	Cost generated by the agents participating in the service, plus the cost generated by the tasks of the service.
$cost(building)$	Variable	$cost(building.Services) + cost(building.Roles)$
Costs defined by the VOF definition		
Cost	**Value**	**Comments**
$costPlayRole(Client, building, t)$	100	Least costly role to play because it has the simplest functionality.
$costPlayRole(BuildingManager, building, t)$	500	Most costly role to play because it has the most complex functionality.
$costPlayRole(RoomManager, building, t)$	200	More complex that a client, but less than a building manager.
$costPlayRole(Scheduler, building, t)$	300	More complex functionalities than a room manager, but less than a building manager.

Table 6.1: Organizational costs

Regarding the thresholds, they are defined by organizational designers following a criteria that guarantees a correct detection of the factors they are related to. For example, the threshold $th_{MaxPopulation}$ is set to 40 because the maximum number of spots in the room with the initial room structure is 30. Therefore, having 40 or more client agents populating the organization would increase the number of non-fulfilled requests in a way that the organizational management considers that is not acceptable. The same stands for the rest of the thresholds. As an example, th_{minReq} is set as 8, since the initial number of clients is 30, and each one can send a request, so in comparison 8 is a low number of received requests.

Organizational threshold		
Threshold	**Value**	**Comments**
$th_{MaxPopulation}$	40	Maximum number of clients supported by the building. More clients would make the performance of the organization to decrease.
$th_{MaxBudget}$	7500	Maximum budget allowed in the organization so as to keep the current structure without reducing the performance.
$th_{allowedComp}$	3	Maximum number of organizations allowed in the environment providing similar services without considering a competition situation.
th_{minReq}	8	Minimum number of requests that a service has to receive so as to consider it useful for the organization.

Table 6.2: Organizational thresholds

In the next section, it is described how an implementation of the defined organization using Agents & Artifacts conceptual framework has been carried out, following the proposal detailed in Chapter 5.

6.2 Implementation based on Agents & Artifacts

In the previous section, the organization (Figure 6.1) has been defined using VOF, which describes the organization in a formal way. For implementing the organization, it is necessary to be provided with a tool to represent organizational knowledge at the implementation step. In this case, we use the Artifacts for Organizational Mechanisms (presented in Chapter 5) because they represent a proper way to introduce organizational concepts into the implementation of multi-agent systems. Therefore, the artifacts defined in this chapter comply with the given definition of the Artifacts for Organizational Mechanisms.

Agents and artifacts are implemented using the Jason and CArtAgO components of JaCaMo as follows:

- **Agents** are: *Building manager* (responsible of selecting the solution), *client* (generates a request for activity), *room manager* (manages the room and the role it plays is the activity carried out inside the room), and *scheduler* (distributes the requests around the different rooms and calculates the failures of the requests). Each type of agent has a different set of skills and capabilities, related to the roles defined at the design time.

- **Artifacts**, implemented as CArtAgO classes, provide functionalities to the agents. Four artifacts are defined: *monitor*, *room*, *date generator*, and *normative* artifacts. The *monitor* and *room* artifacts store the number of requests and occupancies. The *room* artifact is controlled by the *room manager* agent. The *monitor* artifact (mentioned in Table 4.6 of Chapter 4) stores useful information about different aspects of the organization that may trigger a force and agents can check to take a decision about a reorganization. To carry out the experimentations, a *date generator* artifact generates random requests of activities in the system. Finally, the *normative* artifact contains the norms of the organization, and it can be checked by agents.

Following, a brief description of the artifacts is given:

- The **room** artifact (Figure 6.3, left) is defined as:

$$Ar_{room} = \langle St, \{ActivityType, Capacity, Schedule, RoomManager,$$

$$Equipment\}, \{modifySchedule, modifyEquipment\}, \emptyset, CSSch\rangle$$

It is an informative artifact that stores and returns information about the room, and its associated equipment and room manager. The internal state St of this artifact contains the information of the agents that may execute the operations (e.g., the $modifySchedule$ operation is able to be accessed by an agent playing the role $Scheduler$). The properties of the artifact are a set of values that describe the main features of the room (i.e., the type of activities being carried out, the capacity of the room, its schedule, who is the room manager, and which is the equipment of the room). The operations available are employed to modify the schedule and the equipment of the room. The workspace $CSSch$ represents the place where this artifact is located. Finally, this artifact has no link operations ($\emptyset$), so it can be used by agents, and not by other artifacts.

- The **monitor** artifact (Figure 6.3, right) is defined as:

$$Ar_{monitor} = \langle St, \emptyset, \{getData, insertData\}, \{insertData\}, CSSch\rangle$$

where $insertData$ is the link operation that other artifacts use to introduce relevant data in the artifact, and $getData$ is an operation to get personalized data in terms of the type of information or the role of the agent which is requesting it. This artifact stores in its internal state St the different attributes that are required to state whether a factor of a force is triggered or not.

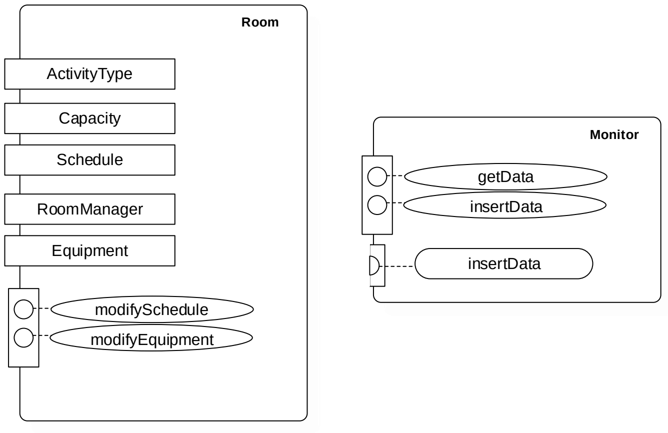

Figure 6.3: *Room and monitor artifacts*

- The **date generator** artifact (Figure 6.4, left) is defined as:

$$Ar_{dateGen} = \langle \emptyset, \emptyset, \{generateDate\}, \emptyset, CSSch \rangle$$

  It is a basic artifact whose only operation, *generateDate*, returns a random date. It is used by a client agent when it generates a random request for an activity to the organization. For the sake of simplicity and readability, here the artifact only carries out a simple operation, but in a more general situation this operation may be more complex. If the operation is executed by different agents (no matter whether is simple or complicated), it is more efficient to encapsulate it into an artifact.

- The **normative** artifact (Figure 6.4, right) contains the norms of the organization, and it is defined as:

$$Ar_{normative} = \langle \emptyset, \{Norms\}, \emptyset, \emptyset, CSSch \rangle$$

The property $Norms$ are the norms of the organization. Being a property of an artifact, $Norms$ norms can be checked by agents, and only gives information that is relevant and/or useful to the role they are playing. $CSSch$ is the workspace where the artifact is located.

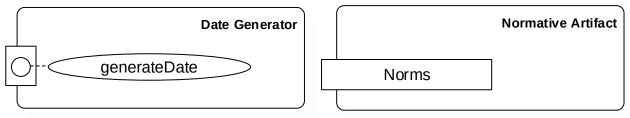

Figure 6.4: *Date generator and normative artifacts*

In a regular execution of the scenario, at the first execution cycle, the *Building manager* creates both the *Monitor* artifact and the *Date generator* artifact. Additionally, in this phase each of the *Room manager* agents randomly receives a type of role (teaching, meeting, and brainstorming), creates the room artifact that it manages, and sends this information to the *scheduler*.

In this case of study, the resources required for the execution of services are considered to be the rooms. Access to the rooms is managed by the reservation services. Therefore, we consider here that the access to a resource fails if the execution of a reservation service fails.

Then, on each execution cycle, each *Client* executes an operation of the *date generator* artifact to generate a random petition for an activity. Each petition for a room includes a specific day of the week (from Monday to Friday), with a specific start time (from 9am to 6pm, in 1-hour intervals, thus having the opportunity to start the activity in 10 different hours each day), and a length (of 1, 2, or 3 hours).

All clients send this information to the Scheduler, which tries to allocate all the petitions around the different rooms at the required times.

Using this case of study and its associated formal representation and implementation based on Agents & Artifacts, some forces are described in Section 6.3 to test the validity of the proposal of this thesis.

6.3 Forces analysis

In this section, some forces that drive organizational change are described for this case of study, thus depicting how to detect the forces by means of the factors, and then applying the specific solutions for the proposed scenario. The selected forces represent examples on both the external and internal forces that affect the organization in different ways. They have been selected here because they are the forces that are most likely to appear in the organization. The external forces presented here are: 'obtaining resources', 'market forces', and 'laws and regulations'. The internal forces that have been chosen are 'growth' and 'economical restrictions'. The following subsections depict how these forces are detected and solved in the building case of study.

6.3.1 *Obtaining resources* Force

As previously explained in Chapter 4, the obtaining resources force (see Table 6.3) is triggered when a service is not able to get the resources that are its preconditions. In this case of study, the resources required for the execution of reservation services are considered to be open spots in the schedule of the rooms. Therefore, we consider that a reservation service fails if it is not able to get a spot for an activity in one of the rooms of the building.

In a regular execution cycle, clients send requests to the *scheduler*. After all the requests have been processed, and once the *scheduler* decides whether they can be allocated into a room or not (failures are calculated using Equation 4.3.1.1 of Chapter 4), the *Monitor* artifact computes whether an adaptation is required or not, following the *force detection condition* of Table 6.4. After the computation, the *Monitor* sends a specific signal if an adaptation is required. According to the 'Obtaining resources' force, an adaptation is required when this expression holds:

$$Monitor.Failures(Res, t_1, t_2) > ((1 - Res.QoS) * Monitor.Requests(Res, t_1, t_2))$$

where $Res \in \{TeachingReservation, MeetingReservation,$ $BrainstormingReservation\}$ represents one of the services of the organization. Thus, the value of the quality of service ($Res.QoS$) specifies the number of requests that have to be fulfilled when executing the service.

Field	**Description**
Name	ObtainingResources
Description	A high number of requests cannot be allocated in the rooms of the organization.
Type	External
Factors	$FailedServiceCallsRate$
Force detection condition	$FailedServiceCallsRate.Condition = True$
Solutions	$ReachAgreement$, $ExtendOrganization$, $ChangeRoomActivity$
Force Solution Condition	$argmax_{x \in \{ReachAgreement, ExtendOrganization, ChangeRoomActivity\}} utility_x$

Table 6.3: *Obtaining resources* force for the case of study

Taking an action in the organization implies decreasing the number of requests that are not properly allocated, thus increasing the number of spots for the type of activities that are more requested. In the situation that a change is necessary because the number of failures when requesting one specific service is low, the

Factor	
Name	FailedServiceCallsRate
Description	If the failed service calls rate (i.e., requesting a spot for developing an activity) is higher than the allowed failure rate threshold, then this force is considered as acting.
Parameters	$Res \in \{TeachingReservation, MeetingReservation, BrainstormingReservation\}$ t_1, t_2
Condition	$Monitor.Failures(Res, t_1, t_2) > ((1 - Res.QoS) * Monitor.Requests(Res, t_1, t_2))$
Monitor	*Monitor* artifact

Table 6.4: *FailedServiceCallsRate* factor for the case of study

Building manager gets the data from the *Monitor* artifact (using the *getData* operation) to check the utility of the three possible solutions (*ReachAgreement*, *ExtendOrganization*, *ChangeRoomActivity*) so as to decide which solution to apply, or if it is better to not apply any solution. The following operation is carried out for calculating the utility:

$$utility_{sol} = benefit_{sol} - cost_{sol} \tag{6.1}$$

where $benefit_{sol}$ is the benefit of the solution when applying the solution sol and $cost_{sol}$ is the cost generated by the organization after applying the solution sol.

