

Project Number	ESPRIT / LTR / 24 939
Project Title	AGENT
Deliverable Type	Report

Deliverable Number	D A1
Internal ID	AG-98-07
Contractual Date of Delivery	01/09/98 (technical Annex)
Actual Date of Delivery	25/08/98
Version	3.0
Workpackage/Task contributing	A1
Authors	INPG - IGN
Confidentiality	Public

Title of Deliverable	Generalisation Modelling using an agent paradigm
-----------------------------	---

Abstract

This report examines the need for modelling the generalisation process within the (multi-)agent paradigm. It gives a short introduction to Multi-Agent Systems and the AEIO paradigm. It tackles the question "what are the agents in the generalisation process". Identifying which are the agents and the organisations in the generalisation process, it gives a framework for the modelling in the next phase of the project.

Keyword List

Agent, Attribute, Environment, Goal, Hierarchical approach, Interaction, Life-cycle, Organisation, Parallel decomposition, Problem Decomposition, Recursive approach.

Executive Summary

This report describes the research conducted for the A1 task of work package A.

As set out in the Technical Annex, the objective of this work package is to provide insight into the mapping of different generalisation techniques onto the modelling of agent and multi-agent systems. The tasks are focused on the following questions:

How to best represent geographical agents? An agent encapsulates a basic geographical object or a set of geographical objects, possibly using a standard data model, in terms of their geographical content. But we also need a model that defines and delimits the specific interactions of an agent. This ensures that an agent is capable of behaving in a co-ordinated and collaborative way, within the map environment (or GIS environment) and with the other agents. This allows an agent to strive for a solution to a generalisation problem while respecting the overall (global) and its own (local) constraints.

How to represent a society of agents? In Multi-Agent Systems, agents need to co-operate in order to achieve their local goals and the goals of the society as a whole. No agent possesses sufficient abilities nor resource or information to solve the entire problem alone. Therefore, appropriate societies and specific organisational structures are required for the generalisation purposes.

The first section of this report presents briefly multi-agent systems, the AEIO multi-agent oriented paradigm and its relevance to the purpose of generalisation. Section 2 aims to analyse in general terms the decomposition problem, identifies the most appropriate agents for generalisation and finally proposes a model for the agents. The possible approaches to describe a society of agents in terms of organisations are analysed within section 3, and a first approach to their use in the domain of generalisation is proposed.

Table of content

Executive Summary	2
Table of content.....	3
1 Multi-Agent Systems and Generalisation	5
1.1 Multi-Agent Systems	5
1.1.1 Introduction to Multi-Agent Systems	5
1.1.2 Distributed Problem Solving and Decentralised System Simulation	5
1.1.3 Contribution of the project to Multi-Agent Systems	6
1.2 AEIO paradigm	6
1.2.1 Agent, Environment, Interaction, Organisation	7
1.2.2 Multi-Agent Oriented Programming.....	10
1.3 Possible contribution of MAS for generalisation	12
2 Agents for Generalisation.....	14
2.1 Problem Decomposition	14
2.1.1 Extrinsic Decomposition.....	14
2.1.2 Intrinsic Decomposition.....	15
2.1.3 Problem decomposition, Agents, and Organisations	16
2.2 Identification of the Agents for generalisation.....	16
2.2.1 Procedural Approach (scenario 1).....	18
2.2.2 Constraints Approach (scenario 2).....	20
2.2.3 Phenomenological Approach (scenario 3)	21
2.2.4 Choice of Geographical agents for generalisation	22
2.3 Geographical agent modelling	23
2.3.1 Geographical objects versus Geographical agents	23
2.3.2 Lifecycle of geographical agents	24
2.3.3 Choice of an agent architecture.....	27
3 How to structure society of agents ?	29
3.1 Organisations in Multi-Agent Systems	29
3.1.1 Hierarchical approach	29
3.1.2 Recursive approach.....	31
3.1.3 Hybrid Approach.....	36
3.2 Identification of the Approach for Generalisation	36
3.2.1 Geographical agents and Geographical organisations	37
3.2.2 Geographical organisation needs	37
3.2.3 Choice of an Hybrid approach.....	38
4 Conclusion	39
5 Bibliography.....	40

List of figures

<i>Figure 1: The three statements of a MAS</i>	<i>11</i>
<i>Figure 2: Computational multi-agent systems.....</i>	<i>11</i>
<i>Figure 3: Extrinsic (left) and intrinsic (right) decomposition</i>	<i>14</i>
<i>Figure 4: The actors of generalisation</i>	<i>18</i>
<i>Figure 5: Procedural approach.....</i>	<i>19</i>
<i>Figure 6: Example with procedural approach</i>	<i>19</i>
<i>Figure 7: Constraints approach</i>	<i>20</i>
<i>Figure 8: Example with constraints approach</i>	<i>20</i>
<i>Figure 9: Phenomenological approach</i>	<i>21</i>
<i>Figure 10: Geographical agent and phenomena</i>	<i>22</i>
<i>Figure 11: Agent architecture (Demazeau 90)</i>	<i>28</i>
<i>Figure 12: recursive decomposition</i>	<i>32</i>
<i>Figure 13: Task hierarchy.....</i>	<i>32</i>
<i>Figure 14: Organisation in recursive approach.....</i>	<i>33</i>
<i>Figure 15: Agents in recursive approach</i>	<i>34</i>
<i>Figure 16: Environment in recursive approach</i>	<i>35</i>
<i>Figure 17: Interaction in recursive approach</i>	<i>36</i>
<i>Figure 18: Creation of meso-objects</i>	<i>38</i>

List of abbreviations

AEIO	Agent Environment Interaction Organisation
AI	Artificial Intelligence
CC	Communication Capabilities
DAI	Distributed Artificial Intelligence
DC	Decision Capabilities
DPS	Distributed Problem Solving
DSS	Decentralised System Simulation
GIG	Geographical Information System
MAS	Multi-Agents System
OO	Object-Oriented
PC	Perception Capabilities
RC	Reasoning Capabilities

1 Multi-Agent Systems and Generalisation

This first part introduces Multi-agent Systems (MAS). A particular approach of multi-agent oriented programming, the AEIO paradigm, is then described with some details because it will be used by the project. It has been developed for many years at INPG by one of our partners. The intuition why using MAS to solve the generalisation problem is finally briefly discussed in part 1.3.

1.1 Multi-Agent Systems

1.1.1 Introduction to Multi-Agent Systems

Multi-agent systems are ones in which several computational entities, called agents, interact with one another. The concept of ‘agent’ implies a problem solving entity that both perceives and acts upon the environment in which it is situated, applying its individual knowledge, skills, and other resources to accomplish high-level goals. Agents thus integrate many of the algorithms and processes that have been independently studied by researchers in artificial intelligence and more widely in computer science. Much of the conceptual power of this exciting new paradigm arises from the flexibility and sophistication of the interactions and organisations in which agents participate. Because an agent is relatively self-contained, it has a considerable degree of freedom in how it interacts with other computational and human agents. The study of multi-agent systems concentrates on the opportunities and pitfalls afforded by this freedom. Agents can communicate, cooperate, coordinate, and negotiate with one another, to advance both their individual goals and the good (or otherwise) of the overall system in which they are situated. Agent societies can be structured and mechanisms instituted to encourage particular kinds of interactions among the agents. Populations of agents acting on their individual perspectives can converge to systemic properties. Teams of agents, each providing a particular suite of capabilities needed by one another, can be constructed and deployed to collectively solve problems that are beyond their individual abilities. This teaming can even be done on the fly, and can include humans as well as heterogeneous computational agents [Demazeau 98a]. MAS is a real cross-disciplinary as researchers in physical, computational, natural, economical, social and life sciences pool their insights. They are drawn together by their mutual interest in understanding the phenomena of interacting agents, in investigating the interplay between agents as individuals and as participants in collective settings, and in formulating languages, architectures, and mechanisms that are specifically applicable to multi-agent systems, and the many horizons from which come the several partners of the projects are respecting this diversity.

1.1.2 Distributed Problem Solving and Decentralised System Simulation

Two traditional ways of approaching and using MAS have been identified in the past: distributed problem solving [Bond 88] and decentralised system simulation [Demazeau 97].

- Distributed problem solving (DPS) [Bond 88], traditionally identified as being Distributed AI (DAI), considers how the task of solving a particular problem can be divided among a number of agents that co-operate in dividing and sharing knowledge about the problem and about its evolving solution. In a pure DPS system, all interactions (co-operation, co-ordination) and strategies are incorporated as an integral part of the global system.
- Decentralised system simulation (DSS) [Demazeau 97], often identified as Multi-Agent Systems (MAS) [Moulin 96] for historical reasons [Demazeau 90], is concerned with the behaviour of a collection of (possibly pre-existing) autonomous agents aiming at solving a given problem. A DSS is a loosely-coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities. In a pure DSS, the control of the problem solving is decentralised among the agents.

We will actively use this distinction in the next section, when identifying the adequate matching of our generalisation problem with the work in MAS.

1.1.3 Contribution of the project to Multi-Agent Systems

The MAS community currently determines four main areas of work [Demazeau 98a]:

- The specification and understanding of the context in which a multi-agent system executes: how the languages, protocols, organisational structures, goals, and incentives influence the kinds of interactions among agents. This covers: communication languages and protocols (semantics, pragmatics), organisation and social structure (agent roles, social laws), co-operative problem solving (collective goals driving individual choices), decentralised systems (individual choices driving collective behaviour), and mechanism design (incentives for aligning individual and group goals).
- Techniques that agents use to reason about the multi-agent system: conflict resolution and negotiation, multi-agent planning, coalition formation and organisation self-design [Malville 98], agent modelling and plan recognition, multi-agent learning [Brauer 98], distributed search and constraint satisfaction, and foundations (multi-agent logics, game-theory, economics, philosophy).
- The fundamental reasoning techniques coupled with the multi-agent context need to be implemented and tested to evaluate the strengths and weaknesses of the theoretical underpinnings, as well as to provide practical tools for developing complex software systems. This includes : agent programming languages [Bussmann 98][Ferber 98], multi-agent programming frameworks, agent models and architectures, standards for multi-agent technology (interaction protocols, languages), development and engineering methodologies, evaluation of multi-agent systems, testbeds and development environments, and user interfaces and personalisable agents.
- Applications of multi-agent systems provide touchstones for measuring progress as well as illuminating important, previously overlooked problems that arise in the real world. The ICMAS'98 call covered the following domains: electronic commerce [Gimenez 98], cooperative information systems [Armstrong 98], distributed resource allocation, information agents on the internet [Itoh 98], multi-agent simulations of social and biological systems [Picault 98], multi-agent vision and robotics, believable agents in multi-agent settings, and interacting personal digital assistants.

While the research community is too large and diverse to agree on a particular research methodology, it is often the case that well-balanced research activities span several of the above-mentioned topics. That is, investigations into aspects of multi-agent systems often tie together ideas on the nature of the agent interactions, how the computational agents should operate within this framework, how the results have been developed and evaluated, and the practical significance of the work. At a first glance, the work performed in our project is relevant to the fourth topic about Applications of Multi-Agent Systems, and will provide there a new domain for applying MAS. It is also our hope and belief that our project will have non-neglectable impact at the other levels, especially from the point of view of agents and organisations models, MAS methodology, MAS oriented programming, testbed, and evaluation.

1.2 AEIO paradigm

Any MAS paradigm is based on the existence of agents that are entities that have goals to reach and communication capabilities to interact one with another. An agent owns mechanisms to act according to certain situations to reach its goals. The MAS paradigm comes from two different influences: the systemic and the AI.

- The systemic stresses on the holistic aspect (macroscopic view) of a problem and on the flow of information in between components. The systemic is interesting as it has developed many of its ideas in terms of interactions and communications.

- AI is the discipline which introduces the concept of cognition within processes. Different techniques such as blackboards, neural networks and expert systems have been developed to allow for a dynamic choice of actions according to certain situations. AI techniques have been heavily used for control and diagnostics issues.

The problems of such paradigms come from the difficulty in representing different behaviours and different interactions between the entities, which compose a general system. The systemic considers the macroscopic view of a problem and flow of information but does not help in describing behaviours and the traditional AI approach is very centralised and does not consider interactions.

Multi-agent systems were originally developed to artificially simulate systems modelling in different natural, life, economical and social sciences, where it is necessary to distinguish the component behaviours and to find a way to enrich communication capabilities (perception, information or order sending). Its application is spreading more and more either for distributed tasks and resources allocation (Web) or for specific applications where components are heterogeneous.

In the following we will describe the 4 main components of a MAS: Agent, Organisation, Environment and Interactions, as they have been identified in [Demazeau 95] and as they likely start to generate similar work in the MAS area.