The chosen solution would be the one which maximizes the utility of the organization, because it would be the one with better tradeoff between the produced benefits and the generated cost.

As explained in Chapter 4, the first solution for 'Obtaining resources' force (see Table 6.5) consists on reaching an agreement with an agent that should be able to bring the required resources to the building. In this organization, the resources are the rooms that are used to allocate activities. Let us suppose that exists another building *Building*2 in the environment. Therefore, an agreement may be reached between the *Building Manager* bm_1 of the building of the case of study, and a

Building Manager bm_2 from a different building $Building2$ of the environment, located in the workspace $AIDep$ (see Figure 6.5). When reaching this agreement, the external building manager bm_2 has to be added to the organization, and gets the role *Building Manager*. This role will describe the duties and obligations of this agent inside the organization. Then, the set of actions to be followed for the first solution is:

$$building.Agents.Add(bm_2); bm_2.Roles.Get(BuildingManager)$$

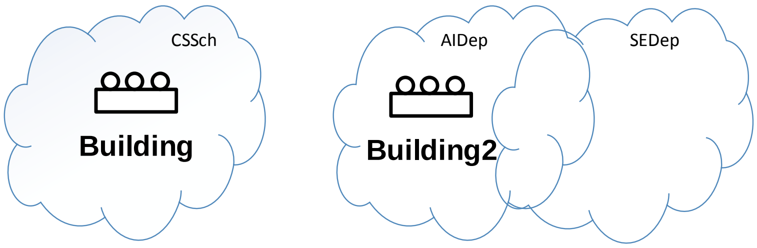

Figure 6.5: *Environment dimension when the obtaining resources force is active*

Let us suppose that we have the following costs (based on the organizational costs defined in Table 6.1):

- $cost(building.Agents.Add(bm_2)) = 50$
- $cost(bm_2.Roles.Get(BuildingManager)) = 100$

Therefore, the cost of this solution is: 50 + 100 = 150 units.

The 'ExtendOrganization' solution (as explained in Chapter 4) requires to extend the organization to get more resources (see Table 6.6). In this case, the organization is extended by means of more rooms, which are the resources of the

Field	**Description**
Name	ReachAgreement
Description	The building manager reaches an agreement with the building manager of another building to get new spots for activities.
Condition	FailedServiceCallsRate
Parameters	$bm_2 \in \mathcal{A}, BuildingManager \in \mathcal{R}$
Action	$building.Agents.Add(bm_2); bm_2.Roles.Get(BuildingManager)$
Cost	$cost(building.Agents.Add(bm_2)) + cost(bm_2.Roles.Get(BuildingManager))$
Executor	$BuildingManager$

Table 6.5: *ReachAgreement* solution for the case of study

services of the organization, so new OUs are added to the organization, and also the agents a_i that have to manage them. The following operations have to be carried out for each room $room_i$ to be added to the organization:

$$building.OUs.Add(room_i); room_i.Agents.Add(a_i); a_i.Roles.Get(RoomManager)$$

Field	**Description**
Name	ExtendOrganization
Description	The building is extended with new rooms that allow getting the required spots for activities.
Condition	FailedServiceCallsRate
Parameters	$NewRooms$
Action	$\forall room_i \in NewRooms : building.OUs.Add(room_i); room_i.Agents.Add(a_i); a_i.Roles.Get(RoomManager)$
Cost	$\lvert NewRooms\rvert * (cost(building.OUs.Add(room_i)) + cost(room_i.Agents.Add(a_i)) + cost(a_i.Roles.Get(RoomManager)))$
Executor	$BuildingManager$

Table 6.6: *ExtendOrganization* solution for the case of study

The costs of this solution are defined in this way (based on costs of Table 6.1):

- $cost(building.OUs.Add(room_i)) = 400$
- $cost(room_i.Agents.Add(a_i)) = 50$
- $cost(a_i.Roles.Get(RoomManager)) = 60$

Therefore, the cost of this solution when adding two rooms is: 2 * (400 + 50 + 60) = 1020 units.

The generic solutions presented at the template of the force (i.e., 'ReachAgreement' and 'ExtendOrganization') are intended to bring more resources to the organization. In this case of study, the required resources to properly execute the services are the slots in the schedule of the rooms. The first solution brings new slots by associating with another building that has its own rooms. The second solution extends the building, thus adding new rooms. However, the type of activity developed in a room may be dynamically changed by changing the role of a room manager. Therefore, in case there are rooms that may switch its type of activity (e.g., roles with no assigned activities), new rooms for a specific activity would be available. Table 6.7 describes this solution, named *ChangeRoomActivity*. In this solution, we have defined a set of tuples $RoomManagerList$, where each tuple contains the room manager agent of a room (rm_i) and the new role that it will play.

Field	**Description**
Name	ChangeRoomActivity
Description	The type of activity being carried out in a room is modified by changing the role of the room manager agent.
Condition	FailedServiceCallsRate
Parameters	$RoomManagerList = \{\langle rm_i, nr_i \rangle\}$, where $rm_i \in RoomManager \wedge$ $nr_i \in \{MeetingManager, TeachingManager, BrainstormingManager\}$
Actions	$\forall \langle rm_i, nr_i \rangle \in RoomManagerList : rm_i.Roles.Leave(Role); rm_i.Roles.Get(nr_i)$
Cost	$\sum_{\forall \langle rm_i, nr_i \rangle \in RoomManagerList} cost(rm_i.Roles.Get(nr_i)) + cost(rm_i.Roles.Leave(Role))$
Executor	$Scheduler$

Table 6.7: ChangeRoomActivity solution, design step

The costs associated to this solution are defined as indicated in Table 6.1:

- $cost(rm_i.Roles.Get(TeachingManager)) = 60$
- $cost(rm_i.Roles.Get(MeetingManager)) = 60$

- $cost(rm_i.Roles.Get(BrainstormingManager)) = 60$
- $cost(rm_i.Roles.Leave(TeachingManager)) = 30$
- $cost(rm_i.Roles.Leave(MeetingManager)) = 30$
- $cost(rm_i.Roles.Leave(BrainstormingManager)) = 30$

In this scenario, there is one room for teaching ($room_1$, managed by rm_1), one or meeting ($room_2$, managed by rm_2), and one for brainstorming ($room_3$, managed by rm_3). Let us suppose that more rooms for teaching are required, and the schedule of $room_3$ is empty. Therefore, the role of rm_3 is changed from $BrainstormingManager$ to $TeachingManager$. Therefore, the cost of this solution is:
$cost(rm_i.Roles.Leave(BrainstormingManager)) +$
$cost(rm_i.Roles.Get(TeachingManager)) = 30 + 60 = 90$.

It has been stated that the chosen solution is the one that maximizes the utility. With the three solutions the requests properly fulfilled would be the same thus their benefits will have the same value, so it is necessary to chose the solution that minimizes the cost, which in this case is $ChangeRoomActivity$.

6.3.2 *Growth* Force

The Growth internal force (see Table 6.8) may also appear in the organization. In the case that the number of clients of the organization increases, thus increasing the number of requests received by the services of the organization, or if the budget of an organization arises above a threshold (this increase in the budget normally provokes an increase in the number of services, thus attracting more clients, etc.), then the organization would require a change in its structure to handle this situation. The factors of the number of agents ($NumberAgents$, see Table 6.9) and high budget ($HighBudget$, see Table 6.10) are monitored by the Monitor artifact. If the number of agents populating the organization is higher than the maximum allowed number of agents in the organization ($th_{MaxPopulation}$), in this case defined as 40, or its budget is higher than a specific threshold ($th_{MaxBudget}$), defined as 7000 units in this situation, then a solution is required. Regarding the number of agents, it is a reactive factor. This is, the performance of the organization will be reduced in case the organization does not change but the number of clients is still high. In the case of high budget, it is a preventive factor. Having a budget above the threshold could suppose in the near future to increase the number and variety of services of the organization, thus attracting a higher number of clients. Therefore, it is a good solution to prepare the organizational structure to prevent any possible future damage. However, there is no direct relationship between the thresholds in this case of study, but in other scenarios the designers may establish one.

The $HighBudget$ factor is related to the $EconomicalRestrictions$ force in a way that, the higher the budget growths, the less probable is that $EconomicalRestrictions$ appears.

Let us suppose that in a specific moment, the number of clients raises from 30 to 42. Therefore, a change becomes necessary to properly handle the new number of clients.

In the solution of this force ($MakeHierarchy$, see Table 6.11), it is necessary to increase the size of the virtual building so as to distribute the load of requests between more rooms. It is decided by the management to add 2 rooms (since they are considered to be enough because of the number of spots for activities they provide), including their respective manager agents. However, the number of rooms would become relatively high for using a plain organization, so it is necessary to transform the structure of the building into a hierarchy to facilitate

Guideline for detecting a force	
Field	**Description**
Name	Growth
Description	The number of clients or budget of the building increases
Type	Internal
Factors	$NumberAgents$, $HighBudget$
Force detection condition	$NumberAgents.Condition = True \vee HighBudget.Condition = True$
Solutions	$MakeHierarchy$
Force solution condition	-

Table 6.8: *Growth* force for the case of study

Name	NumberAgents
Description	The number of client agents populating the building is higher than the maximum allowed population for correctly managing the organization
Parameters	$building$, $th_{MaxPopulation}$
Condition	$building.Population > th_{MaxPopulation}$
Monitor	*Monitor* artifact

Table 6.9: *Number of agents* factor for the case of study

Name	HighBudget
Description	If the budget of the building increases above a threshold, then a change is necessary.
Parameters	$building$, $th_{MaxBudget}$
Condition	$building.Budget > th_{MaxBudget}$
Monitor	*Monitor* artifact

Table 6.10: *High budget* factor for the case of study

its management. Therefore, a new organizational unit is required to place the new rooms inside it. This new unit will be a new 'wing' of this virtual building, added using the operation $building.OUs.Add(wing)$. It is also required to add a new intermediate manager role which has to be in charge of managing this unit ($wing.Agents.Add(wingManager)$). Therefore, the following operations for adding rooms are executed: $wing.OUs.Add(room_4)$ and $wing.OUs.Add(room_5)$, and also two agents are required: $room_4.Agents.Add(rm_4)$ and $room_5.Agents.Add(rm_5)$, which play the $RoomManager$ ($rm_4.Roles.Get(RoomManager)$, $rm_5.Roles.Get(RoomManager)$). They create new rooms, not in the main building, but in the created wing. The new state of the organization is represented in Figure 6.6. For the sake of simplicity and readability, the figure only represents the organizational units, as well as the agents that manage them.