1.2.1 Agent, Environment, Interaction, Organisation

This section aims at describing briefly the AEIO paradigm that will be used by the project [Demazeau 97]. This approach promotes that a multi-agent system can be decomposed according to four components. The first component is (A) the Agent as the basic component of a multi-agent system, the second one is the (I) interaction which range from very simple ones like physical forces exerted between agents, to very complex ones like speech acts (from [Searle 69] to [Vanderveken 94] as for the last studies) or interaction protocols ([Smith 80] [Sian 91] [Berthet 92] [Chang 92] [Campbell 92] [Burmeister 93] as initial studies in this very active direction currently). The third component is (O) the Organisation which is often identified as the entire multi-agent system or the society of agents, but which may clearly be exhibited as an independent feature of a MAS [Demazeau 97] [Baeijs 98]. The last component is (E) the Environment, which is, where the agents evolve; this environment can be physically or virtually modelled according to whether one chooses software or hardware agents.

1.2.1.1 A as Agents

Agents are the main entities of the MAS, and people usually conceive MAS starting by the agents first. There exist a huge literature about agents now, especially thanks to the ATAL workshops which are now being held and published yearly, [Wooldridge 95] for referencing the first one as example. We would like here to list some features of the agents without pretending generality, but limiting ourselves to our needs for the project.

An agent has goals to reach and mechanisms to act in order to reach those goals. This aspect defines what we call **autonomy**. If no supervisor chooses for the agent what it should do, however, we will see hereafter that an agent can be an *underling* when it receives orders from others entities and when it accepts such orders: some agents can refuse whilst others cannot, due to their possible roles in the organisations. Finally, the autonomy of an agent is not only related to its behaviours but also to the availability of the resources it needs to exist and behave within the global system.

An agent can also communicate with other agents in different ways. In order to communicate an agent needs perception mechanisms to know who he can communicate with. We call these communication *interactions*, which will be described hereafter, and we will explain why we make the distinction. More generally, an agent needs to have some actions, which external effect are perceived as *behaviours* (methods in the OO paradigm). If an agent exhibits different behaviours it needs to have a mechanism to choose the best mechanism to use. The mechanism of choice is the real intelligence of the agent. We distinguish between two different kinds of choice capacities, which make the difference between *reactive* and *cognitive* agents, though, despite the separation, the distinction is not always clear.

- A **reactive** agent reacts. This means that when it identifies a certain configuration - which is known as foreseeing - it knows what to do and how it should act. This is a *case based approach*. Potential cases are already described in its knowledge base. The agent uses information it perceives from the environment and from evaluating itself and compares this to the cases that are described in its knowledge base. When it recognises a configuration it acts according to the case description. Most of the progresses made in MAS with regards to reactive agents have been performed in close connection with natural and life sciences.
- A **cognitive** agent reasons. It also owns a knowledge base and also needs to obtain information about itself and other agents to act. The difference is that it is able to act even if the case it perceives is not already described in its knowledge base. It can also foresee what it should do next (planning capacities) and it is able to learn from its own experience or from the experience of others. To reason a cognitive agent often needs explicit representations of itself and of a part of the environment and more specially a representation of the agents with which it communicates. Most of the progresses made in MAS with regards to cognitive agents have been performed in close connection with social and human sciences.

In essence the main difference between the two kinds of agent is that the cognitive agent has much more flexibility in terms of its behaviours, as it is able to compare, to foresee and to learn. But it clearly needs more information and involves harder design to be built so that the real key is to always adapt the complexity of the choice of the agent to the only needs of the problem to be solved. According to the application domain an agent can be either information or a process.

1.2.1.2 E as Environment

The environment of the agent is characterised by every thing that is not itself. The environment represents the space where all agents live. For robots, the Environment represents the Euclidean space in which agents are moving. For software agents [Genesereth 94], the computer network might represent the environment. Whenever agents are located, the environment is often the metric space when available or easily defined. Obviously, if the environment of our agents refer to the geographical space, it will be the case in our project, but it is not always easy to determine, as in telecommunications domain for example [Van Aeken 99]. The agent needs a certain representation of its environment, even if it is not the entire environment. At least, it needs to be able to locate itself within the environment. The environment of an agent both defines and limits the capacity of interaction of an agent towards the rest of the system.

1.2.1.3 I as Interactions

Interactions rather than communications

In the MAS community, it is admitted that the term communication in MAS means more than it means in traditional Distributed Systems. Unfortunately, whereas communication already is standardised in this latter, there is not yet a common agreement about how communication should be treated in MAS, even some attempt has been made in the framework of KQML [Finin 94], and as there exists such current effort at an international level now at the FIPA (Foundation for Intelligent Physical Agents) level. How rich the syntax of a message should be to aid its fast interpretation? If it is possible to define application independent protocols? If there exists a set of primitives and if one could build protocols from these primitives? The answers to these questions have to be found at the MAS level and not only at the Distributed System level that might implement part of the communication between agents. To distinguish the MAS work with the standardised communication issues that exist in Distributed Systems, and for discussing communication issues between agents, we will usually refer to *interaction* rather than to communication.

Types of Interactions

Interactions are dynamic relationships between agents. These relationships are the results of a set of agent actions. The type of interaction between agents depends on:

- their goals: Agents goals can be compatible or not. If goals are compatible, agent can *cooperate* if not there is *competition*.

- their resource allocation: are agents in competition with other agents to realise their own tasks ? Among resources there is CPU, space, time schedule, access to information, etc. Conflicts occur when a set of agents need the same resources at the same time. To manage or avoid such conflicts, agents need to *co-ordinate* their actions. Three principles of co-ordination exist: the negotiation in between agents by means of information exchange, the arbitration where a upper level agent chooses for the agents what should be done, the reactive co-ordination through the environment, as usually performed by reactive agents but not only them [Demazeau 98b] [Ferber 95].
- their capabilities: An agent's behaviours, alone, are not sufficient to allow it to reach its goals. In this case, the agent must interact with others to find help.

Nature of Interactions

Interaction structures and languages range from physics-based interactions to speech acts. Physics-based models like electrostatic forces permit the expression of simple interactions of attraction and repulsion, and such models are widely used to model the interactions between reactive agents. Illustrations of such types of interactions for reactive agents can be found in [Demazeau 91a] [Ferber 91] [Demazeau 93]. This kind of interactions is implemented by communication through the environment, using the environment as a blackboard or a shared resource. Reactive agents do not encompass deliberative control, nor explicit reasoning, and nobody would expect such agents to hold structured conversations supported by speech acts as it is possible between cognitive agents.

Modes of interactions

The interactions between agents can be either direct (agents communicate one to another) or indirect, by means of the environment: the consequence of an agent's acts of any kind changes the environment that thus becomes different for the other agents. Some groups of agents, that will be described hereafter, are the results of agents' specific interactions. As an example, specific kind of interactions such as cooperations will be described in report B1.

1.2.1.4 O as Organisation

An organisation means that there is a sort of unity between agents that form a group. Whenever a group can be represented explicitly, we will call it an organisation. Implicit groups may exist from the point of view of a single agent whilst nor the other agents nor the user may not be aware about them [Van Aeken 98], but as they can be represented by the agent itself explicitly, they will be considered as organisations too. If we limit our definition of organisation here, it should be clear that the notion of organisation has several meaning in the MAS domain (see [Baeijs 95] for a complete bibliography). As an example of an alternative and more general definition, an organisation can be defined as a set of agents performing roles, interacting between them along organisational links [Demazeau 96] [Demazeau 98b]. But we should limit in our study to the necessary needs of modelling which the project requires.

In this simple framework of what are organisations, groups can be defined externally (prior to any process) according to external knowledge (i.e. exogene organisations) or can *emerge* from specific state of a group of agents (i.e. endogene organisations). So its means that organisation building mechanism can be top-down (from system specifications) or bottom-up (from the agent).

An organisation has different functions in a MAS:

- It can be used to represent more global goals and to realise global actions. In such a case an organisation is an agent, has its own goals and acts and interacts to reach its goals.
- It can be used to help agents to perform its tasks or for resources allocation. The organisation can change agent goals or give them information, for instance by increasing an agents perceptive abilities, or give them orders.
- It can be used to control agents. As local and more global goals are not always compatible, an organisation provides an overview to check if agents' actions respect or not global goals.

Thus an organisation can mediate, plan, decide, inform, order or execute at a more collective level than a single agent could perform it.. Such a role depends on the application domains. It should be clear to that an organisation can have different functions at different times, which will be the case in our project.

1.2.2 Multi-Agent Oriented Programming

By localising the intelligence of the system in the components and between the components of system, multi-agent systems try to validate the principle of the non-reducibility of the complexity. The final objectives of MAS consist of the development of theoretical studies, software tools, and practical realisations for the decentralised simulation of complex systems, and for the distributed solving of complex problems. Most of the validation of these studies is through comparison with other approaches, at the level of the models and at the level of the tools, leading to a clear perspective of agent oriented programming [Demazeau 97].

1.2.2.1 From Agent Oriented Programming to Multi-Agent Oriented Programming

The notion of Agent Oriented Programming has been introduced by [Shoham 92], as a natural extension of OO programming (we will go back to this heritage when detailing our choices for modelling geographical objects as agents). As introduced previously, it is the case that people design their agents first, before thinking about any other component, and in fact, AOP, the language introduced by Shoham, is following this idea. As it has been shown in [Demazeau 97], it is not always the case, and one may think to design his multi-agent system by the environment, the interactions, or the organisations first, before considering the agents themselves. This natural extension of Agent Oriented Programming is called Multi-Agent Oriented Programming and is actually developed only at INPG.

1.2.2.1 Multi-Agent Oriented Programming at INPG/LEIBNIZ

Basics

The MAGMA approach at INPG to the emerging paradigm of agent-oriented programming, underlies that the operational part of how the solution is found (DPS) — how the equilibrium states of the systems are reached (DSS) — is taken in charge by the multi-agent system itself and no more by the conceiver / user. This point of view of decoupling declarative semantics and operational semantics, which remembers the one that generated logic programming, enables us to consider the emerging paradigm of agent-oriented programming as a natural evolution of object-oriented programming, as soon as objects become agents, when they are associated with perception capabilities, communication capabilities, reasoning and decision capabilities. These capabilities enable the agents to autonomously process information relative to the problem to solve (DPS) — the system to simulate (DSS) —, the information relative to the other agents, to the domain, to the conceiver, and to the user.

Hypotheses

As hypotheses for our research, MAGMA studies adopt the three following statements :

- From a declarative point of view, a multi-agent system will be composed of several agents, an environment, a set of possible interactions, and possibly at least one organisation. This first statement will be called *the declarative equation*.
- From a more computational point of view, the functions that will be ensured by the multi-agent system consist of those of the agents — enhanced by the programmed interactions between the agents and the environment / the other agents, with respect to the organisation — , in addition to the ones which result from the added value generated by the agents evolving in a multi-agent world, which are usually encompassed under the name collective intelligence. This second statement will be called *the functional equation*
- To ensure completeness, and to try to reach the real notion of what could be agent-oriented programming, societies of agents should be considered as coarser agents at a higher level of abstraction and should be handled as such [Boissier 92] [Pleiad 92]. This assumption which has to be operationalised, will constitute our third statement, *the recursion principle*.

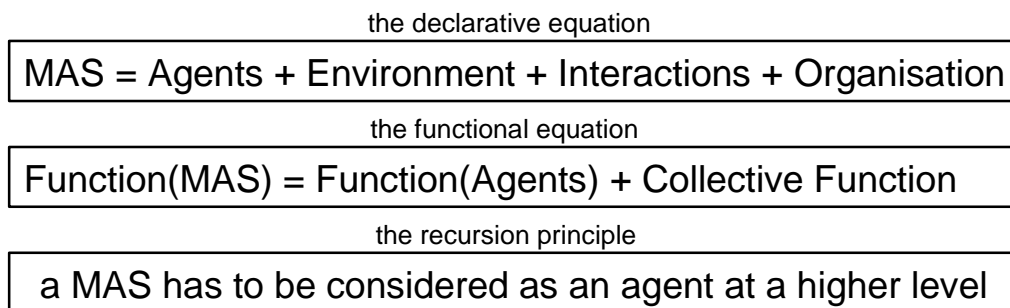


Figure 1: The three statements of a MAS

Approach

The agent models, or agent architectures, range from simple fine-grained automata to complex coarse-grained knowledge-based systems [Demazeau 90] [Demazeau 91b]. The environments are domain dependent, but at least are always spatial environments. Interaction structures and languages range from physics-based interactions to speech acts [Demazeau 94a]. Organisations range from dynamic ones inspired by biological studies, to more governed by social laws ones inspired by sociological studies.

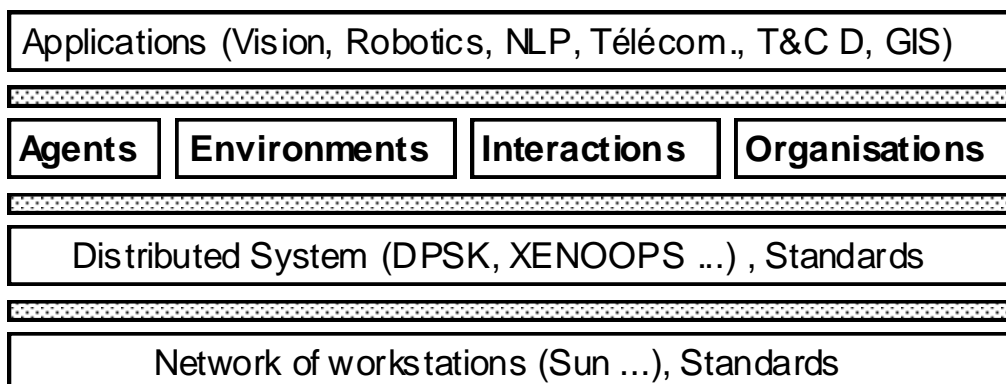


Figure 2: Computational multi-agent systems

For a given problem to solve — a system to simulate —, the user chooses the agent models, the environment model, the interactions models, and the organisation models to be instantiated. The specialisation of these generic tools in the context of the problem and regarding the application domain enables to build computational multi-agent systems. This clear methodology has led to efficient results in the past, and we will adopt it for our own work.

Background

- In order to tackle MAGMA research issues, the work has been academically split into two parts [Demazeau 91b]. The first part (e.g. [Boissier 94b]) deals mainly with the study of internal agent architectures (mainly cognitive ones), while the second part (e.g. [Demazeau 91a] [Demazeau 93]) studies mainly the external behaviour of agent groups (mainly reactive ones) evolving in a multi-agent world. This splitting might appear unclear, but in fact, when thinking of the second part as dealing with the study of the internal model of a society of agents, and remembering the recursion principle, it becomes obvious [Boissier 92].

- The agent models [Boissier 94b] [Ocelllo 94], the environment models, the interaction models [Berthet 92] [Populaire 93] [Demazeau 94a] [Sichman 95], the organisation models [Sichman 94] [Demazeau 96] [Baeijs 98], comprise the many toolboxes of the MAGMA integrative environment [Demazeau 95] [Ocelllo 97], which ensures the computational grouping of the selected models when building a multi-agent system to solve a given problem — to simulate a given system —. One of the toolboxes of the MAGMA integrative environment is devoted to interactions between agents — and by extension, between the agents and the environment, as soon as one may consider the environment as being an active

one. Such systems are built above performant distributed systems such as DPSK [Cardozo 93] or XENOOOPS [Bijnens 94] with respect to communication standards (TCP/IP, CORBA), that enable the implementation of low-level interactions between processes which are involved in the building of the multi-agent systems. In addition, these processes may be localised in different places in a network of computers, here again in respect with existing standards (UNIX, X, FRESCO).

- The generic tools which the MAGMA develop at the multi-agent level are tested through the building of numerous systems in various application domains: Computer Vision [Boissier 94a] [Boissier 94c] [Demazeau 94b] and Robotics [Demazeau 91a] [Demazeau 93] [Hassoun 92], Natural Language Processing [StÈfanini 93] and Telecommunications [Koning 95] [Van Aeken 99], Town and Country Development [Ferrand 94] and Geographical Information Systems [Baeijs 95b].

1.3 Possible contribution of MAS for generalisation

Generalisation has been studied for years by different researchers who all propose some algorithms to tackle different problems. However what is missing is a global system which could be used to choose dynamically what should be used, when and on what kind of data. Two modules exist CHANGE from Hanover University and MGE/MG from Intergraph. The first proposes a Batch process, which is only relevant for large scale and small-scale change. It owns very few contextual operations, and a user has to check for inconsistency and correct interactively the data. The second system MGE/MG provides a large set of algorithms (but mainly non-contextual ones) which have to be chosen by the user and controlled visually. The aim of our project is to include in the system enough information to let the system chose by itself what to do where and how. One could argue that expert systems could give an interesting answer for such a problem but actually it is nearly impossible to develop a large enough set of rules which foresee all the potential situations that could occur and which could provide coherent solutions. The point is that rules are in competition and can not be applied every where. Such aspect incites us to try to use autonomy to tackle generalisation problems. This paradigm of autonomy is not absolutely new for generalisation, as it has already been tried on two prototypes *Strategie* from the IGN/COGIT laboratory and *SIGMA* from the INPG/LEIBNIZ/MAGMA. This previous research helped to define this project and give us confidence for its success, taking benefit of the latest current studies toward multi-agent oriented programming.

The underlying principle of the agent based approach to generalisation is that we already have focused knowledge on generalisation in the sense that we know some mechanisms to generalise certain kinds of information such as buildings, roads, streets and land cover. Such knowledge can be viewed as micro-theories that can be used in certain situations only. Moreover, we know that some operations can occur at very local levels (geographic objects) whereas other operations such as displacement or selection are contextual and thus need to be applied to a set of geographical objects. So we have knowledge related to objects independently of one another: generalisation of a building, generalisation of a road, and we have knowledge related to the generalisation of a group of objects. We need to be able to represent different levels of information which correspond to different levels of analysis. We call such levels micro, meso and macro level [Ruas 98]. The agent paradigm seems thus very appropriate to tackle our problems as we can have knowledge at the agent level (which correspond to the previous micro-theory) and we can also represent the concept of groups of objects to tackle different problems. The Delaunay or Voronoï data structure use is now accepted as able to solve contextual conflicts, but the project will considered these groups of objects correspond as organisations in the MAS paradigm. As objects and groups of objects may have the same kind of functionalities in generalisation, we generically name them *situations* which correspond to a certain geographical configuration of information that should be considered in generalisation process. Situations are distinct from phenomena in that they may comprise a single object or a part of an object, such as a line segment.

The aim of A1 and B1 reports is to check whether our hypothesis of modelling generalisation by means of MAS is right or not and to propose a first MAS model in terms of static representation (the information required) and in terms of functionalities (the functions required to let the system work). The A1 report

focuses more on static representation: what could be Agent, what could be Organisation, whereas the B1 report focuses on the dynamic of this system. These two reports aim to give guidelines for the rest of the project. The real instantiation of the model will be done in the A3, A4, and B2 tasks from the results of A1, A2, B1 proposals. The results of A3 and A4 will be implemented in the E tasks.

2 Agents for Generalisation

This section concerns the modelling of agents for generalisation. Even if the choice of geographical information looks a priori the most obvious, other approaches might initially be proposed but do not seem to give satisfaction for generalisation purposes. These approaches are presented in 2.1 from a MAS point of view, before analysing them in 2.2 from a generalisation point of view. After adopting the geographical approach, the agent architecture we propose to use is presented in 2.3, which also present the life-cycle of such an agent for generalisation through an example.

2.1 Problem Decomposition

In MAS, we can distinguish four distinct phases to solve a problem or to simulate a system: problem decomposition, sub-problem allocation, sub-problem solution and sub-problem integration or result synthesis [Uma 93]. These four phases may differ in complexity based on a given application, but they are either explicitly or implicitly present in any distributed approach to problem solving. In spite of the dependency of the global solution on the first phase, problem decomposition, the research has mainly addressed the others phases. This section, based on current work by [Alvares 98], studies the problem decomposition in a conceptual way, without wondering if the decomposition is done by a designer (typical in pure DPS) or it is done by a society of agents that already exist (typical in pure DSS). In fact, solving a problem is not only a problem of the designer of a DPS, it is also the one of the every agent in a MAS which will interact with other agents as well as possibly with the user.

A first distinction can be made between extrinsic and intrinsic decomposition, depending on the point of view of agents. In the first one, more than one agents are able to solve the whole problem, and the use of many of them in parallel permits to speed up the problem solving. In the second one, the agents are themselves specialised according to some criterion; this *internal* specialisation may have been defined at the design time or acquired by learning during the activity of the agents. We illustrate these two kinds of decomposition on the following figure before detailing them :

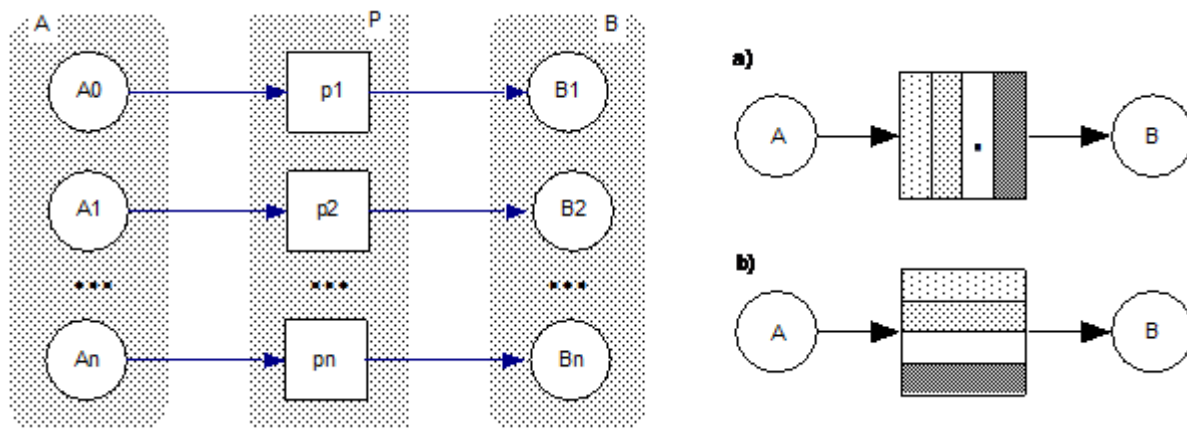


Figure 3: Extrinsic (left) and intrinsic (right) decomposition

2.1.1 Extrinsic Decomposition

In this kind of decomposition, the work is extrinsically spread among the agents. It is a pure physical decomposition of the work between the agents in order to achieve the goal, and that is what the designer of the MAS is classically performing when grounding his work, but essentially, this is what is done when the MAS is parallelized. Each agent executes the same kind of work. It means that the agents are identical

or at least that they have the same role in the problem solving process. Each single agent has all the knowledge and resolution strategies to do the whole work. Therefore, there is no agent specialisation or at least it is not necessary. Moreover, all agents have the same goal and there is no dependence nor hierarchy between them.

The extrinsic decomposition is performed:

- Spatially : The decomposition is based on a quantitative division of the pre-conditions or post-conditions of the problem. In this case, a problem is in fact a sum of several occurrence of an elementary problem.
- Temporally : The decomposition is based on time slices, i.e., there is an amount of time that an agent may be allocated to the problem.

In reference to the left part of the previous figure, and suppose a problem P with A and B as its preconditions and post-conditions, respectively, denoted by $P: A \rightarrow B$. Then, according to the spatial or temporal criterion above, we have that $P = n * p$, $A = n * a$ and $B = n * b$ such that $p: a \rightarrow b$, where n represents the number of times that the elementary problem occurs. The figure illustrates this physical criterion.

Spatial and Temporal decomposition allow a maximal parallelism between allocated agents and thus reduces the interval of time for solving the problem. As we have seen it too, they induce a typical similar between agents which then belong to the same type and usually infer no direct organisation between them.

2.1.2 Intrinsic Decomposition

In intrinsic decomposition, the decomposition is based on specialisation criteria. It means that the agents may be different or may have different roles and that a choice may occur in a sub-problem allocation whenever more than one agent can be allocated to the same sub-problem. Referring to the right part of the previous figure, this specialisation may be done according to two possible ways: *to solve the problem partially for any case*, or *to solve the problem entirely for some cases*. In a) the decomposition is *sequential*, in the sense that each sub-problem is a step towards the final goal. In b) the decomposition is *parallel*, in the sense that the sub-problems are independent [Menezes 96]. Thus, to detail the two approaches, the intrinsic decomposition is performed in two possible ways :

- Sequentially : There exists a *total order* of precedence between sub-problems. Each agent will solve only a part of the whole problem, according to its competence, i.e., each agent is specialised to solve a given sub-problem. Formally, a problem $P: A \rightarrow B$, can be sequentially decomposed by subproblems $P1: A \rightarrow A1$, $P2: A1 \rightarrow A2, \dots, Pn: A(n-1) \rightarrow B$ if $P = Pn \circ \dots \circ P2 \circ P1: A \rightarrow B$, as illustrated the following figure. In order to decompose sequentially a problem, one can consider some criteria like the abstraction level (generates level-specific tasks), the resource minimisation, an historical criterion, i.e., the way the sequence of tasks were traditionally or historically done, etc. The expertise of each agent is limited to a sub-problem; the knowledge which is required to handle the problem is limited to a part of the whole (we are considering only the competence that is needed to solve the problem, forgetting about all other possible competences of the agent). Therefore, while an allocated agent has the competence for solving the sub-problem, it may not have the competence for solving the whole problem and should again reallocate his work.