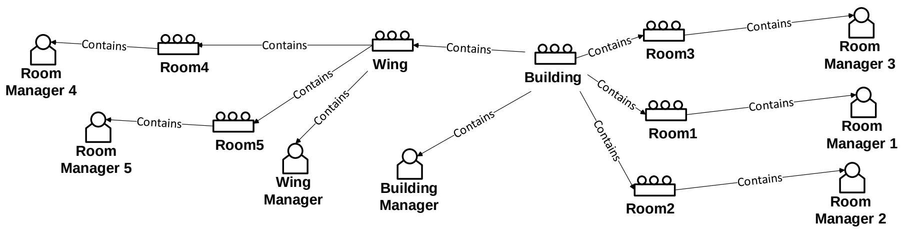

Figure 6.6: *Structure of the building after a wing with two rooms is added*

The cost of this solution is:

$$\begin{aligned}&cost(building.OUs.Add(wing)) + cost(wing.Agents.Add(wingManager)) +\\ &cost(wing.OUs.Add(room_4)) + cost(wing.OUs.Add(room_5)) +\\ &cost(room_4.Agents.Add(rm_4)) + cost(room_5.Agents.Add(rm_5)) +\\ &cost(rm_4.Roles.Get(RoomManager)) + cost(rm_5.Roles.Get(RoomManager)) =\\ &400 + 50 + 400 + 400 + 50 + 50 + 60 + 60 = 1470\end{aligned}$$

6.3.3 *Market* Forces

The *Market Forces* (see Table 6.12) describe the situation where the number of requests to one of the services is lower than the established threshold (see Table 6.13), so this service has to be removed from the set of services being offered by the organization in order to reduce the cost it generates. Not removing the service would not lead the building to a critical fail like it happens when other forces (such

Solution for preventing damage or taking advantage of a force	
Field	**Description**
Name	MakeHierarchy
Description	Make a hierarchy in the building, add organizational units and roles to improve the building management.
Factor	$NumberAgents \vee HighBudget$
Parameters	$building$, $NewRoles$
Action	$building.OUs.Add(wing); wing.Agents.Add(wingManager);$ $wing.OUs.Add(room_4); wing.OUs.Add(room_5);$ $room_4.Agents.Add(rm_4); room_5.Agents.Add(rm_5);$ $rm_4.Roles.Get(RoomManager); rm_5.Roles.Get(RoomManager)$
Cost	$cost(building.OUs.Add(wing)) + cost(wing.Agents.Add(wingManager))+$ $cost(wing.OUs.Add(room_4)) + cost(wing.OUs.Add(room_5))+$ $cost(room_4.Agents.Add(rm_4)) + cost(room_5.Agents.Add(rm_5))+$ $cost(rm_4.Roles.Get(RoomManager)) + cost(rm_5.Roles.Get(RoomManager))$
Executor	$BuildingManager$

Table 6.11: *Make a hierarchy* solution for the case of study

as ‘Obtaining resources’) are active. However, since the building has to stick to a budget, it is more convenient to carry this change out to improve the performance and utility of the organization.

The detection of the force is also carried out by the *Monitor* artifact, which tracks the number of requests to services. The threshold for the minimum number of requests to consider a service as useful for the organization of our case of study is defined here as $th_{minReq} = 8$. As previously explained, this threshold is defined taking into account that the initial number of clients of the building is 30 (the building receives 30 requests for activities), so 8 is a low number of requests to be received.

Let us suppose that the monitor artifact detects that the number of requests received by the $BrainstormingReservation$ service between t_1 and t_2 is 5 ($Monitor.Requests(BrainstormingReservation, t_1, t_2) = 5$). Therefore, it is necessary to deploy the solution of this force (see Table 6.14), which consists on the removal of the service that received a low number of requests. Therefore, the action to delete the $BrainstormingReservation$ service is carried out

Field	**Description**
Name	MarketForces
Description	One of the services of the building has a low number of requests.
Type	External
Factors	$ServiceRequests$
Force detection condition	$ServiceRequests.Condition = True$
Solutions	$DeleteService$
Force solution condition	-

Table 6.12: *Market forces* for the case of study

Factor	
Name	ServiceRequests
Description	If the number of requests for a service is lower than a specific value, then this force is acting.
Parameters	$building$, th_{minReq}, t_1, t_2, $s_i \in \{TeachingReservation, MeetingReservation, BrainstormingReservation\}$
Condition	$\exists s_i \in building.Services : Monitor.Requests(s_i, t_1, t_2) < th_{minReq}$
Monitor	*Monitor* artifact

Table 6.13: *ServiceRequests* factor for the case of study

(*building.Services.Delete*(*BrainstormingReservation*)). The cost of this solution is calculated as *cost*(*building.Services.Delete*(s_i)) = 80 units.

Field	**Description**
Name	DeleteService
Description	A service with a low number of requests is deleted from the building.
Condition	ServiceRequests
Parameters	$building$, $s_i \in \{TeachingReservation, MeetingReservation, BrainstormingReservation\}$
Action	$building.Services.Delete(s_i)$
Cost	$cost(building.Services.Delete(s_i))$
Executor	$BuildingManager$

Table 6.14: *DeleteService* solution for the case of study

6.3.4 *Laws and regulations* Force

The *laws and regulations* force (see Table 6.15) is related to the norms that are located at the environment of an organization. In this case of study, the building is located in a university. At the same time, the university is located in a society whose government promulgates norms that the individuals, groups, and institutions inside this society have to follow. Then, it is possible that the building promulgates an internal norm that contradicts a norm promulgated by the university or the government because they act over the same element in a contradictory way (see Table 6.16). Therefore, this conflicting norm of the building should be removed or modified because the norms promoted by the university or the government have a higher priority since these organizations are more important than the building inside the organization.

Field	**Description**
Name	LawsRegulations
Description	Norms coming from a superior organization (e.g., university, government, etc.) of the building are in conflict with its own norms.
Type	External
Factors	$ConflictingNorms$
Force detection condition	$ConflictingNorms.Condition = True$
Solutions	$ModifyNorms$
Force Solution Condition	-

Table 6.15: *Laws and regulations* force for the case of study

The norm $ClosingTime$ of the building obliges the building manager to carry out the operation $CloseAt21$ (states that the closing time of the building is 21:00). Let us suppose that the university also has the norm $ClosingTime'$, which is similar to $ClosingTime$, but in this case the operation that the building manager has to carry out is $CloseAt20$ (which forces to close all the building of the university at 20:00). As it can be checked, both norms are incompatible because it is impossible to close the building at two different times. Therefore $conflict(ClosingTime, ClosingTime') = True$.

Therefore, the factor is active in the organization. As stated in the solution of this force, the norm $ClosingTime$ has to be removed or modified from the

Factor	
Name	ConflictingNorms
Description	There are norms coming from the environment that are in conflict with the organizational goals.
Parameters	n_1, n_2, $building$, $university$
Condition	$\exists n_1 \in building.Norms, n_2 \in university.Norms, building \in university.OUs \wedge conflict(n_1, n_2) = True$
Monitor	$Monitor$ artifact

Table 6.16: *ConflictingNorms* factor for the case of study

building (see Table 6.17). In this situation, since norm $ClosingTime'$ also affects the building due to the fact that it is a norm from the university that affects the whole institution, norm $ClosingTime$ is removed to avoid further conflicts, by using the operation $building.Norms.Delete(ClosingTime)$, with an associated cost of 30 units.

The change to be made in the building is mandatory because it has to comply with the norms promoted by the university, which is a superior organization. The building does not have the possibility of not applying a solution for this force.

Field	**Description**
Name	ModifyNorms
Description	The norms of the building are modified to comply with the norms of the university.
Condition	$ConflictingNorms$
Parameters	$building$, $ClosingTime$
Action	$building.Norms.Delete(ClosingTime)$
Cost	$cost(building.Norms.Delete(ClosingTime))$
Executor	$BuildingManager$

Table 6.17: *ModifyNorms* solution for the case of study

6.3.5 *Economical restrictions* Force

The *economical restrictions* force (see Table 6.18) refers to the budget that the organization has available (in terms of money, computational costs, etc.). The resources available in an organization are not unlimited, so the organization has to stick to a budget (see Table 6.19). Therefore, this equation (previously presented in Chapter 4, Table 4.40) must hold to make sure that the organization sticks to its budget:

$$cost(building) < building.Budget \tag{6.2}$$

where $cost(building)$ is the cost of the organization, and the means to calculate it can be found in Chapter 4, Equation 4.13.

Since the budget of an organization is a key issue (because no budget means that no functionalities are offered), this force is important to be correctly detected and faced.

Field	**Description**
Name	EconomicalRestrictions
Description	The cost of the building is higher than its associated budget.
Type	Internal
Factors	$CostOrg$
Force detection condition	$CostOrg.Condition = True$
Solutions	$ModifyServices$
Force solution condition	-

Table 6.18: *EconomicalRestrictions* force for the case of study

Factor	
Name	CostOrg
Description	The cost of the building is higher than the budget it has assigned.
Parameters	$building$
Condition	$cost(building) > building.Budget$
Monitor	$Monitor$ artifact

Table 6.19: *CostOrg* factor for the case of study

If the *Monitor* artifact receives the information that this equation does not hold, then this force is active in the building. As stated in Chapter 4, the solution to this force is to change the definition of services (or just deleting them) to make them less resource-consuming, so being possible to adjust the budget of the organization (see Table 6.20). In this case, we are not interested on the deletion of services, because it would suppose to reduce the functionalities of the organization. Therefore, the operations to modify the services are:

$$building.Services.Delete(TeachingReservation);$$

$$building.Services.Add(TeachingReservation');$$

$$building.Services.Delete(MeetingReservation);$$

$$building.Services.Add(MeetingReservation');$$

$$building.Services.Delete(BrainstormingReservation);$$

$$building.Services.Add(BrainstormingReservation')$$

where $TeachingReservation'$, $MeetingReservation'$, and $BrainstormingReservation'$ are modified versions of the services. These modified versions have a lower cost than the previous versions, thus reducing the whole cost of the organization below the organizational budget:

$$(cost(building') < cost(building)) \wedge (cost(building') < building.Budget) \tag{6.3}$$

where $building'$ corresponds to the building after the modification of the services (i.e., it includes the modified versions of the services $TeachingReservation'$, $MeetingReservation'$, and $BrainstormingReservation'$).