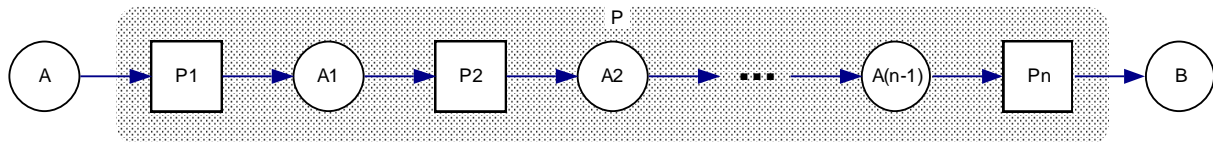


Figure 4: Intrinsic sequential decomposition

- Temporally : The pre- and post-conditions of the whole is the sum of the pre- and post-conditions of its sub-problems. Formally, a problem $P: A \rightarrow B$, can be decomposed in parallel by sub-problems $P1: A1 \rightarrow B1$, $P2: A2 \rightarrow B2$, ..., $Pn: An \rightarrow Bn$ if $A = A1 + A2 + \dots + An$ and $B = B1 + B2 + \dots + Bn$ (where "+" stands for sum or coproduct in a suitable category). For instance, in organisation theory induced by this kind of decomposition is called Product division: grouping all tasks related to each product [Fox 81] [Malone 87]. An agent allocated to a sub-problem is restricted to the pre and post-conditions of the sub-problem and thus encompasses only part of the entire domain knowledge. Therefore, there is a restriction in the search space of the agent improving the time needed to solve the sub-problem. Moreover, depending on the pre and post-conditions of the whole problem it may be the case that only one agent solves the problem. In this case, the agents are qualitative different, since they are specialised according to different preconditions.

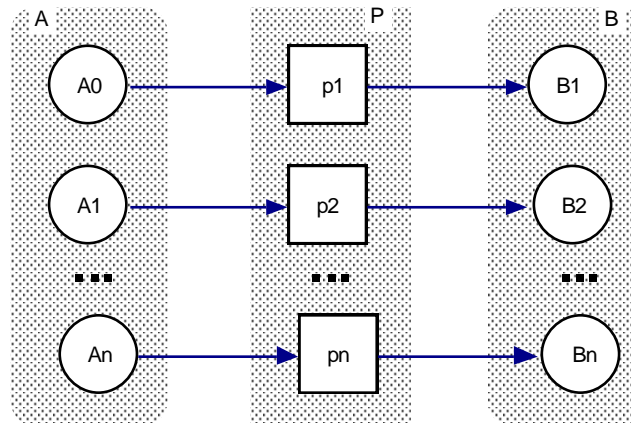


Figure 5: Intrinsic parallel decomposition

If sequential decomposition does not allow a maximal parallelism between allocated agents, it enable a minimisation of the complexity of the agents and induces a simple organisation between them, usually a master-slave dependency. The parallel decomposition allows some maximal parallelism between allocated agents, but induces a high complexity of agents which have to cooperate and synchronise, fixing thus a complex organisation between them.

2.1.3 Problem decomposition, Agents, and Organisations

Spatial, temporal, sequential and parallel decomposition are not mutually exclusive. Formally, we can combine these criteria, and each P considered above can be iteratively or recursively decomposed. Then, any kind of combination between the above is valid. Each combination will characterise a specific problem solving strategy. However, at each step of the design or of the reasoning of the agent, only one decomposition criteria can be chosen. But as we have seen in both previous sections, the consequences of each decomposition are not weak : decomposing a problem may restrict the possible choice of the agents that will be able to perform the subproblems, and induces specific features among agents in terms of their possible organisation. Knowing this, we will now try to identify from a more geographical point of view, which entities, and principally which agents and which organisations, can be identified and used to solve our generalisation problem. The end of this chapter will focus on agents. Organisations themselves have been hardly studied these last years from a design point of view, and we will postpone the study and the choice of organisation models from a geographical point of view after a deeper analysis of these MAS models. This will constitute the next chapter.

2.2 Identification of the Agents for generalisation

In order to tackle the generalisation problem using a Multi-Agent System approach, several questions arise.

- Which geographical entities should be modelled as agents?
- Which decomposition approach are we choosing and which constraints does this impose on the agents and related MAS entities?
- Which are the other components of the MAS, apart from the agents ?
- Which MAS entities (agents, interactions, environment, or organisations) will have the grounding power for solving the generalisation problem in terms of multi-agent oriented programming.

There are basically three kinds of entities involved:

- The algorithms used for generalisation,
- The user specifications (or required product), represented by means of constraints,
- The geographical information (GI) itself which has to be generalised

Each of these three entities play different roles in the generalisation process and previous research has shown that different kinds of knowledge are related to each of them and that different relationships exist between them. We will first address the main questions related to each of them, as well as their interrelationships, before proposing, and specifying in detail, three possible approaches (or scenarios) to model the generalisation process.

Procedural entities

Using a procedural approach means that the basic active entities are the generalisation algorithms such as smoothing, filtering, aggregation, selection, displacement, caricature, etc. The following questions then arise:

- On what kind of information can a particular algorithm be used in a generic way?
- Which constraint violations can be solved by a given algorithm?
- Which characteristics might be degraded by a given algorithm?
- What are the best parameter values for an algorithm?
- Which sequence of algorithms is the most efficient?

Constraints

Report A2 aims at defining constraints of generalisation from user needs. Constraints are related to the characteristics of geographical information, which either need to be changed or need to be maintained. Examples of constraints are accuracy, shape, quantity, repartition, ... The following questions arise from this:

- Can we express every user need in terms of constraints? What are the relevant measures to define a constraint?
- For a specific generalisation, which characteristics should be maintained, which ones should be generalised?
- How do we represent constraints that occur at meso or macro levels such as repartition constraints or quantity constraints?
- At a specific time, and related to specific information, what is a status of a constraint: what is its severity?, what is its flexibility?
- If different constraints are related to the same information, which is the best to solve first?
- Which algorithm should be used to solve a constraint violation?
- Which algorithms degrade a characteristic to maintain?

Geographical Information (explicit and Implicit)

Geographical information is the information that has to be generalised. Its content will change during the generalisation process by means of the algorithms in order to respect the constraints. As generalisation implies a change of the level of description of the geographic information, different views on the information are required (micro, meso and macro levels).

Theoretically each of these three approaches can be modelled within an agent or multi-agent systems context, or even a combination of the three approaches could be envisioned. But the system is more manageable if only one kind of information owns the decision making process.

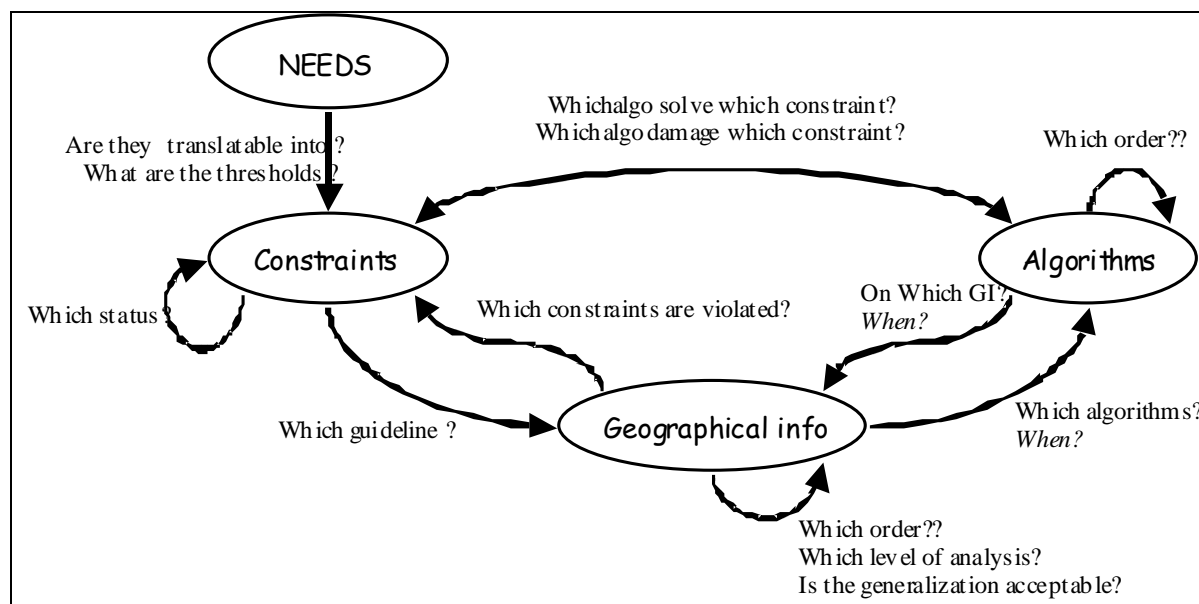


Figure 4: The actors of generalisation

In the next section we set out the three possible approaches (or scenarios) described above, and give their advantages and drawbacks if such an approach should be chosen within the AGENT project from a geographical point of view and from the point of view of the corresponding multi-agent systems approach.

2.2.1 Procedural Approach (scenario 1)

2.2.1.1 From a geographical point of view

If such an approach were used for the generalisation process then the procedures or algorithms would be the decision making entities of the system. This implies that they should know when to trigger themselves and on which geographical information (or type of geographical information) they are able to act. The problem is then to determine when and where to act (the goals of the algorithms). The inter-action framework between the algorithms could then be used to optimise the process meaning to find the best sequence of operations (co-ordination). On the other hand, the interactions between the algorithms could be used to negotiate between them to find the most appropriate action to be taken on an object.

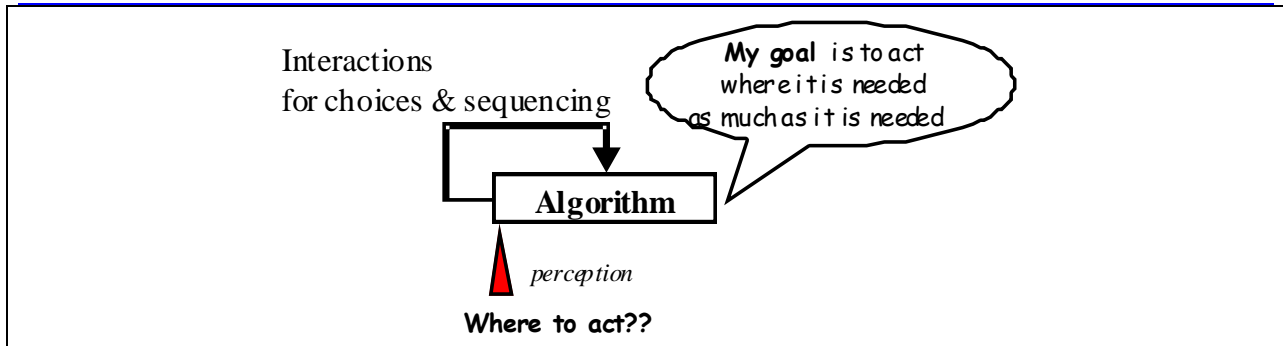


Figure 5: Procedural approach

For some operations we have enough knowledge to choose geographical objects to act upon (such as selection or displacement). A selection process governs the choice of geographical objects according to criteria on the objects. Generally, contextual operations have internal mechanisms to choose where to act. The limit of the process (where to stop) depends on geographical character: delete streets from smallest district until minimum size of district is over $XX \text{ m}^2$. But the operator (in this case the selection) is not able to know that it has to trigger itself!