The specific costs of playing roles have been given in the organizational definition (Section 6.1.2). Let us suppose that there are 33 agents playing client roles, 3 agents playing manager roles, an agent playing the building manager role, and an

Field	**Description**
Name	ReduceCost
Description	The cost generated by the organization is reduced by modifying its services.
Condition	$CostOrg$
Parameters	$building$, $s_i \in \{TeachingReservation, MeetingReservation, BrainstormingReservation\}$
Action	$\forall s_i \in org.Services : building.Services.delete(s_i); [building.Services.add(s'_i)]$
Cost	$\sum_{\forall s_i \in org_i.Services} cost(building.Services.delete(s_i)) + cost(building.Services.add(s'_i))$
Executor	$BuildingManager$

Table 6.20: *ReduceCost* solution for the case of study

agent playing the scheduler role, this results in a cost of 4700 units[1] for the agents playing roles. The cost of the services are:

- $cost(TeachingReservation) = 800$
- $cost(MeetingReservation) = 800$
- $cost(BrainstormingReservation) = 800$

Therefore, the cost of the organization is calculated as in Equation 4.13 from Chapter 4:

$$cost(building) = \sum_{\forall s_i \in building.Services} s_i.Cost + \sum_{\forall r_i \in building.Roles} building.Cost(r_i) = 2400 + 4700 = 7100 \tag{6.4}$$

The budget is 7000, so the organization has to change its services. A change is carried out in the three services, consisting on a modification of the way the information of the schedule of a room is stored, making it less expensive from the

[1] $33 * costPlayRole(Client, building, t) + 3 * costPlayRole(RoomManager, building, t) + costPlayRole(Scheduler, building, t) + costPlayRole(BuildingManager, building, t) = 33 * 100 + 3 * 200 + 300 + 500 = 4700$ units.

point of view of the computational cost. Let us suppose that this reduces the cost of each service by 100 units. Therefore, the cost of the organization after the change would be 6800, which is below the budget.

The cost of this solution is (taking into account the costs of Table 6.1):

$$\sum_{\forall s_i \in building.Services} cost(building.Services.delete(s_i)) +$$
$$cost(building.Services.add(s_i')) = 160 + 80 + 160 + 80 + 160 + 80 = 720$$

6.3.6 *Demographical features* Force

The building of our case of study is an open system, in which external agents coming from the environment of the building may join it and request rooms. During the time they stay in the organization, these agents play the 'Client' role, which has associated norms that they have to follow. This kind of agents may have a self-interested behavior that contradicts the norms of the organization. This makes the demographical features force to be active (see Table 6.21), and the building has to keep these agents under control by applying a sanction over them to prevent more self-interested behaviors.

Field	**Description**
Name	DemographicalFeatures
Description	Due to the openness of the organization, some agents could not comply with the organizational rules and regulations.
Type	External
Factors	$ViolatedNorms$
Force detection condition	$ViolatedNorms.Condition = True$
Solutions	$SanctionAgent$
Force solution condition	-

Table 6.21: Definition of the *Demographical features* force

The detection of this force in this scenario is carried out by the monitor artifact, which is able to detect any violations of a norm of the building. At a specific time, the monitor detects that the norm $ReservationBeforeActivity$ is violated because the agent c_1 (where $Client \in c_1.Roles$) has carried out an activity without making a prior reservation. Therefore, the force is active, and a solution has to be carried out (see Table 6.22).

Factor	
Name	ViolatedNorms
Description	A self-interested client agent violates a norm of the building.
Parameters	a_i, n_i, $building$ $Client \in a_i.Roles$
Condition	$\exists n_i \in building.Norms, \exists a_i \in building.Agents : Monitor.Violations(a_i, n_i)$
Monitor	*Monitor* artifact

Table 6.22: *ViolatedNorms* factor for the case of study

The solution to this force (Table 6.23) is to sanction the agent by using the action $building.Agents.Sanction(c_1)$. The severity of this sanction ranges depending on the severity of the violation. In this case, since it is the first time that the agent has violated the norm, only a warning is given to the agent. However, the next time that agent c_1 violates the norm, this may imply to be ejected from the organization. As stated in Chapter 4, the solution to this force may be considered more a regulation process rather than an adaptation one since there is no modification of any of the structural elements of the organization. The cost of the solution depends on the kind of sanction applied. A warning has a low cost, while ejecting the agent from the organization is more costly.

Field	**Description**
Name	SanctionAgent
Description	The client agent that is not following its assigned norms is sanctioned to avoid damage to the building.
Condition	$ViolatedNorms$
Parameters	$building$, a_i $Client \in a_i.Roles$
Action	$building.Agents.Sanction(a_i)$
Cost	$cost(building.Agents.Sanction(a_i))$
Executor	$BuildingManager$

Table 6.23: *SanctionAgent* solution for the case of study

6.3.7 *Competition* Force

The building in this case of study is not alone in the environment it populates. Since this building is inside a university, different buildings around the university may also offer their rooms for the development of activities of teaching, meeting, and brainstorming. Thus, since the buildings of this scenario are open systems, these other buildings would attract clients initially intended for the reference building. If the number of buildings offering the same or similar services is high (and this makes the organization to lose clients), then the 'Competition' force will be active (Table 6.24). Detecting this force is important because if a big number of clients is lost, then other forces may be active in the organization, such as 'Market forces' (in the case that the number of requests to a service is drastically reduced).

Field	**Description**
Name	Competition
Description	Buildings populating the university offer similar products and services representing competence for our building.
Type	External
Factors	$OrgDensity$
Force detection condition	$OrgDensity.Condition = True$
Solutions	$AssociateOrg$, $ModifyServices$
Force solution condition	$argmax_{x \in \{AssociateOrg, ModifyServices\}} utility_x$

Table 6.24: *Competition* force for the case of study

The factor to detect the force (*OrgDensity*, see Table 6.25) is active if the equation $competition(building, t_1, t_2)$ holds (this equation was defined in Chapter 4, Equation 4.5):

$$competition(building, t_1, t_2) = (|\bigcup_{building_i \in building.Environment} \{building_i\} : \exists s_i \in building_i.Services \wedge \exists s_j \in building.Services \wedge s_i.Products = s_j.Products| > th_{allowedComp}) \wedge building.Clients(t_1) > building.Clients(t_2) \wedge t_2 < t_1$$

The threshold $th_{allowedComp}$ has been defined by the organizational management as 3. More organizations providing similar services will suppose to lose a big number of clients. During a period of time between t_1 and t_2, it is detected that the number of buildings offering similar services is 4, and it is also checked that the number of clients is reduced from t_1 to t_2. Therefore, $competition(building, t_1, t_2) = True$.

Factor	
Name	OrgDensity
Description	Other buildings in the organizational environment are offering similar products and services, thus making the number of clients of our building to decrease.
Parameters	$building$, t_1, t_2, $th_{allowedComp}$
Condition	$competition(building, t_1, t_2) = True$
Monitor	*Monitor* artifact

Table 6.25: *OrgDensity* factor for the case of study

The first possible solution (Table 6.26) is to associate this building with another building, so both buildings would share their rooms (and obviously, the free spaces in their schedules) in a way that if a building receives a request and has no available spaces in its own rooms, the request would be automatically forwarded to the other building, in a process that is transparent to the client that made the request. In Chapter 4 it is defined that the solution implies sharing the services.

However, in this specific scenario the services for the two buildings are the same, so sharing the services implies sharing the resources consumed by the services. This means adding the rooms of one building into the organizational entity of the other building. In this case, associating the organization *building* with another $building_2$ where $\exists room_4, room_5, room_6, rm_4, rm_5, rm_6 \in OE(building_2, t)$ implies modifying the organizational entity of *building* by adding the room managers of $building_2$ ($building.Agents.Add(rm_4)$, $building.Agents.Add(rm_5)$, $building.Agents.Add(rm_6)$). Having these agents in the organization will imply to be able to access the rooms they manage to allocate activities, thus being not necessary to add the room in the building, which is a more costly action. The final shape of the organizational entity of *building* after the changes will be:

$$OE(building, t+1) = \langle room_1, room_2, room_3, c_1, c_2, c_3,$$

$$bm_1, Scheduler_1, rm_1, rm_2, rm_3, rm_4, rm_5, rm_6\rangle$$

Field	**Description**
Name	AssociateOrg
Description	The building associates with another building to improve its performance.
Condition	$OrgDensity$
Parameters	$building, building_2$
Action	$\forall rm_i \in building_2.Agents \wedge RoomManager \in rm_i.Roles : building.Agents.Add(rm_i)$
Cost	$\sum_{\forall rm_i \in building_2.Agents \wedge RoomManager \in rm_i.Roles} cost(building.Agents.Add(rm_i))$
Executor	$BuildingManager$

Table 6.26: *AssociateOrg* solution for the case of study

The second possible solution to the *Competition* force consists on modifying the organizational services (by changing the products they generate) to make them different from other buildings, and trying to attract more clients (see Table

6.27). As an example, the service *BrainstormingReservation* may be substituted by the service *ProjectionReservation* for the activities that require projecting images of a movie, a demo, etc. that is not available in any of the buildings of the university. This is carried out in the building by the actions $building.Services.Delete(BrainstormingReservation)$ and $building.Services.Add(ProjectionReservation)$.

Field	**Description**
Name	ModifyServices
Description	A service is modified in a way that it would fulfil a different set of goals.
Condition	$OrgDensity$
Parameters	s_i, s'_i, $building$ $\nexists g_i \in building.Goals : g_i \in building.Goals \wedge \exists g_j \in s'_i.Goals : g_j \in building.Goals$
Action	$building.Services.Delete(s_i); building.Services.Add(s'_i)$
Cost	$cost(building.Services.Delete(s_i)) + cost(building.Services.Add(s'_i))$
Executor	Building manager

Table 6.27: *ModifyServices* solution for the case of study

The chosen solution would be the one that maximizes the organizational utility. In this case, the utility of the first solution is calculated as:

$$utility_{AssociateOrg} = benefit_{AssociateOrg} - cost_{AssociateOrg} \tag{6.5}$$

where $benefit_{AssociateOrg}$ is the benefit (in terms of number of clients) of the organization. The $cost_{AssociateOrg}$ is the cost of adding the rooms in the organizational entity of the building. Since it is added only virtually, the room manager agent is added instead of the room organizational unit. At a specific time, let us suppose that there are 28 clients whose requests have been fulfilled, with a benefit of 100 units generated by each one, and the cost of adding a new room manager agent is 50, and 3 room manager agents have been added. Then, the utility is:

$$utility_{AssociateOrg} = (28 * 100) - (50 * 3) = 2650 \tag{6.6}$$

The utility of the second solution is calculated as:

$$utility_{ModifyServices} = benefit_{ModifyServices} -$$
$$(cost(building.Services.Delete(BrainstormingReservation)) +$$
$$cost(building.Services.Add(ProjectionReservation)))$$

where $benefit_{ModifyServices}$ is the number of clients of the organization when applying this solution times the units of benefit each client returns. Let us suppose that this solution makes that 28 clients would be satisfied in the organization, with a benefit of 100 units generated by each of them. The cost of deleting the brainstorming reservation service is 80, and the cost of adding the projection reservation service is 160. So, the utility of this solution is:

$$utility_{ModifyServices} = (28 * 100) - (160 + 80) = 2560 \tag{6.7}$$

Then, the solution of associating with another organization (*AssociateOrg*) is chosen by the building manager to be deployed.