The following problems then arise:

- In order to know where to act, the algorithm should be able to know which objects need its operation. For contextual operations the algorithm should be able to construct appropriate phenomena.
- Many algorithms have more or less the same goals (how to realise efficient co-operation without having to go through an intensive negotiation phase)
- Choosing between different processes that have the same goals can be very difficult.
- It is very difficult to control the convergence of the problem solving process by means of algorithms as we therefore need to evaluate the appropriate corresponding constraints

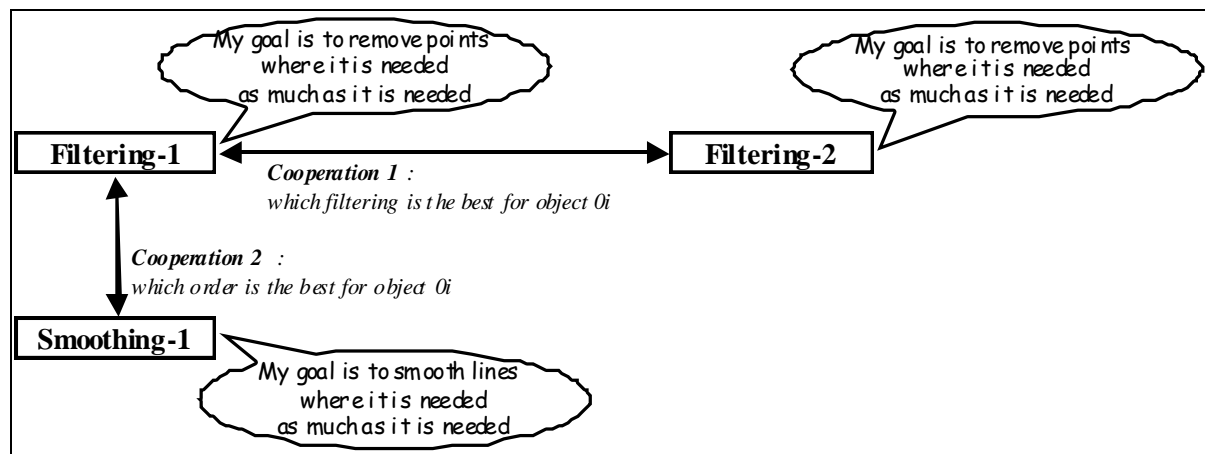


Figure 6: Example with procedural approach

2.2.1.2 From a MAS point of view

- Which geographical entities should be modelled as agents? The **procedures** or **algorithms**
- Which decomposition approach are we choosing? **extrinsic spatial** decomposition

- Which are the other components of the MAS ? **E** corresponds to the objects the procedures act upon. **I** is limited to negotiation to who becomes active to take an action on an object. **O** is used for finding the sequencing between the processes (coordination).
- Which MAS entities will have the power for solving the generalisation problem ? **A**

2.2.2 Constraints Approach (scenario 2)

2.2.2.1 From a geographical point of view

The second possibility is to model the constraints as active entities and thus the agents. The constraints then govern the process and trigger generalisation operations as their own methods. The system reaches a stable state when all constraints are satisfied.

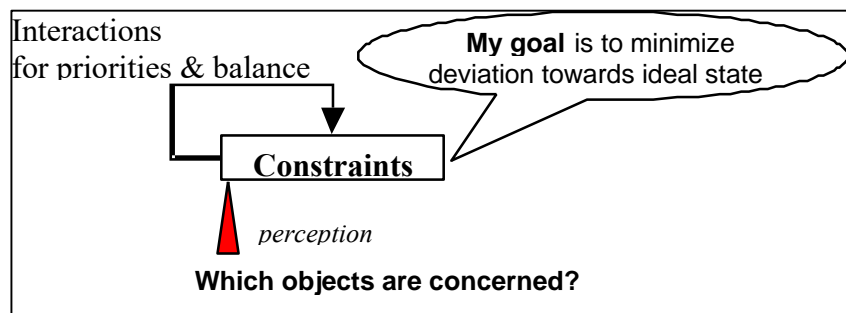


Figure 7: Constraints approach

This could be a very useful approach as constraints characterise conflicts and translate the user needs for the generalisation process. But, each constraint has its own scope and interest, and constraints might not all be satisfied. There is a competition between constraints that seek satisfaction, and this may require the building of an interaction framework between constraints which might be very complex. This interaction framework would allow for decision making (by means of priority between the constraints).

The problems are:

- Constraints occur at different levels (micro, meso and macro), which means that the interaction network might be very complex.
- Each time (because of research progress) a constraint is added n-1 interactions have to be added.
- The number of constraint-agents (decision makers) can be tremendous as each geographical information can be characterised by and is related to several constraints.

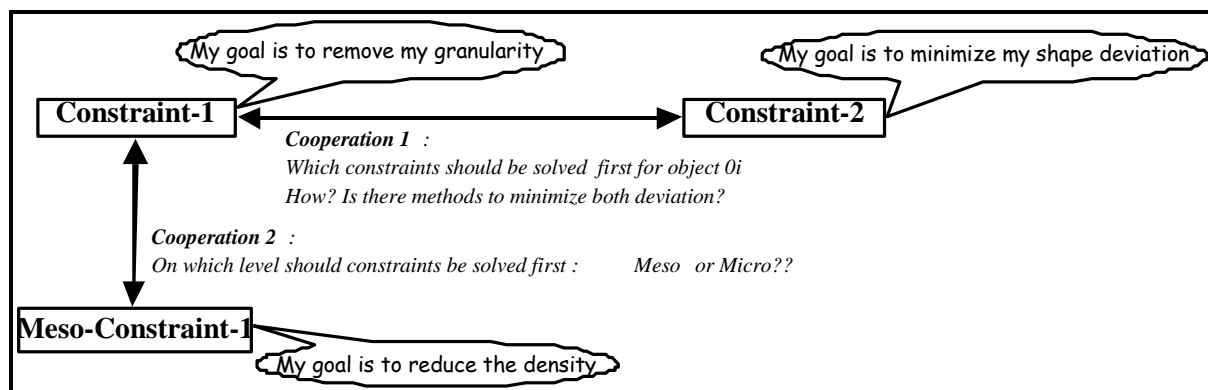


Figure 8: Example with constraints approach

2.2.2.2 From a MAS point of view

- Which geographical entities should be modelled as agents? The **constraints**
- Which decomposition approach are we choosing? **intrinsic temporal** decomposition
- Which are the other components of the MAS ? **E** corresponds to the objects the constraints are responsible for. **I** is limited to negotiation to who tries to fulfil the constraints. **O** is used for prioritisation among the constraints.
- Which MAS entities will have the power for solving the generalisation problem ? **O**

2.2.3 Phenomenological Approach (scenario 3)

2.2.3.1 From a geographical point of view

The most natural and straightforward approach from a geographer's point of view is to use the geographical information as the carrier of the generalisation process through the use of phenomena or situations. Every geographical phenomenon is characterised by means of its constraints and chooses its best behaviour to gradually solve its conflicts. The agents related to the phenomenon then reach a coherent stable state when the set of its constraints are globally respected.

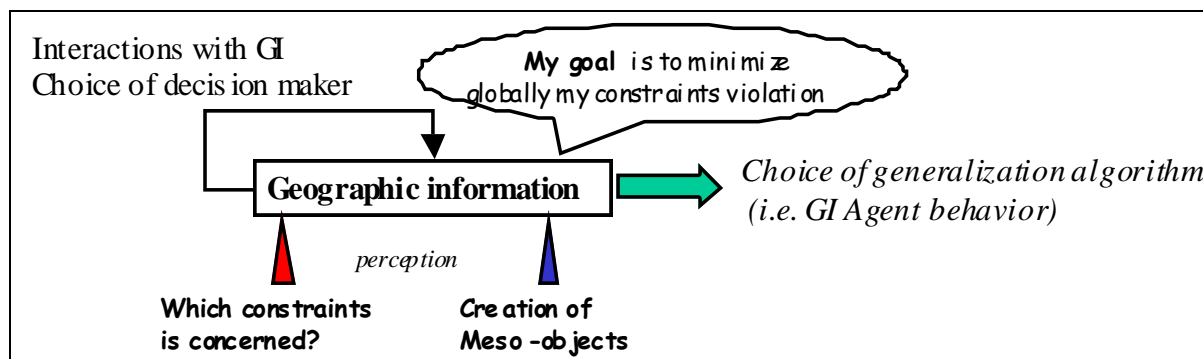


Figure 9: Phenomenological approach

This scenario is attractive because the decision level is geographic. Each geographical agent checks its constraints and chooses amongst its behaviours (generalisation algorithms) the one that might improve, in the most efficient way, its status according to its own constraints. In this way, constraints are used to help the decision making process (advise on what has to be done through characterisation of the situation) and for evaluation mechanisms (through the measures).

In this case, whenever a constraint is added, only one link is added which simplifies the evolution of the system. Moreover, if a constraint is refined (because a better measure is found) the system can easily be updated. This approach requires the explicit formalisation of geographical phenomena on which reasoning is performed, and on which the entire generalisation process then relies.

A geographical entity (or agent representing a phenomenon or situation) therefore has to possess its own methods for self-evaluation, through the use of measures evaluating the constraints.

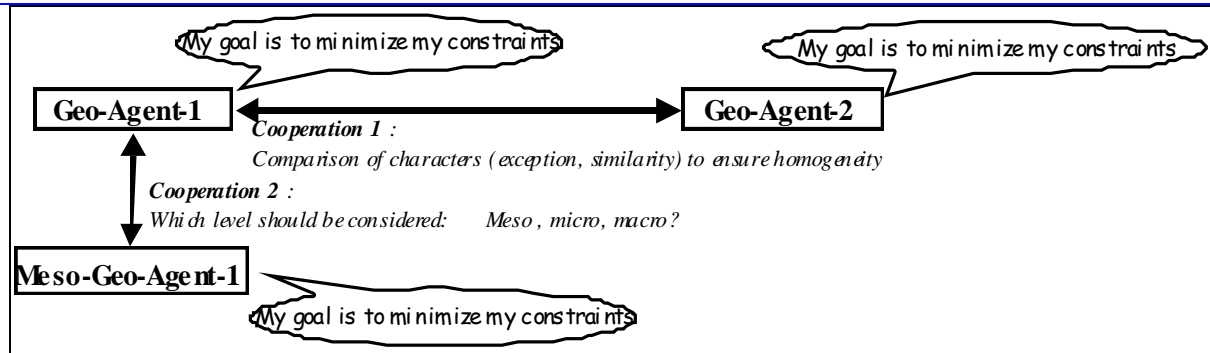


Figure 10: Geographical agent and phenomena

2.2.3.2 From a MAS point of view

- Which geographical entities should be modelled as agents? The **geographical information**
- Which decomposition approach are we choosing? A flexible and combined approach involving **intrinsic** and **extrinsic** decomposition, according to the types of geographical information which is handled at a given time.
- Which are the other components of the MAS ? **E** corresponds to the geographical database or geographical space. **I** corresponds to generalisation algorithms. **O** is used as a control structure in order to ensure coordination and cooperation between agents and for the ordering of the geographical object types.
- Which MAS entities will have the power for solving the generalisation problem ? **A** and **I**

2.2.4 Choice of Geographical agents for generalisation

We listed above the different possibilities of agent modelling for generalisation purposes. The choice between the three strategies relies mainly on our current knowledge of generalisation and on the necessary evolution of a GIS generalisation package. The aim of generalisation is to obtain generalised data that fulfils users' needs. The way procedures are done, the way constraints are computed can evolve, being improved by means of this project but also by others people efforts. What we do want, is create a system able to change and become enriched according to general progress in GIS research. Consequently, from a geographer's point of view, the decision maker should be the data and neither the algorithms nor the constraints. So that we should adopt the last approach.

Two others major reasons help us to take this decision:

- The current algorithms follow certain logic, which correspond to the procedural method of programming. We believe that using *constraints on characteristics* - as it is proposed in the project - will certainly prove an evolution in the conception of new algorithms devoted to generalisation. The main criticism of current algorithms is that most of them can not be properly governed because they do not use clear quantities that are related to generalisation constraints.
- In generalisation, a constraint can be not satisfied at the end of a process. What is important is that the geographical information is globally satisfied with its state which means that it is as close as possible to its heterogeneous goals. If we take constraints as agents, the quantity of information to handle by the MAS will be very difficult as most of the constraints are in competition with one another. It means that co-operation and collaboration would be the most difficult part of the project. Our knowledge in constraint network management is too fuzzy and too new to take such a risk. If we are not able to compute all the necessary constraints, then the system will not work, if we take constraints just as information to help geographical agent to generalise themselves, then the lack of some constraints does not block the whole process. It will just limit the quality of the results.

2.2.4.1 Choosing geographical objects as agents

At the lowest level of the system (micro level), the elementary agents are therefore coupled to a geographical object for which it is *responsible*. Task A3 must provide the basic geographical agent that will be used by the project.

2.2.4.2 Case of the geographical phenomena

However, the use of a basic geographical agent is not sufficient to provide generalisation. Views at other levels are also required. These views, called phenomena or situations, are geographical entities composed of sets of basic geographical objects (such as a district composed of houses and streets, or a street network within a town).

These phenomena may not be initially present in the database, but are required for different purposes (partitions, contextual operations, etc.). They therefore need to be detected and taken into consideration for different tasks within the generalisation process.

The questions related to geographical information are:

- What are the phenomena we have to consider? How do we detect them?
- Which geographical characteristics of information do we need to describe?
- At a specific time, how to choose the best information to generalise?
- When a piece of information is chosen, which algorithm should be applied to it?

From modelling point of view, these geographical phenomena can be modelled in MAS in different ways:

- As an agent (see next paragraph) which has to reach its own goals, and its own attributes (static model). It is linked to the recursive approach (a MAS has to be considered as an agent at a higher level). Task A4 must provide the agents that will be used by the project.
- As a set of agents (see section on organisation)
- As an organisation with capabilities of co-ordination and communication structure among a set of agents of a lower level.

As we will show it in the next chapter, we are leaning to model the phenomena that cannot be modelled as agents as organisations, and we will detail these in the report DB1.

2.3 Geographical agent modelling

This section aims at modelling the geographical agents in their static part. Generalities can be applied to basic geographical agents and to phenomena agents as well. The communication and interaction processes are detailed within report B1.

2.3.1 Geographical objects versus Geographical agents

2.3.1.1 Recap of Object-Oriented (OO) technology

This part is a recap of report E1 (Part 2- Gothic Database).

In an OO database, data is held as *objects* the nature of which are defined in *classes*. Classes may possess *methods* with associated *behaviours*, which are invoked in response to a message. A class may *inherit* from or be inherited by other classes. The *schema* of a database defines the classes and their inheritance hierarchies [Wegner 90].

Objects and Classes

An object is an instance of a class. For each class there may be many objects, but each object belongs to only one class. The class determines what values can be ascribed to an object. For example a *'Building'* class might sensibly have values such as *'Address'*, *'Owner'* and *'Number of occupants'*. These values can be used to differentiate between different building objects.

Methods and Behaviours

Object-orientation introduces the concept of messages; these are sent to objects. Exactly what messages an object can respond to, and the nature of that response is determined by the class of the object. Thus objects from different classes may respond differently to the same message. For example a *'Road'* object might respond to the message *'display_geometry'* by plotting a red line, whereas a *'River'* object might plot a blue line.

In Gothic, in common with many object-oriented systems, these responses are known as behaviours, and the messages are called methods. Methods provide the perfect mechanism for writing generic applications. The definition of the method provides a clear and exact specification of what an object must do if it is to behave correctly.

Inheritance

Inheritance allows the definition of new classes in terms of those that already exist. The simplest example is that of specialisation. One might wish to define the new class *'Hospital'*. Essentially hospital objects are like *Building* objects except they have an additional value, *'Number of beds'*. A *Hospital* is a special type of *Building*. Rather than define the entire class, one can use a shorthand and say that the *Hospital* class inherits from the *Building* class and that it has, in addition, the *'Number of beds'* value. The *Hospital* is said to be a sub-class or child of the *Building*, and the *Building* is the super-class or parent of the *Hospital*. Using this technique one can develop a tree of classes, known as the class hierarchy.

Inheritance also applies to methods and behaviours. For the purposes of customisation it is important to note that it is possible to override the behaviour of an inherited method

2.3.1.2 Agent vs Object

Using the Multi-Agent paradigm does not impose some kind of limitation to the classical object-oriented approach but completely encapsulates it; but using the multi-agent approach adds autonomy and local decision making capabilities to classical active objects.

Similar to traditional Object-Oriented Programming, agent might have:

- Attributes that characterise it or the entity it is representing. For basic geographical agent, basic attributes are its geometry (or a set of geometry) and its semantic attributes (as the line number of a road). A phenomena agent might possess semantic attributes (as density) and complex structure which can represent it (as Delaunay triangulation- see section 5).
- Execution capabilities (behaviours).

This report principally highlights the differences between an agent and object, i.e. the capabilities of an agent to take by itself decision (instead of receiving a message) in order to reach its goals, and more generally its autonomy.

2.3.2 Lifecycle of geographical agents

This section will present briefly the lifecycle of an agent. The dynamic of the system will be more developed in report B1 precisely in chapter 1. We will then just give an example of the kind of goals a geographical agent can have and what it could do to reach its set of heterogeneous goals. To illustrate this part we will take the example of building generalisation, without considering its interactions with its neighbourhood.

The life cycle of an agent represents its gradual generalisation from a state where it has a lot of conflict to a final state where it respects generalisation specifications. In agent terminology, the geographical agent has goals to reach and uses its behaviours, which are its generalisation operation, to reach them. At final stage, an agent should be happy with itself. Its happiness is based on its self-evaluation of the distance between what it is and what it would dream to be its state. As generalisation reduces information, the distance is rarely null, but still has to be acceptable. The main principle of the lifecycle is then based on self-dynamic characterisation and on comparison with generalisation constraints. Moreover, each decision an agent takes should be checked by itself, as it can have taken a bad decision.

2.3.2.1 Example in generalisation (building case)

It is useful to analyse how a geographical agent will work according to the proposed architecture.

One characteristic of this kind of agent architecture is that it uses goals. It seems generally possible for an agent to develop its own goals, to give priorities to goals and to modify them, but for this type of application certain goals will be externally defined and imposed upon the agents. Achieving goals - or making as much progress toward them as may be possible - will translate into reaching acceptable generalisation solutions. One therefore should be able to formulate goals for the agents from identified constraints to generalisation. At least sometimes, direct mappings can be found between goals and constraints, and in the process various key parameters and types of data that are involved.

Report A2 identified the main constraint in generalisation. For buildings, the main constraints are divided into contextual and internal constraints. Contextual constraints are related to the maintenance of specific relations (such as proximity) towards other object. We will just describe internal constraints:

- 1□ Size-constraint: Building should have minimum size to be interpretable. This size depends on scale and visual interpretation thresholds
- 2□ Granularity constraint: The internal shapes of a building should be big enough to be readable
- 3□ Width-constraint: The width within a building should be wide enough to avoid misinterpretation
- 4□ Shape-constraint: The shape of a building should be preserved as possible
- 5□ Accuracy-constraint: The absolute position of building should be preserved as possible.
- 6□ Orientation constraint: The orientation of building should be preserved as possible

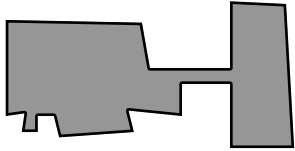
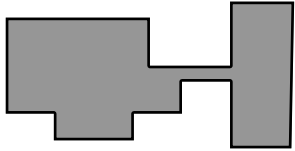
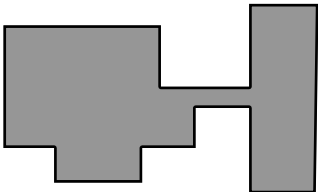
The goal of a building-agent will thus be the following:

- 1□ I should be big enough
- 2□ My internal shapes should be readable, without any overlapping
- 3□ I should not have internal conflict in my border opposite sides: I should be wide enough
- 4□ I should try to preserve my shape
- 5□ I should be not too far to my initial position
- 6□ I should preserve my main orientation.

In order to evaluate it, the agent needs quantitative values of itself and of goals to reach. The following goals have to be translated in values to reach. Such values to reach are computed from generalisation specifications (see report A2) and depend on the measures used:

- 1□ Size: $> 300\text{m}^2$
- 2□ Internal-shape: $> 10\text{m}$
- 3□ Internal-width: $> 20\text{m}$
- 4□ Elongation: $\varepsilon [\text{Initial-elongation} - 0.1, \text{Initial-elongation} + 0.1]$
- 5□ Hausdorff-distance: $< 20\text{m}$ (from initial position)
- 6□ Main orientation: $\varepsilon [\text{Initial-main_orientation} - 0.1, \text{Initial-main_orientation} + 0.1]$

All of these goals co-exist and may be in contradiction one to another. The building-agent will try to balance the satisfaction of all of them.

	<div>Size = 250 m² Goal unsatisfied</div> <div>Smallest shape = 7m Goal unsatisfied</div> <div>Square angles dev = 5° Goal unsatisfied</div> <div>Border width = 11 m Goal unsatisfied</div> <div>Δ_Elongation = 0 Goal satisfied</div> <div>Δ_Orientation = 0° Goal satisfied</div> <div>Hausdorff = 0 Goal satisfied</div>
<i>Initial stage</i>	
	<div>Size = 243 m² Goal unsatisfied</div> <div>Smallest shape = 20 m Goal satisfied</div> <div>Square angles dev. = 0° Goal satisfied</div> <div>Border width = 11 m Goal unsatisfied</div> <div>Δ_Elongation = 0 Goal satisfied</div> <div>Δ_Orientation = 0° Goal satisfied</div> <div>Hausdorff = 2 m Goal satisfied</div>
<i>A simplification (constrained by square angles)</i>	
	<div>Size = 300 m² Goal satisfied</div> <div>Smallest shape = 35 m Goal satisfied</div> <div>Square angles dev. = 0° Goal satisfied</div> <div>Border width = 15 m Goal unsatisfied</div> <div>Δ_Elongation = 0.15 Goal unsatisfied</div> <div>Δ_Orientation = 0° Goal satisfied</div> <div>Hausdorff = 18 m Goal satisfied</div>
<i>A 'bad' dilation</i>	

Within this framework, goals represent criteria derived from constraints to generalisation. For an agent to realise its goals, it may or may not have to act. Whether it acts or not depends on if problems are detected in its realm of its competence. Such diagnosis can be performed by the agent itself or for it by other agents, which then inform the one responsible for handling the problems. In general, these diagnostics are called measures, as they are (normally) quantitative assessments of spatial relations that measure geometric, topological and other properties. They can qualify a map as a whole, regions within a map, feature classes, individual features, or groups of features.

Likewise, once agents act to invoke generalisation operators in response to problems described by measures, it is usually necessary to re-assess the situation to determine if the solutions are satisfactory. More often than not, the same measures would be used following generalisation as were employed to trigger it.

Assessing goal achievement in large measure amounts to determining whether constraints have been satisfied. Three types of results can be anticipated:

2.3.2.2 Life-cycle

As mentioned before, agents behave in an autonomous way and throughout the entire problem solving process recast to what is happening in their local environment. They are constantly looking to satisfy their own goals and the goals of the society as a whole. Minor changes in their environment cause an immediate reactivation of the agents looking for a new stable state. This implies that the agents are persistent and continue 'to live' during the entire generalisation process.

The internal behaviour of an agent, from a functional perspective within its lifecycle, can be defined as follows:

Process	Description
IDENTIFY CONSTRAINT	Determine how an object or phenomena are constrained.
CHARACTERISE	Determine the actual situation through the use of measures.
EVALUATE	Compare the characterised situation and constraint. Evaluate whether the constraint is violated
PROPOSE	Suggest a variety of solutions to resolve the conflict.
REASON	Choose an appropriate solution from those proposed according to the global goal.
TRIGGER	Execute the chosen solution
CONCLUDE	Characterise & evaluate solution. Reason again if required.

For further information about the processes we would like to refer to deliverable E1, in which the processes are detailed from a geographical point of view. We would like to highlight here the mapping between this functional description and the agent architecture:

- The first three steps (**identify, characterise, and evaluate**) feed the agent's knowledge and instantiate its goals (K and G in the architecture), through the perception or communication capabilities (PC and CC).
- Once these items are analysed using its reasoning capabilities (RC) the agent will establish its set of possible plans (P) and thus will enter the phase of **proposing** a variety of solutions.
- Given its decision capabilities (DC) it will **reason** upon its possible plans and make a choice (C) of the plan that has to be instantiated.
- Confronting this choice with its execution capabilities (EC), it will then **trigger** and execute a set of actions (A) either interacting with the environment or with other agents.
- Closing the loop by perceiving or communicating once again, it will then **conclude** by characterising the situation once again and evaluating the obtained solution.

This approach means that the agent constantly goes through the mentioned perception-reason-decide-action cycle in order to reach a stable equilibrium state.

2.3.3 Choice of an agent architecture

Within the context of this research several agent architectures were examined ranging from pure reactive ones [Baeijs 95] to pure cognitive ones [Sichman 95] as well as agent architectures encompassing both reactive and deliberative aspects [Boissier 93] [Occello 97].

ESPRIT/LTR/24 939

Given the nature of the generalisation problem, according to the approach which has been chosen as identified above (which leads to constraints on the agent themselves), according to the modelling experience of the MAGMA group, we will choose the architecture developed in [Demazeau 90] for solving our generalisation problem. It uses the advantages of a BDI-like architecture, while being able at the same time to behave in a reactive way when needed. It fully offers the possibility of the use of a mixed decomposition problem, while ensuring computational power along the A and the I, as required by the chosen approach.