6.3.8 *Integration and externalization processes* Force

If two buildings have not achieved the utilities that their respective managements expect, or there is a high number of buildings offering the same services in the university, a merger or an association is necessary (Table 6.28). This merger or association would increase the performance and utility of the current building since it would have the capability of appealing more clients. In this case of study, we analyze the merger and association with the *building* organization presented here, and $building_2$, another building that is located in the same university, with the same services, goals, norms, and roles, as *building*.

Field	**Description**
Name	Merging
Description	Two buildings join to improve their situation inside the university.
Type	Internal
Factors	$OrgUtilities$, $OrgDensity$
Force detection condition	$OrgUtilities.Condition = True \vee OrgDensity.Condition = True$
Solutions	$MergeOrgs$, $AssociateOrg$
Force solution condition	$argmax_{x \in \{MergeOrgs, AssociateOrg\}} utility_x$

Table 6.28: *Integration and externalization processes* force for the case of study

The first factor takes into account the utilities of both buildings (see Table 6.29). This utility is calculated as the mean of the utilities of the services of the building. On the one hand, let us suppose that the utility of the organization *building*, in a period of time from t_1 to t_2, is:

$$utility(building, t_1, t_2) = benefits(building) - cost(building) = \\ (30 * 100) - 6800 = -3800$$

because there are 30 clients producing a benefit of 100 units each, while the cost was calculated using the Equation 6.4 (in this case, the cost is 6800). On the other hand, let us suppose that the utility of the organization $building_2$, in the same period of time, is:

$$utility(building_2, t_1, t_2) = benefits(building_2) - cost(building_2) = \\ (40 * 100) - 6800 = -2800$$

Moreover, building managers estimate that the joint utility in the new building would be:

$$utility(mergedBuilding, t_1, t_2) = (70 * 100) - 6800 = 200$$

The joint building will fulfil the request of 70 clients (producing a 100 units of benefit each) with the cost of one (in this situation, the cost of *mergeBuilding* is the same as *building*, 6800, because both buildings have the same services, goals, and roles, so the cost of the joint building equals the cost of one). Therefore, since the joint building would have a higher utility, then a joining process will be deployed.

Factor	
Name	OrgUtilities
Description	The utility of the buildings by their own is lower than the utility in the case they were merged.
Parameters	$building$, $building_2$, t_1, t_2 $mergedBuilding$ where $mergedBuilding.MergeOrgs(building, building_2)$
Condition	$(utility(building, t_1, t_2) < utility(mergedBuilding, t_1, t_2)) \wedge (utility(building_2, t_1, t_2) < utility(mergedBuilding, t_1, t_2))$
Monitor	$Monitor$ artifact

Table 6.29: *OrgUtilities* factor for the case of study

The second factor to take into account is the organizational density, as it was taken into account in the *Competition* force (see Table 6.25). If this factor appears, it means that there is a high number of buildings offering similar products or services, so a merge would give birth to a stronger building within the university. However, in this example, since the first factor is active, it is not necessary to check the second factor to state that the force is active.

Since in this case of study is more interesting to have a bigger building to allocate more petitions instead of a production chain between buildings (as the second solution would provide), the taken solution for this force is to merge the two buildings whose utility is not as higher as expected or which have to face a dense environment (see Table 6.30). In this case, the two buildings present the same set of roles, services, goals, and norms. This facilitates the merging process ($mergedBuilding.MergeOrgs(building, building_2)$), and the new merged organization *mergedBuilding* will have the same elements as the standalone buildings. However, the environment is different, because the $building_2$ is located in $AIDep$, while *building* is located in $CSSch$. Then, *mergedBuilding* is located in both workspaces.

- $mergedBuilding.Services = \langle TeachingReservation, MeetingReservation, BrainstormingReservation, DeleteReservation\rangle$
- $mergedBuilding.Goals = \langle ManageActivities\rangle$
- $mergedBuilding.Roles = \{Client, BuildingManager, RoomManager, TeachingManager, MeetingManager, BrainstormingManager, Scheduler\}$
- $mergedBuilding.Norms = \langle CloseTime21, ReservationBeforeActivity\rangle$
- $mergedBuilding.Environment = \langle CSSch, AIDep\rangle$

Their organizational entities are different, with the following entities populating $building_2$:

$$OE(building_2, t) = \langle room_{21}, room_{22}, room_{23}, c_6, c_7, c_8, bm_2, Scheduler_2, rm_{21}, rm_{22}, rm_{23}\rangle$$

Room organizational units, and client and room manager agents will be located in the merged building. They will contribute to make the building stronger inside the university. However, there are required only one building manager and one scheduler per building. Therefore, the OE of the merged building is:

$$OE(mergedBuilding, t) = \langle room_1, room_2, room_3, room_{21}, room_{22}, room_{23}, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, bm_1, Scheduler_1, rm_1, rm_2, rm_3, rm_{21}, rm_{22}, rm_{23}\rangle$$

Notice that the solution of this force gives birth to a new organization. However, this new organization is not immune to the forces. Therefore, factors may be detected by the organization. For example, joining two organizations may merge their clients. Therefore, the number of clients arises, thus triggering the 'Growth' force. However, merging two organizations of the same environment makes the force 'Competition' less likely to appear because less organizations would populate the environment after the merge and also the merged organization would be stronger. Additionally, it is necessary to redefine the thresholds to detect this force. For example, if the merged organization is more likely to appeal a higher number of clients, then the threshold $th_{MaxPopulation}$ has to be increased.

Field	**Description**
Name	MergeUnits
Description	The two organizations merge to improve their utility.
Condition	$OrgUtilities \vee OrgDensity$
Parameters	$mergedBuilding$, $building$, $building_2$
Action	$mergedBuilding.MergeOrgs(building, building_2)$
Cost	$cost(mergedBuilding.MergeOrgs(building, building_2))$
Executor	$BuildingManager$ from $building$ and $building_2$

Table 6.30: *MergeUnits* solution for the case of study

6.4 Conclusions

In this chapter, a subset of the forces that drive organizational change have been selected to illustrate a case of study about adaptation in OCMAS based on a virtual smart building. This case of study has been presented using a formal description with VOF and a graphical description using the same notation as the GORMAS methodology. Then, the selected forces have been presented by instantiating the templates that define a force, the factors that help detecting it, and the solutions to apply, previously presented in Chapter 4. They have been chosen here because they represent the situations that may appear more commonly in an organization like this, and they are a proper set to check how forces are defined in an OCMAS.

Therefore, the forces have been presented as useful to be detected and solved in an application, and the templates facilitate these tasks to designers and developers of OCMAS. The changes that may be carried out in this organization affect to different elements of the organization such as the role allocation, services, norms, and so on. This makes this case of study interesting to showcase from the point of view of the forces that appear, and the different factors and solutions cover an interesting range of possibilities.

In a general situation, a reorganization implies to improve the performance of the organization. In the specific case of the building, after a reorganization, the number of clients whose requests are fulfilled increases, thus increasing its utility. Therefore, taking into account the forces in the design of an OCMAS clearly improves the system because of the addition of dynamical properties that allow the organization to change makes it to be more open to new opportunities that increase the organizational utility.

7

Conclusions and Future Work

The final chapter of this thesis presents the conclusions on this work. First, the different contributions of this thesis are listed, explaining the advances developed during this work. Additionally, the future work section details the future challenges that are promising to be developed on this topic. Finally, a list of publications and research visits related to this thesis is included.

7.1 Contributions

This thesis presents a new proposal to model a Virtual Organization (VO) not only from a static point of view, without considering change during the runtime of the VO but in a dynamic way, taking into account the reasons that make an organization to adapt itself to guarantee a correct performance and/or to keep the organization alive. These reasons are known in the social sciences domain as Forces that Drive Organizational Change. They are classified into external and internal forces, depending on where the reason for change comes from.

The main contribution of this thesis is to present these forces for the domain of Virtual Organizations, thus creating Adaptive Virtual Organizations (Adaptive VOs). Each force is characterized by the factors that help organizational managers to detect the force and the solutions that can be carried out to take advantage of that force or to prevent damage to the organization. In order to express the forces properly, it is required to be provided with different tools that facilitate this task:

- **Virtual Organization Formalization (VOF)**, a formal language that has been defined to represent a VO that defines it following the Organizational Dimensions (structural, functional, environmental, normative and dynamical). Its formal approach makes this kind of languages more convenient to work in a computational environment rather than natural language.

- **Artifacts for Organizational Mechanisms**, a set of artifacts that encapsulate the functionalities of the three types of Organizational Mechanisms: informative, incentive, and coercive. They promote coordination within an organization from both a micro and a macro perspective. They comply with the specifications of the Agents & Artifacts conceptual framework, also including the workspaces to define the environment in a similar way than the physical environment is defined.

- **New Environment Dimension of the Virtual Organization Model.** Having both the Artifact for Organizational Mechanisms, and the workspaces taken from the Agents & Artifacts conceptual framework available, a new environment definition for the Virtual Organization Model is presented. Including both concepts enhances the Environment Dimension of VOM with new features that facilitate and improve the definition of the environment.

- **Templates to define forces in the design phase.** Each force is defined using three different types of tables: (i) a table specifying the general information of the force; (ii) a set of factors, where each factor that affects the force is represented by another table; and (iii) a set of solutions where, similarly to the factors, each solution is represented individually in a table. The templates are then used to specify the implementation phase of the organization.

These contributions have been tested using a case of study based on a smart virtual building, defining the organization using VOF, and applying the templates to represent some of the most representative forces.

Then, these forces can be integrated in a VO at implementation time using, for example, the Jacamo framework that makes use of Jason agents, and CArtAgO artifacts and workspaces. Moreover, we state that the proposal of this thesis is intended to be inserted into different organizational definitions and frameworks so as to enhance them with the possibility of detecting the forces that drive organizational change and also solving them.

7.2 Future Work

This section describes the different open issues and future challenges that the researchers and developers on the topics developed in this thesis must face in the future:

- **Introducing adaptation in different OCMAS proposals**. As stated in the previous section, the main proposal of this thesis, which is the description of the forces that drive organizational change, is able to be adapted into different frameworks for defining an OCMAS. As seen in the background chapter, most of the proposals for OCMAS do not take into account the forces, so it is interesting and promising to introduce the forces and supporting adaptation in other proposals.

- **Refinement of the templates for the solutions**. As it can be checked along this thesis, the templates that provide solutions for the forces are generic, not being as specific as the templates for the factors. There could be different possibilities to tackle this problem, such as: (i) study a wider range of cases of study to provide a template that fits all the range; or (ii) provide different templates for each solution, giving the decision about which template to choose to the organizational designer.

- **Integration in an OCMAS framework**. We have stated that the forces can be implemented by means of the Jacamo framework or any other framework. However, they could be not only used, but integrated into a framework in form of an abstract class, e.g., containing the basic features, which can be extended. For example, in the case of the THOMAS framework (ABC^{+}11), artifacts containing the different templates could be introduced as first class abstractions for the implemented VO that could be instantiated by organizational managers with the properties and operations of a specific domain.