The general architecture is illustrated in the following figure 13:

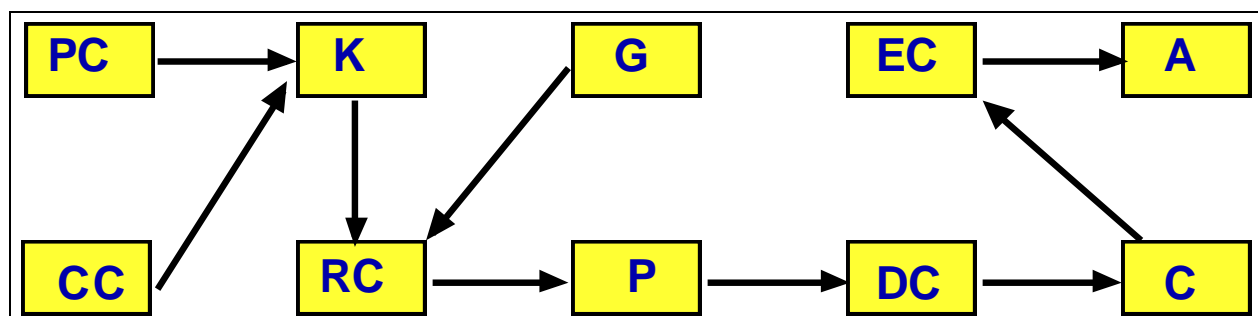


Figure 11: Agent architecture (Demazeau 90)

Within this architecture, an agent gathers his input through perception stimuli. These can be either first hand (obtained through perception of the environment) by using its perception capabilities (PC), or second hand information (obtained through communication with other agents) by using its communication capabilities (CC). Once the input has been gathered it feeds its knowledge (K), using its own local goals (G) and given its reasoning capabilities (RC) the agent will create a set of possible plans (P) in order to reach its goals;

Once the plans created, the agent uses its decision capabilities (DC) to choose the best plan to be instantiated with respect to its goals and the knowledge it possesses. This choice (C) is then confronted with its execution capabilities (EC) and generates a set of actions (A). These actions can be either interactions with the environment or interactions with other agents.

An agent goes continuously through this perception-reason-decision-action cycle until its goals are satisfied, which means that it has reached a stable equilibrium state. As agents behave in an autonomous way, a small disturbance in the agent's environment, or through interactions with other agents, leading to a non-stable local state of the agent, will trigger the agent's behaviour to look for a new stable state.

3 How to structure society of agents ?

Geographical agents and premises of organisations for generalisation have been identified in the previous section. The problem decomposition approach which has been chosen induces no real constraints on agents nor organisations as they may adapt to any circumstances. This chapter aims at structuring the society of agents, i.e. the set of all agents, through organisations, and thus, if grounded on MAS knowledge, the decision may be taken from a pure geographer point of view, according to existing (or future) possible MAS organisations which are (or will be soon) available. The possible architectures of organisations (as we have limited it before, we remind here that the society is seen there mainly through organisations), hierarchical and recursive approaches are presented within a first part. As for the agent modelling point of view, the case of organisations for generalisation is correspondingly covered in the last part. This section is entirely related to the report B1, which details organisation structures and their capabilities.

3.1 Organisations in Multi-Agent Systems

Two complementary approaches are presented here: Hierarchical approach that is the most classical one, and Recursive approach that is more recent.

3.1.1 Hierarchical approach

3.1.1.1 Definition

In the Multi-Agent literature hierarchical architectures have been introduced by [Fox 81] and [Malone 87]. This notion deals with three types of structures [Baeijs 95b]: simple hierarchies, multi-level hierarchies, and multi-division hierarchies, also called decentralised organisations. A formal definition found in [Boissier 93] and taken from [Mesarovic 70] comments that "each sub-system constrains the operation of the lower level sub-systems, which in turn send information back, it is then translated and passed on to the upper level".

The features of such hierarchies are:

- A hierarchy shows a tree-like structure.
- A node can be considered as giving orders to the ones below, and as a slave for the ones above.
- Prior to giving orders a node is given a task that it breaks into several subtasks and distributes them to some nodes at a lower level according to their abilities.
- This type of approach allows large tasks to be achieved since at each level several agents ensure the work will be decomposed and spread between the agents thus increasing productivity.
- Each operation is seen as an order for the executing nodes. This enables to reduce the latent period between the request for a job and its achievement, since there is no back and forth communication process.
- Relations between levels are clearly defined.
- Each node's role is identified. There is no possible migration of a node from one level to another.
- Each node that is capable of giving orders needs to know the skills of its slaves and how many it can rely on in order to accomplish the task.
- The task must be (1) decomposable and (2) not sensitive to serialisation since there is no predefined sequencing between the subtasks.

3.1.1.2 State of the Art

There are three types of hierarchical architectures: simple hierarchy, multi-level hierarchy and decentralised hierarchy. The hierarchical architecture will be used in B1.

A *Simple Hierarchy* is essentially a tree with two levels where the top level is composed of a single node and each of the bottom level nodes work on achieving part of the overall task. Control and decision making within such a hierarchy is exclusively located at top level.

The *Multi-Level Hierarchy* model extends the simple hierarchy model by inserting complementary levels thus allowing the control to appear at various levels. Furthermore, communication within a same level is now possible in this model. Nodes from a same level can co-ordinate their effort through negotiation or co-operation. A major difference with the simple hierarchy model is that a single node may have control on the entire next sub-level since there is communication between all the nodes of a level.

A *Decentralised Hierarchy* can be seen as an extension of a multi-level hierarchy in that it has several levels. However there is no interaction between the nodes of the same level. Besides each node in such a hierarchy can be another organisation in itself. One advantage with such a structure is the distribution of control. Each sub-division has access to its resources and chooses its own appropriate organisation; it thus gains more control capabilities. Decision is decentralised within each division. The top level has the most decision power especially for long term operations.

3.1.1.3 In terms of AEIO

After defining what a hierarchical architecture is and describing some examples, we are presenting this structure in comparison to approach AEIO.

We can define agents according to a hierarchical architecture as we find it in [Ferguson 92] and [Muller 94]. This approach says that one agent is a structure with a number of facilities which are stored in a layered architecture. This architecture is put forward to ensure that agents can evolve in dynamic environment and is defined from the subsumption architecture of Brooks.

An agent has layers. Each of which constrains another layer if it is located above. When one event occurs in the environment, each layer receives the event and computes an action. At the end of the computation, only one action is possible. The highest layer that gives rise to an action is the one that will carry it out since it is on top of all the other layers that have proposed partial actions.

This architecture is designed for dynamic or highly dynamic environments. In [Ferguson 92], we have a horizontally layered architecture, so each layer has access to events, and each layer can communicate with each other. This approach is considered by Muller as heavy, because there is a bottleneck for the control since each layer is linked to all other layers.

The solution of Muller is a vertically layered architecture where one and only one layer has access to the environment, and each layer has links with its upper and lower layer (one finds again this approach in the S-R-K architecture and its implementation by [Chaib-Draa 96]).

The second component of the Vowels model is the Interaction one, i.e., the communication between the agents. Interaction does not have any meaning in terms of decomposition in a hierarchical manner. There are no speech acts nor interaction protocols defined as a hierarchy.

An interaction as a hierarchical architecture is only possible with a hierarchy-like Organisation. In that case, there are two kinds of interaction: "order" communication and "inform" communication.

A hierarchical architecture for an organisation exemplifies a form of military command with one commander and several entities, which carry out the tasks given. It is like a master/slave architecture. So, an "order" communication from a master to a slave requests the achievement of a task or constrains somehow the slave's behaviour.

An "Inform" communication from a slave to a master is for giving back information on the task performed.

An organisation can straightforwardly be defined as a hierarchical architecture, like that we can find in human organisations. Here, in the tree-like structure nodes are agents, where inter-node communication is replaced by inter-agent communication. And therefore full negotiation/co-operation can take place.

However, one needs to allow for new agents to dynamically enter the organisation and therefore the multi-agent system. This means one needs to take into account a possible structure reorganisation depending on the task at hand and the number of agents.

Each multi-agent system evolves within an environment. One cannot define an environment with a hierarchical structure unless one considers that each agent has a local view of the environment and it is hierarchically organised. In that case, the environment is a hierarchy of environments, but this alternative has not been approached for building multi-agent systems.

3.1.2 Recursive approach

Recently, theoretical studies in the MAS field have introduced the concept of recursion as decomposition mechanism. This concept has been applied to the MAS structure, the organisation of MAS, planning, interaction protocols ... giving very good results. The objective is to reduce the complexity of MAS modelling. In this section, we will make an attempt to define the recursion concept (in a general computer science framework), then we will describe the main recent research activities on these recursive aspects of MAS and finally we will characterise how recursion can be applied to each of the A, E, I, O concepts used to design a MAS in our approach.

3.1.2.1 Definition

The classical example of recursivity is the factorial computing.

A recursive decomposition provides a way to solve a problem by considering this problem as a set of sub-problems of decreasing complexity. To solve X, building a recursive program is based [Berlioux 83]:

- on the solving of X in certain particular cases.
- on the choice of a decomposition of X in sub-problems of the same nature of X such that nested decompositions always end on a particular case we can solve.

To build a recursive program, we have to decompose the problem and to find a condition to end this decomposition. To ensure the end of a recursive procedure, we have to verify that the chosen decomposition converges, i.e. successive decompositions result always in a particular case we can solve. Sub-problems have to be closer to the solved case than the initial problem. This can be described using a "size" for the initial problem, measured by an integer n (strictly positive), such that sub-problems recursively processed have sizes strictly less than n. Solved particular cases will have by convention the size 1. A recursive decomposition in sub-problems of the same size must be adopted to build efficient algorithms.

3.1.2.2 State of the art

Recursion is an alternative approach to decomposition. The recursive mechanism can supply a way of decomposition for a multi-agent system. In our case, it seems to be an efficient property to reduce complexity of the modelling of a MAS. Recently, some work on MAS focused on the use of recursion as a way of conceptualisation for the elements of a MAS. We classified in the following the main works according to some important criteria such as a recursive description of:

- the **structure** of the MAS (specifying components of the system),
- the **organisation** (as the framework in which agents can co-operate),
- the **knowledge** about the environment, the tasks, the interaction between agents, the system structure,
- the **decision making process** of each agent.

We examine in the following paragraphs these approaches.

Structure

Designing complex systems consists of integrating, distributing and organising many components that are often heterogeneous. For [Occello 97] such systems can be built as a whole in a hierarchical manner, using a recursive approach. He defines a hierarchical structure of decomposition levels of one agent into agents of a lower level. In the functional analysis of a system, components can be identified as sub-systems. In an agent, therefore, the perception/evaluation/ decision/action cycle is applied to each capability that can be viewed as an agent. The encapsulation of a MAS in an agent of a higher level consists in finding functions which represent agents for the encapsulated MAS.

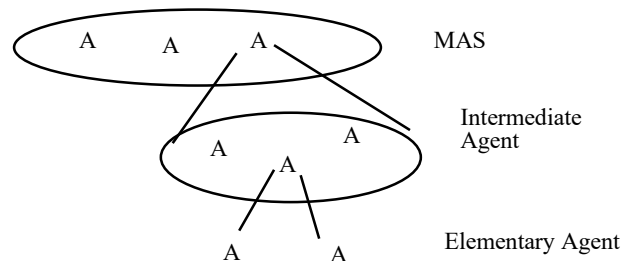


Figure 12: recursive decomposition

Brazier et al. [Brazier 95] propose a formal specification of a component-based architecture for the MAS design. They base their architecture on the tasks hierarchy. Components (modules) are identified tasks. They can be composed or primitive. The decomposition ends when a component is primitive. Each task in the MAS (communication, control ...) can be allocated to one or more agents and one agent can realise one or more tasks.

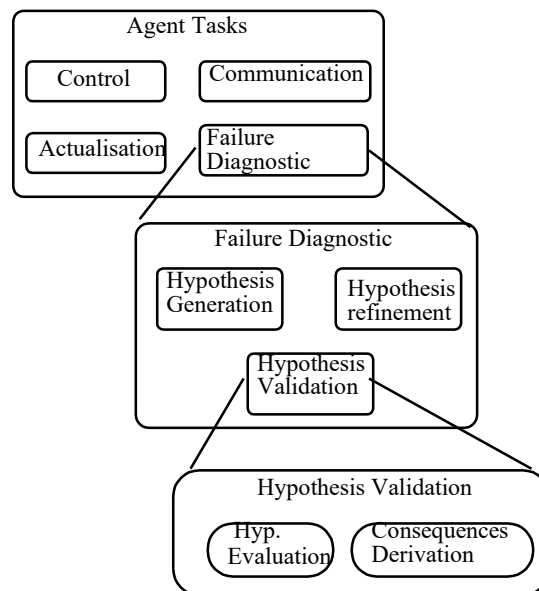


Figure 13: Task hierarchy

Organisation

[Baeijs 98] proposes a grammar for the definition of the organisation in MAS. For him every organisation is named and is composed of groups. Groups can be agents or groups of agents. Here is the recursive grammar:

```

<organisation> := <name><link>(<link>)*
<link> := <member> | <member><weight><member>
<member> := <agentname><mass> | <organisation>

```

This grammar emphasises the recursive property of the structure of an organisation. The weight notion can be viewed as the social distance between two members, i.e. the influence of an agent on an other. The mass permits the representation of the importance an agent can have in a group. We can consider groups on different abstraction levels.