- **Study of new forces**. In this thesis, a set of forces coming from the social sciences has been studied. However, new forces could appear in the social sciences domain (such as the globalization force that appeared a few decades ago) and also new forces focused only on agent organizations (not based on forces from human organizations) may be studied and discovered.

- **Emotional-based reorganization**. Introducing human emotions into agents is one of the most promising proposals in agent development (Sun06). It would be interesting to have available models and mechanisms to obtain the best reorganization decisions in an agent society. These models would be used along with the forces described in this thesis to know how, when, where, and why to change the social emotion, structure, functionality, etc. of a group.

7.3 Publications

Next, all of the publications that contain the results of this thesis are listed. These contributions are separated in three different listings, depending on the format of its publication: (i) journals that appear at the Science Citation Index (SCI); (ii) conferences, indicating the ranking awarded by the Computing Research and Education Association of Australasia (CORE); and (iii) book chapters.

7.3.1 Publication in SCI Journals

- **S. Esparcia**, E. Argente, R. Centeno and R. Hermoso. Enhancing MAS Environments with Organizational Mechanisms. International Journal on Artificial Intelligence Tools (IJAIT) Vol. 20 N. 4 pp. 663-690. (2011) (Impact factor: 0.321)

  Extended version of the definition of the Artifacts for Organizational Mechanisms described in Chapter 5.

7.3.2 Publications in Conferences

- **S. Esparcia**, E. Argente and V. Botti. An agent-oriented software engineering methodology to develop Adaptive Virtual Organizations. 22nd International Joint Conference on Artificial Intelligence Vol. 3 pp. 2796-2797. (2011) **(CORE A*)**

A short paper with a general overview of this thesis. This is described in Chapter 1.

- **S. Esparcia**, R. Centeno, R. Hermoso and E. Argente. Artifacting and Regulating the Environment of a Virtual Organization. 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2011) pp. 547-554. (2011) **(CORE B)**

  Extension of the Environment Dimension of the Virtual Organization Model to include Artifacts for Organizational Mechanisms and workspaces, as presented in Chapter 5.

- **S. Esparcia**, R. Centeno, R. Hermoso and E. Argente. Enhancing MAS Environments with Organizational Mechanisms. 22th International Conference on Tools with Artificial Intelligence (ICTAI 2010) Vol. 1 pp. 457-464. (2010) **(CORE B)**

  Definition of the Artifact for Organizational Mechanisms described in Chapter 5.

- **S. Esparcia**, O. Boissier and E. Argente. Design of Forces Driving Adaptation of Agent Organizations. 12th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2014) pp. 110-121. (2014) **(CORE C)**

  Paper with the presentation of the templates for the design of forces that drive organizational change, as presented in Chapter 4.

- **S. Esparcia** and E. Argente. Forces that drive organizational change in an Adaptive Virtual Organization. Sixth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2012) pp. 46-53. (2012) **(CORE C)**

Presentation of a first version of the templates, focusing on the analysis of the forces from Chapter 4.

- **S. Esparcia** and E. Argente. Formalizing Virtual Organizations. 3rd International Conference on Agents and Artificial Intelligence (ICAART 2011) Vol. 2 pp. 84-93. (2011) **(CORE C)**

  Definition of the Virtual Organization Formalization, as presented in Chapter 3.

- **S. Esparcia**, R. Centeno, R. Hermoso and E. Argente. Artifacting the Organizational Mechanisms: Adding Functionality in MAS Environments. 2010 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-10) pp. 352-355. (2010) **(CORE C)**

  Short paper presenting the Artifacts for Organizational Mechanisms described in Chapter 5.

- **S. Esparcia** and E. Argente. A functional taxonomy for artifacts. 5th International Conference on Hybrid Artificial Intelligence Systems (HAIS '10) Vol. 6077 pp. 159-167. (2010) **(CORE C)**

  Paper containing a description of different kinds of artifacts and grouping them into a taxonomy.

- **S. Esparcia**, V. Sanchez-Anguix and R. Aydogan. A negotiation approach for energy-aware room allocation systems. 1st Workshop on Conflict Resolution in Decision Making (COREDEMA 2013) Vol. 365 pp. 280-291. (2013)

  An alternate approach to the case of study presented in Chapter 6, focusing on a negotiation approach.

7.3.3 Book chapters

- E. Argente, H. Billhardt, C. E. Cuesta, **S. Esparcia**, J. Gormer, R. Hermoso, K. Kirikal, M. Lujak, J. S. Perez-Sotelo and K. Taveter. Adaptive Agent Organisations. Agreement Technologies pp. 321-356. (2013)

  Collaboration between relevant European researchers to give a state of the art about Adaptive Agent Organizations, being the Forces that Drive Organizational Approach one of the studied proposals.

- E. Argente, O. Boissier, **S. Esparcia**, J. Gormer, K. Kirikal and K. Taveter. Describing Agent Organisations. Agreement Technologies pp. 253-276. (2013)

  Collaboration between different European researchers to present a state of the art about descriptions of agent organizations, where VOF is included.

- **S. Esparcia** and E. Argente. Defining Virtual Organizations Following a Formal Approach. Agents and Artificial Intelligence Vol. 271 pp. 365-381. (2013)

  Extended and revised version of the first definition of VOF.

7.4 Research Visits

During the development of this thesis, the following research visits were carried out:

- 01-09-2013 to 30-11-2013. *École Nationale Supérieure des Mines de Saint-Étienne, France*, research visit supervised by Prof. Olivier Boissier on Forces that Drive Organizational Change.

- 21-02-2013 to 23-05-2013. *National Institute of Informatics, Tokyo, Japan*, research stay supervised by Prof. Ichiro Satoh on Mobility in Agent Coordination Patterns.

- 10-09-2012 to 22-09-2012. *École Nationale Supérieure des Mines de Saint-Étienne, France*, COST Action IC0801 Short Term Scientific Mission supervised by Prof. Olivier Boissier on Forces that Drive Organizational Change.

7.5 Funding

This PhD thesis has been funded by the FPI grant BES-2010-029822, awarded to the author by the *Ministerio de Economía y Competitividad* of the Spanish Government. This grant is related to the project “OVAMAH: Adaptive Virtual Organizations: Mechanisms, Architectures and Tools” under grant TIN2009-13839-C03-01. This project is described in Chapter 1.

Bibliography

[ABC^{+}08] R. Ali, V. Bryl, G. Cabri, M. Cossentino, F. Dalpiaz, P. Giorgini, A. Molesini, A. Omicini, M. Puviani, and V. Seidita. MEnSA Project - Methodologies for the Engineering of complex Software systms: Agent-based approach. Technical Report 1.2, UniTn, 2008. 219, 220

[ABC09] G. Aranda, V. Botti, and C. Carrascosa. MMOG based on MAS: The MMOG Layer. In *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 1149–1150. Decker, Sichman, Sierra and Castelfranchi (eds.), 2009. 12

[ABC^{+}11] E. Argente, V. Botti, C. Carrascosa, A. Giret, V. Julian, and M. Rebollo. An Abstract Architecture for Virtual Organizations: The THOMAS approach. *Knowledge and Information Systems*, pages 1–35, 2011. 76, 274

[ABFF^{+}11] E. Argente, G. Beydoun, R. Fuentes-Fernández, B. Henderson-Sellers, and G. Low. Modelling with Agents. *Agent-Oriented Software Engineering X*, pages 157–168, 2011. 176, 177

[ABJ11] E. Argente, V. Botti, and V. Julian. Gormas: An organizational-oriented methodological guideline for open mas. In *Agent-Oriented*

Software Engineering X, pages 32–47. Springer, 2011. 13, 17, 63, 178, 191, 220, 227

[AGV$^+$04] E. Argente, A. Giret, S. Valero, V. Julian, and V. Botti. Survey of mas methods and platforms focusing on organizational concepts. In *Frontiers in Artificial Intelligence and Applications*, volume 113, pages 309–316. IOS Press, 2004. 176

[AH05] M. Aguer-Hortal. *La era de las organizaciones virtuales.* Ediciones Pirámide, 2005. 14

[AJB09] E. Argente, V. Julian, and V. Botti. MAS Modeling based on Organizations. In *Post-Proceedings 9th International Workshop AOSE'08*, volume 5386, pages 16–30. Springer, 2009. 178, 190, 222

[AJGF11] J. Alberola, V. Julian, and A. Garcia-Fornes. A cost-based transition approach for multiagent systems reorganization. In *10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1221–1222. IFAAMAS, 2011. 129

[AJGF12] J. Alberola, V. Julian, and A. Garcia-Fornes. Multi-dimensional transition deliberation for organization adaptation in multiagent systems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1379–1380. International Foundation for Autonomous Agents and Multiagent Systems, 2012. 3, 62

[Ald99] H. Aldrich. *Organizations evolving.* Sage Publications Ltd, 1999. 2, 20, 28, 31, 125, 126, 158, 159

[Ald07] H. Aldrich. *Organizations and environments.* Stanford Business Books, 2007. 31, 32, 35, 36

[Ant04] R.J. Anthony. Emergence: A paradigm for robust and scalable distributed applications. In *Proceedings of IEEE International Conference on Autonomic Computing (ICAC'04)*, pages 132–139, 2004. 43

[APDD08] H. Aldewereld, L. Penserini, F. Dignum, and V. Dignum. Regulating Organizations: The ALIVE Approach. *Proceedings of ReMoD 2008*, 2008. 3, 21, 52, 53, 57

[ASM80] J.R. Abrial, S.A. Schuman, and B. Meyer. Specification language. *On the Construction of Programs: An advanced course*, pages 343–410, 1980. 70

[Bar95] C. Barnatt. Office space, cyberspace and virtual organization. *Journal of General Management*, 20:78–78, 1995. 15

[BBP93] J.A. Byrne, R. Brandt, and O. Port. The virtual corporation. *Business week*, 8(1993):98–103, 1993. 14

[BC95] W.P. Barnett and G.R. Carroll. Modeling Internal Organizational Change. *Annual review of sociology*, 21, 1995. 20, 32, 33, 36

[BdT04] G. Boella and L. Van der Torre. Regulative and Constitutive Norms in Normative Multi-Agent Systems. In *Proc. of KS*, pages 255–265. AAAI Press, 2004. 13

[BG08] V. Botti and A. Giret. *Anemona. A Multi-agent Methodology for Holonic Manufacturing Systems.* Springer Series in Advanced Manufacturing. Springer, 2008. 63

[BGG$^+$04] P. Bresciani, P. Girogini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An agent-oriented software development methodology. *Autonomous Agent and Multi-Agent Systems*, 8:203–236, 2004. 48, 66, 220

[BGPP02] C. Bernon, M.P. Gleizes, S. Peyruqueou, and G. Picard. ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering. In *ESAW'02*, 2002. 47

[BHW07] R. Bordini, J. Hübner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. Wiley. com, 2007. 8

[BKKW07] T. Bucher, M. Klesse, S. Kurpjuweit, and R. Winter. Situational method engineering. *Situational Method Engineering: Fundamentals and Experiences*, pages 33–48, 2007. 66

[BN00] T. Blecker and R. Neumann. Interorganizational Knowledge Management: Some Perspectives for Knowledge Oriented Strategic Management. *Knowledge management and virtual organizations*, pages 63–83, 2000. 15

[BSD07] M. Bača, M. Schatten, and D. Deranja. Autopoietic information systems in modern organizations. *Organizacija*, 40(3), 2007. 15

[CAB10] N. Criado, E. Argente, and V. Botti. Rational Strategies for Autonomous Norm Adoption. In *Proc. COIN@AAMAS*, 2010. 107

[CAJB09] N. Criado, E. Argente, V. Julian, and V. Botti. Designing virtual organizations. In *Proc. International Conference on Practical Ap-*

plications of Agents and Multi-Agent Systems, pages 440–449, 2009. 75, 83, 84

[Cal09] J. Calta. A taxonomy of adaptive systems. In *7th European Workshop on Multi-Agent Systems (EUMAS 2009)*, pages 1–13, 2009. 44

[Car97] Kathleen M Carley. Organizational adaptation. *Annals of Operations Research*, 75:25–47, 1997. 36

[CBHO09] R. Centeno, H. Billhardt, R. Hermoso, and S. Ossowski. Organising MAS: a formal model based on organisational mechanisms. In *Proc. Symposium on Applied Computing*, pages 740–746, 2009. 4, 9, 23, 24, 26, 175, 177

[CCS95] J. Coyle, M. Coyle, and N. Schnarr. The Soft-Side Challenges of the Virtual Corporation. *Human Resource Planning*, 18:41–42, 1995. 14

[Cla94] T. Clancy. The latest word from thoughtful executives. *Academy of Management Executive*, 8(5):8–10, 1994. 14

[Cla99] E.M. Clarke. Model checking. In *Foundations of Software Technology and Theoretical Computer Science*, page 54. Springer, 1999. 70

[Cos05] M. Cossentino. From requirements to code with the PASSI methodology. *Agent-oriented methodologies*, pages 79–106, 2005. 13, 66, 220

[CS96] K.M. Carley and D.M. Svoboda. Modeling organizational adaptation as a simulated annealing process. *Sociological Methods & Research*, 25(1):138, 1996. 36

[CZ10] L. Cernuzzi and F. Zambonelli. Adaptive Organizational Changes in Agent-Oriented Methodologies. *The Knowledge Engineering Review*, 2010. 52

[Daf03] R.L. Daft. *Organization Theory and Design*. South-Western College Pub, 2003. 2

[DD06] V. Dignum and F. Dignum. Towards formal semantics for reorganization. *Technical report UU-CS*, 2006. 22, 126

[DD07] V. Dignum and F. Dignum. A logic for agent organizations. In *Proceedings of the 3rd Workshop on Formal Approaches to Multi-Agent Systems (FAMAS)*. Citeseer, 2007. 68, 74

[DDF$^+$05] V. Dignum, F. Dignum, V. Furtado, A. Melo, and L. Sonenberg. Towards a Simulation Tool for Evaluating Dynamic Reorganization of Agents Societies. In *Proc. Socially Inspired Computing, AISB Convention*, volume 230, 2005. 58

[DeL09] S.A. 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. 65, 219

[Dig03] V. Dignum. *A model for organizational interaction: based on agents, founded in logic.* PhD thesis, Utrecht University, 2003. 74, 121

[Dig09] V. Dignum. *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models.* Information Science Reference, 2009. 16, 18, 40, 41

[DM98] G. Desanctis and P. Monge. Communication processes for virtual organizations. *Journal of Computer-Mediated Communication*, 3(4):0–0, 1998. 14

[DMWD02] V. Dignum, J.J. Meyer, H. Weigand, and F. Dignum. An organization-oriented model for agent societies. In *Proceedings of International Workshop on Regulated Agent-Based Social Systems: Theories and Applications*, 2002. 176

[DOM08] S. Deloach, W. Oyenan, and E. Matson. A capabilities-based model for adaptive organizations. *Autonomous Agents and Multi-Agent Systems*, 16(1):13–56, 2008. 21, 126

[dRCD08] A. da Rocha Costa and G. Dimuro. Semantical concepts for a formal structural dynamics of situated multiagent systems. *Proc. COIN*, pages 139–154, 2008. 67, 116, 121

[DSD04] MV Dignum, E. Sonenberg, and FPM Dignum. Dynamic Reorganization of Agent Societies. In *Proceedings of Workshop on Coordination in Emergent Agent Societies*, 2004. 18

[DW04] K.H. Dam and M. Winikoff. Comparing agent-oriented methodologies. In *Agent-Oriented Information Systems*, pages 78–93. Springer, 2004. 12

[DWH05] T. De Wolf and T. Holvoet. Towards a methodology for engineering self-organising emergent systems. *Self-Organization and Autonomic Informatics (I)*, 135(1):18–34, 2005. 43, 50

[EA10] S. Esparcia and E. Argente. A functional taxonomy for artifacts. In *Proceedings HAIS*, 2010. 180, 212

[EAP97] A. Eardley, D. Avison, and P. Powell. Strategic information systems: An analysis of development techniques which seek to incorporate strategic flexibility. *Journal of Organizational Computing*, 7(1):57–77, 1997. 18

[Eme91] E.A. Emerson. Temporal and modal logic, Handbook of theoretical computer science (vol. B): formal models and semantics, 1991. 68

[ERAS$^+$01] M. Esteva, J.A. Rodriguez-Aguilar, C. Sierra, P. Garcia, and J. Arcos. On the formal specification of electronic institutions. *Agent mediated electronic commerce*, pages 126–147, 2001. 216

[ESAA$^+$10] S. Esparcia, V. Sanchez-Anguix, E. Argente, A. Garcia-Fornes, and V. Julian. Integrating information extraction agents into a tourism recommender system. In *5th International Conference on Hybrid Artificial Intelligence Systems (HAIS 2010)*, volume 6077 of *Lecture Notes in Artificial Intelligence*, pages 193–200. Springer, 2010. 12

[FFMM94] 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. 13

[FG98] J. Ferber and O. Gutknetch. Aalaadin: A meta-model for the analysis and design of organizations in multi-agent systems. In *Proc. 3rd Int. Conference of Multi-Agent Systems (ICMAS-98)*, pages 128–135, 1998. 2

[FGM03] 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. 13, 16, 55, 126

[FK01] I. Foster and C. Kesselman. The Anatomy of the Grid - Enabling Scalable Virtual Organizations. *Int. J. Supercomput. Appl.*, 15:200–222, 2001. 16

[GAD07] D. Grossi, H. Aldewereld, and F. Dignum. Ubi lex, ibi poena: Designing norm enforcement in e-institutions. *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, pages 101–114, 2007. 218

[GAG10] E. Garcia, E. Argente, and A. Giret. Emfgormas: A case tool for developing service-oriented open mas. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1623–1624. International Foundation for Autonomous Agents and Multiagent Systems, 2010. 76

[Gal95] J.R. Galbraith. *Designing organizations.* Jossey-bass, 1995. 14

[GBKD05] B. Gateau, O. Boissier, D. Khadraoui, and E. Dubois. MoiseInst: An organizational model for specifying rights and duties of autonomous agents. In *Third European Workshop on Multi-Agent Systems (EUMAS 2005)*, pages 484–485. Citeseer, 2005. 60, 61, 122

[GCG99] M.P. Gleizes, V. Camps, and P. Glize. A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. In *Fourth European Congress of Systems Science*, 1999. 47

[GDD$^+$07] D. Grossi, F. Dignum, V. Dignum, M. Dastani, and L. Royakkers. Structural aspects of the evaluation of agent organizations. *LNCS*, 4386:3, 2007. 70, 71

[GJR$^+$09] A. Giret, V. Julian, M. Rebollo, E. Argente, C. Carrascosa, and V. Botti. An open architecture for service-oriented virtual organizations. In *Seventh international Workshop on Programming Multi-Agent Systems.PROMAS 2009*, pages 74–88, 2009. 216

[GJW08] M. Ghijsen, W. Jansweijer, and B. Wielinga. Towards a framework for agent coordination and reorganization, agentcore. *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, pages 1–14, 2008. 52

[GM95] R. Grenier and G. Metes. *Going virtual: Moving your organization into the 21st century.* Prentice Hall PTR, 1995. 14

[GM97] N. Glaser and P. Morignot. The reorganization of societies of autonomous agents. *Multi-Agent Rationality*, pages 98–111, 1997. 51

[GMP03] F. Giunchiglia, J. Mylopoulos, and A. Perini. The tropos software development methodology: processes, models and diagrams. In *Agent-Oriented Software Engineering III*, pages 162–173. Springer, 2003. 97

[GODOV08] J. Garcia-Ojeda, S. DeLoach, W. Oyenan, and J. Valenzuela. O-MaSE: a customizable approach to developing multiagent development processes. *Agent-Oriented Software Engineering VIII*, pages 1–15, 2008. 65

[GPL07] J. Gonzalez-Palacios and M. Luck. Towards compliance of agents in open multi-agent systems. In *Software Engineering for Multi-Agent Systems V*, LNCS. Springer, 2007. 13

[Gra59] P.P. Grasse. La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la theorie de la stigmergie: Essai d'interpretation du comportement des termites constructeurs. *Insectes sociaux*, 6(1):41–80, 1959. 45

[GS98] H. Gazendam and J. Simons. An analysis of the concept of equilibrium in organization theory. In *Proceedings of the Computational and Mathematical Organization Theory Workshop*, 1998. 18

[GVCO08] L. Gardelli, M. Viroli, M. Casadei, and A. Omicini. Designing self-organising environments with agents and artefacts: a simulation-driven approach. *International Journal of Agent-Oriented Software Engineering*, 2(2):171–195, 2008. 46

[HBKR10] J.F. Hubner, 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. 180, 214, 215, 216, 223

[HBO10] R. Hermoso, H. Billhardt, and S. Ossowski. Role Evolution in Open Multi-Agent Systems as an Information Source for Trust. In *Proc. AAMAS*, 2010. 212, 223

[HBSS00] M. Hannoun, O. Boissier, J. Sichman, and C. Sayettat. Moise: An organizational model for multi-agent systems. *Advances in Artificial Intelligence*, pages 156–165, 2000. 60, 61

[HBV10] J.F. Hubner, O. Boissier, and L. Vercouter. Instrumenting multi-agent organisations with reputation artifacts. *Proceedings of Coordination, Organizations, Institutions and Norms (COIN@ AAAI)*, pages 17–24, 2010. 180, 217, 223

[HJ85] L. Hrebiniak and W. Joyce. Organizational adaptation: strategic choice and environmental determinism. *Administrative science quarterly*, pages 336–349, 1985. 18

[HJST07] M. Hoogendoorn, C.M. Jonker, M.C. Schut, and J. Treur. Modeling centralized organization of organizational change. *Computational & Mathematical Organization Theory*, 13(2):147–184, 2007. 3, 4, 54, 55, 126

[HSB02a] J.F. Hubner, J.S. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. *Lecture notes in computer science*, pages 118–128, 2002. 83

[HSB02b] J.F. Hubner, J.S. Sichman, and O. Boissier. Moise+: towards a structural, functional, and deontic model for mas organization. In *Proc. of Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 501–502. ACM, 2002. 13, 60, 61

[HWHJ09] R. Haesevoets, D. Weyns, T. Holvoet, and W. Joosen. A formal model for self-adaptive and self-healing organizations. 2009. 49, 70

[IGG99] C.A. Iglesias, M. Garijo, and J.C. Gonzalez. A survey of agent-oriented methodologies. *Intelligent Agents V. Lecture Notes in Artificial Intelligence. Springer-Verlag*, 1999. 13

[JS93] D. Jaffe and C. Scott. Building a committed workplace: An empowered organization as a competitive advantage. *The new paradigm in business: Emerging strategies for leadership and organizational change*, pages 139–148, 1993. 20

[JSTY07] C.M. Jonker, A. Sharpanskykh, J. Treur, and P.I. Yolum. A framework for formal modeling and analysis of organizations. *Applied Intelligence*, 27(1):49–66, 2007. 71

[JT03] C.M. Jonker and J. Treur. A temporal-interactivist perspective on the dynamics of mental states. *Cognitive Systems Research*, 4(2):137–155, 2003. 71

[Kat98] B.R. Katzy. Design and implementation of virtual organizations. In *hicss*, page 0142. IEEE Computer Society, 1998. 14

[KGJ09] R. Kota, N. Gibbins, and N.R. Jennings. Self-organising agent organisations. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 797–804. International Foundation for Autonomous Agents and Multiagent Systems, 2009. 45, 47

[KLR96] S. Kelly, K. Lyytinen, and M. Rossi. Metaedit+ a fully configurable multi-user and multi-tool case and came environment. In *Advanced Information Systems Engineering*, pages 1–21. Springer, 1996. 208

[KM$^+$80] J. Kimberly, R. Miles, et al. *The organizational life cycle: Issues in the creation, transformation, and decline of organizations.* Jossey-Bass Publishers San Francisco, CA, 1980. 28

[Kri01] M. Krieger. Sociología de las organizaciones. Una introducción al comportamiento organizacional, 2001. 38

[Kru04] P. Kruchten. *The rational unified process: an introduction.* Addison-Wesley Professional, 2004. 47, 50

[LE98] C. Lemaitre and C.B. Excelente. Multi-agent organization approach. In *Proc. II Iberoamerican Workshop on DAI and MAS*, 1998. 83

[Lew51] K. Lewin. Field theory in social science. 1951. 2, 56, 126

[LF99] R. Larsson and S. Finkelstein. Integrating strategic, organizational, and human resource perspectives on mergers and acquisitions: A case survey of synergy realization. *Organization Science*, 10(1):1–26, 1999. 34

[Liu01] J. Liu. *Autonomous agents and multi-agent systems: explorations in learning, self-organization, and adaptive computation.* World Scientific, 2001. 125

[LM08] M. Luck and P. McBurney. Computing as interaction: agent and agreement technologies. In *Proc. of the 2008 IEEE International Conference on Distributed Human-Machine Systems*, 2008. 5

[Man96] M. Manzano. *Extensions of first order logic.* Cambridge Univ Pr, 1996. 69

[MCCM10] P. Mariachiara, M. Cossentino, G. Cabri, and A. Molesini. Building an agent methodology from fragments: the mensa experience. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 920–927. ACM, 2010. 66

[MDO08] A. Molesini, E. Denti, and A. Omicini. From aose methodologies to mas infrastructures: The soda case study. pages 300–317. Springer, 2008. 219, 220

[Mis97] F.C. Mish. *The Merriam-Webster dictionary.* Merriam-Webster, 1997. 17

[MLP98] MD Moreno-Luzón and FJ Peris. Strategic approaches, organizational design and quality management: Integration in a fit and contingency model. *International Journal of Quality Science*, 3(4):328–347, 1998. 221

[MMG+09] M. Morandini, F. Migeon, M.P. Gleizes, C. Maurel, L. Penserini, and A. Perini. A goal-oriented approach for modelling self-organising mas. *Engineering Societies in the Agents World X*, pages 33–48, 2009. 49

[MOS96] D. Mowery, J. Oxley, and B. Silverman. Strategic alliances and interfirm knowledge transfer. *Strategic management journal*, 17(S2):77–91, 1996. 34

[MPP08] M. Morandini, L. Penserini, and A. Perini. Towards goal-oriented development of self-adaptive systems. In *Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems*, pages 9–16. ACM, 2008. 48

[MTB+02] S. McShane, T. Travaglione, M. Bazley, P. Hancock, C.W.L. Hill, J. Berk, P. DeMarzo, RJ Stone, NM Tichy, WG Bennis, et al. Organisational Behaviour on the Pacific Rim. 2002. 33

[NMOD08] E. Nardini, A. Molesini, A. Omicini, and E. Denti. Spem on test: the soda case study. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 700–706. ACM, 2008. 123

[NPC+04] T.J. Norman, A. Preece, S. Chalmers, N.R. Jennings, M. Luck, V.D. Dang, T.D. Nguyen, V. Deora, J. Shao, W.A. Gray, et al. Agent-based formation of virtual organisations. *Knowledge-Based Systems*, 17(2-4):103–111, 2004. 16

[Odi65] G.S. Odiorne. *Management by objectives.* Pitman New York, 1965. 36

[Omi01] A. Omicini. SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems. *Agent-Oriented Software Engineering*, 1957:185–193, 2001. 13, 66, 176

[OMO08] E. Oliva, P. McBurney, and A. Omicini. Co-argumentation artifact for agent societies. *LNCS*, 4946:31, 2008. 180, 198, 213, 223

[ORV$^+$04] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In *Proc. AAMAS*, pages 286–293, 2004. 180, 214, 215, 223

[Pap03] M. Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *Proc. Web Information Systems Engineering. WISE 2003*, pages 3–12. IEEE, 2003. 122, 134

[PC09] M. Puviani and G. Cabri. MEnSA Project - Methodologies for the Engineering of complex Software systms: Agent-based approach. Technical Report 2.2, UniTn, 2009. 67

[PGSF05] J. Pavon, J. Gomez-Sanz, and R. Fuentes. The INGENIAS methodology and tools. *Agent-Oriented Methodologies*, pages 236–276, 2005. 13, 50, 63

[PHBG09] G. Picard, J.F. Hubner, O. Boissier, and M.P. Gleizes. Reorganisation and Self-organisation in Multi-Agent Systems. In *1st International Workshop on Organizational Modeling*, page 66, 2009. 22, 51, 52, 53, 54

[Pnu77] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE, 1977. 117

[PS06] V. Popova and A. Sharpanskykh. Process-Oriented Organization Modeling and Analysis Based on Constraints. Technical report, Citeseer, 2006. 69, 122

[PW71] D. Pugh and M. Weber. *Organization theory: selected readings.* Penguin, 1971. 126

[PW07] H. V. D. Parunak and D. Weyns. Special issue on environments for multi-agent systems. *Auton. Agents Multi-Agent Syst.*, 14(1):1–4, February 2007. 177

[Qui13] J. Quinn. Strategic outsourcing: leveraging knowledge capabilities. *Image*, 2013. 34

[RG97] A.S. Rao and M.P. Georgeff. Modeling rational agents within a BDI-architecture. *Readings in agents*, pages 317–328, 1997. 25

[Rob92] R. Robertson. *Globalization: Social theory and global culture*, volume 16. Sage Publications Ltd, 1992. 34

[ROV$^+$07] A. Ricci, A. Omicini, M. Viroli, L. Gardelli, and E. Oliva. Cognitive stigmergy: Towards a framework based on agents and artifacts. In Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, editors, *Environments for MultiAgent Systems III*, volume 4389 of *LNAI*, pages 124–140. Springer, May 2007. 46, 180, 218, 223

[RP09] A. Ricci and M. Piunti. Designing and Programming Agents' Environments in Multi-agent Systems. In *Handouts of the Eleventh* *edition of the European Agent System Summer School*, pages 3–31, 2009. 218, 223

[RVO06] A. Ricci, M. Viroli, and A. Omicini. CArtAgO: An Infrastructure for Engineering Computational Environments. In *Proceedings E4MAS*, pages 102–119, 2006. 8, 180, 219, 220

[RVO07] A. Ricci, M. Viroli, and A. Omicini. Give agents their artifacts: the A&A approach for engineering working environments in MAS. In *Proc. Int. Conf. on Autonomous Agents and Multiagent Systems*, page 150, 2007. 4, 103, 122, 175, 177, 179, 180

[SAJBGF11] V. Sanchez-Anguix, V. Julian, V. Botti, and A. Garcia-Fornes. Analyzing Intra-Team Strategies for Agent-Based Negotiation Teams. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, 2011. 12

[SBPS11] A. Sorici, O. Boissier, G. Picard, and A. Santi. Exploiting the jacamo framework for realising an adaptive room governance application. In *Proc. DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11*, pages 239–242. ACM, 2011. 225, 226

[SCFY96] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996. 193

[Sch13] M. Schatten. Reorganization in multi-agent architectures: An active graph grammar approach. *Business Systems Research*, 4(1):14–20, 2013. 72

[SGKK14] M. Schatten, P. Grd, M. Konecki, and R. Kudelić. Towards a formal conceptualization of organizational design techniques for large scale multi agent systems. *Procedia Technology*, 15:577–586, 2014. 72

[SP08] C. Sansores and J. Pavón. An adaptive agent model for self-organizing mas. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1639–1642. International Foundation for Autonomous Agents and Multiagent Systems, 2008. 50

[Sun06] R. Sun. *Cognition and multi-agent interaction: From cognitive modeling to social simulation.* Cambridge University Press, 2006. 274

[VDP06] H. Van Dyke Parunak. A survey of environments and mechanisms for human-human stigmergy. *Environments for Multi-Agent Systems II*, pages 163–186, 2006. 46

[VG91] J.P. Van Gigch. *System design modeling and metamodeling.* Springer, 1991. 176

[WJ95] M. Wooldridge and N.R. Jennings. Intelligent agents: Theory and practice. *Knowledge engineering review*, 10(2):115–152, 1995. 2

[Woo02] M. Wooldridge. *An Introduction to Multiagent Systems.* John Wiley and Sons, 2002. 12

[ZJW01] F. Zambonelli, N. Jennings, and M. Wooldridge. Organisational abstractions for the analysis and design of multi-agent systems. In *Agent-Oriented Software Engineering*, pages 407–422. Springer, 2001. 13

[ZJW03] F. Zambonelli, N.R. Jennings, and M. Wooldridge. Developing Multiagent Systems: The Gaia Methodology. *ACM Transactions on Software Engineering and Methodology*, 12:317–370, 2003. 13, 66, 219