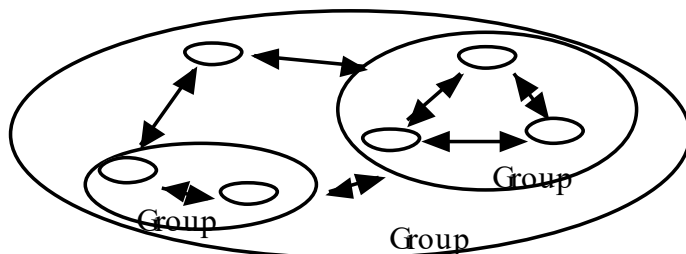


Figure 14: Organisation in recursive approach

Van Aeken and Demazeau have developed the concept of minimal multi-agent system (MMAS) as the simplest model of MAS [Van Aeken 98]. MMAS is a recursive model of a society of agents where the structure of an agent implies the organisation of its sub-agents. The basic element of a MMAS is the atomic agent. In this system all agents are organised in couples of atomic agents. If an agent is added to the society, the result is not two but three agents. The third is generated, it represents the couple. The MMAS is defined recursively as: an atomic agent or an agent composed of two MMAS.

Decision

Gmytrasiewicz and Durfee [Gmytrasiewicz 95] propose a recursive model that allows the agents to take the best decision based on a possible decision of other agents. Each agent builds a payoff matrix in which it can model its own decision in function of probabilities of decisions of the other agents. This is a nested mechanism of decision. The recursion ends when an agent has no more information on the decisions of every other agent. Building the decision matrix can be very expensive because the model increases exponentially according to the number of levels. Vidal and Durfee [Vidal 95] developed an algorithm which ignore parts of the recursive tree to improve the performances.

Durfee and Montgomery [Durfee 90] introduce a hierarchical protocol for agents co-ordination. In order to decide, an agent must know what other agents can do. For this, they propose to build for each agent a tree of anticipated actions of the others. This tree is then "wandered" by each agent in function of the messages received from others in order to evaluate the impact of current actions on the interaction and to take decision on its own actions.

To elaborate plans Elfallah and Haddad [ElFallah 96] proposed a recursive model to represent and model plans with the help of Recursive Petri Networks. The recursive process begins with an action which represents the plan at a higher level of abstraction. The execution of this action can imply the triggering of several methods. Methods can be elementary or abstract. If a method is elementary, it calls a routine executing an action. If it is abstract, it references a sub-plan with a refining corresponding to abstract actions.

In the dynamic environments of MAS, agent monitoring is very important for agent interactions. The agent monitoring observes actions, goals and plans or infers non-observable one. In the context of plane fighting simulation Tambe [Tambe 95] presents the treatment of the proper behaviour of an agent and the monitoring of the other agents as a recursive monitoring.

Knowledge

Considering the components based architecture proposed by Brazier et al. [Brazier 95] for MAS design, it is possible to observe through tasks analysis certain types of knowledge that must be modelled. Knowledge is identified for each of the components and sub-components in the recursive definition of each task at each abstraction level.

Knowledge can be classified as:

- knowledge about the task
- knowledge about the sequence of sub-tasks
- knowledge about the information exchange
- knowledge about the structure and the decomposition
- knowledge about the roles

3.1.2.3 In terms of AEIO

In this section we will define recursion criteria for each of the components of our approach of decomposition of a MAS in Agents, Environment, Interaction, Organisation.

Agents

In the functional analysis of an application, its components can be identified as sub-systems that can be decomposed in the same way. We can consider that the recursive definition of agents implies an agent decomposition into different levels of abstraction; this decomposition allows an external observer of the system to visualise an agent as a part of a MAS at a level n or as MAS at a lower level $n-1$.

We can make a decomposition of the agents in terms of functions, tasks or capabilities. If we address at a first level agents with complex functions or tasks, on a second level these functions can be decomposed into sub-functions on the other levels.

From this notion, it is possible to observe a hierarchical *decomposition* of an agent into MAS.

Otherwise, the *encapsulation* of a MAS from one level into an agent on an other level consists of identifying the functions that represent agents of the encapsulated MAS.

We choose the following definition of the recursion on agents:

- On a given level an agent can be considered as a MAS or as element of a MAS
- Through successive decomposition agents can be viewed as groups of component agents (called intermediate agents).
- On the last level component agents can be viewed as elementary agents.

Thus, on each level of abstraction we can find intermediate agents or elementary agents.

We can define a *Group of Components* of level n as a set of agents of level $n-1$ which possesses capabilities or tasks of the agent of level n in a distributed way.

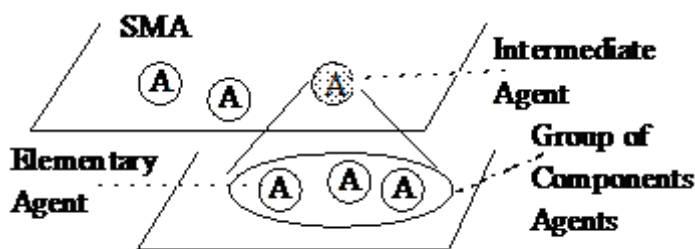


Figure 15: Agents in recursive approach

An elementary agent and an intermediate agent have the same external representation, i.e. an external observer has the same image of the group of components as the image of an elementary agent.

Environment

We can consider that the environment is the representation of the real world in which the agent evolves. As the MAS can be viewed as an intermediate agent (a group of component agents), the decomposition

level (in terms of agents) can define the environment in which agents act. We can say that agents of a given level of abstraction work in the same environment.

We choose the following definition for the recursion of the environment:

The environment of a group of agents on a level $n - 1$ (decomposition in a MAS of an agent of level n) will be defined as the representation of the world perceived by the agent of level n .

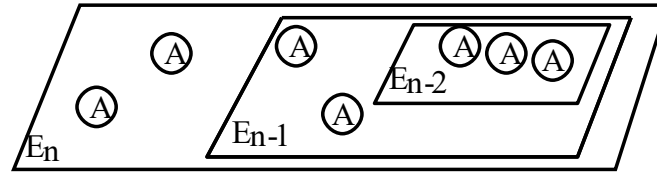


Figure 16: Environment in recursive approach

From one level to another this representation can be partial, filtered, refined or even interpreted in function of the granularity of knowledge used by agents on each level.

Interaction

We consider the interaction as the exchange of information between agents for the purpose of co-operation. This information can be actions, plans, goals, hypotheses or knowledge about the environment.

We can say that it does not make sense to introduce interaction (in the previous sense) between agents and their components since we assume that an intermediate agent and its component group are the same entity viewed from a different level of abstraction. Otherwise, agents on a same level can interact even if they are intermediate agents, or elementary agents, member of a group of component agents.

However communications of a different nature than interaction are defined between intermediate agents and their group of component agents. These internal communication capabilities ensure the transfer of knowledge between levels with eventual adaptations (according to the E and A recursion criteria specifications).

We keep the following definitions for the recursion of interactions:

- Each agent knows if it is elementary or intermediate.
- An intermediate agent knows its component agents.
- Each agent belonging to a group of component agents knows the members of its group.
- Each agent belonging to a group of component agents knows the agent it belongs to.
- There are internal communications between intermediate agents and their components.
- There can be interactions between agents on a given abstraction level.
- Sending a message to an intermediate agent is the same as sending a message to the society of its components

In the case of cognitive agents, agents interact using a protocol which can take the following expression [Demazeau 95] [Populaire 93]:

$\langle \text{interaction} \rangle := \langle \text{communication} \rangle \langle \text{knowledge representation} \rangle$

$\langle \text{communication} \rangle := \langle \text{from} \rangle \langle \text{to} \rangle \langle \text{id} \rangle \langle \text{mode} \rangle \langle \text{via} \rangle$

$\langle \text{knowledge representation} \rangle := \langle \text{multi-agent} \rangle \langle \text{application} \rangle$

$\langle \text{multi-agent} \rangle := \langle \text{type} \rangle \langle \text{strength} \rangle \langle \text{nature} \rangle \langle \text{protocole} \rangle \langle \text{position} \rangle$

The recursive mechanism for the interaction continues until the interaction passes from a group of components to an other. It ends when the interaction arrives at an elementary agent.

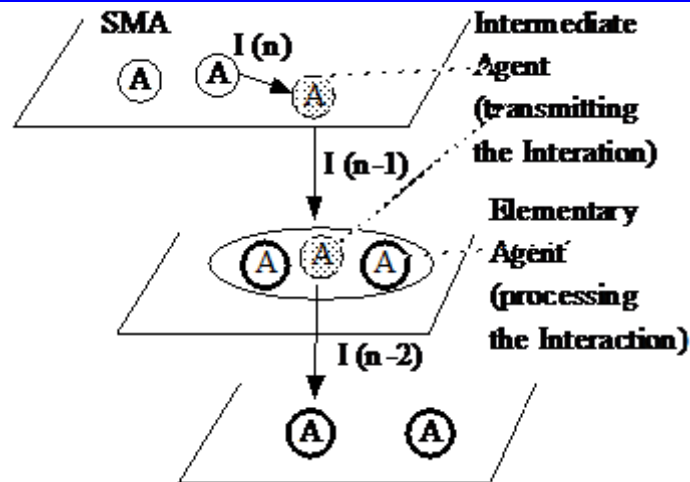


Figure 17: Interaction in recursive approach

Organisation

An organisation can be considered from an external point of view, i.e. the organisation is viewed as external knowledge (global or distributed among agents).

The organisation is expressed through the constitution of groups whose semantics are different from groups of components.

There are different classifications of groups such as simple groups, teams, special interest groups, communities of practice (see deliverable B1).

We maintain the following definitions for the recursion of Organisations:

- The organisational structure can be viewed as a decomposition into groups of agents at several levels of abstraction.
- The MAS is constituted by agents and groups of agents (organisational groups).
- A group is constituted by agents and groups of agents (organisational groups) in the same way.

3.1.3 Hybrid Approach

From a MAS point of view, the hierarchical approach has been extensively studied, not only in MAS but also in Distributed Systems like in any domain where distribution needs to be hardly structured. This intrinsic lack of dynamics as well as the complexity of hierarchies when the number of agents increases leans us to prefer the second approach. At the same time, the new arising recursive approach is very promising but still partially unknown [Occello 97] [Baeijs 98] and possibly not adequate where recursivity is not obvious from a modelling point of view (especially since recursions in MAS applications are usually less than ten steps, and maybe less in some cases), which leans us to prefer the first approach. So, from a pure MAS point of view, there exist advantages and drawbacks in choosing one or the other approach, and since these approaches are complementary, the recommended choice is to mix the two approaches using them when appropriate.

3.2 Identification of the Approach for Generalisation

Two questions are really relevant for organisation in the generalisation process from a geographer's point of view:

- What kind of relationships exist between organisations and agent in the generalisation process, in other words, what are the functions of organisations?

- How to define an organisation?

3.2.1 Geographical agents and Geographical organisations

(This part concerns the kind of interactions that exist between organisations and agents within the generalisation process. We will just give some examples of interactions as an introduction of some concepts that will be developed in report B1 and specified in report B2)

Organisations have three main functions:

Contextual operations

Within the context of generalisation, there is a need for a higher level entity than the basic geographical objects. In order to realise a contextual operations on a group of basic geographical objects, these are grouped into organisational structures (from a multi-agent point of view) and correspond to meso-objects or phenomena from a generalisation point of view. They are either created through the emergence of characteristics at the geographical object level or in an explicit way through the use of spatial relationships or shared semantics (e.g. a town district)

In such a case they are agents. They are the decision makers for an area. According to their goals they choose contextual generalisation functions such as displacement, selection or typification (e.g. area patch generalisation). An organisation which generalises itself is a decider and gives orders to its component agents to execute some order such as delete yourself, displace in such direction, dilate yourself.

Tasks allocation (co-ordination)

In order to optimise the relations between micro-level agents, there is a need for communication and interaction structures, at the same time low level operations have to be sequenced and synchronisation mechanisms have to be included within the problem solving process. These structures are handled through the use of organisational knowledge; to co-ordinate actions between agents when they can not manage alone. Whenever competition for a resource exist such as spatial competition or whenever non compatible goals exist (see chapter 2-3), the organisation can make a choice for its agents. The organisation can give priority to an agent or give some orders for actions. In such a case, the organisation is a mediator or a supervisor for its agents (it depends on the nature of the problem).

Control

Using organisational structures also offers the possibility to have a global view of the agent society and to update goals of groups of agents and use order relationships between them (as it is current practice within the generalisation process). These organisational structures then permit the inclusion of the global constraints (macro-constraints) for a group of agents (or even for an entire class of geographical entities) and maintain the order relationships between meso-agents. The organisation acts to guide the generalisation of agents by giving them contextual information they could not get otherwise. This can be done by changing an agent's goal values to ensure the maintenance of order or similarity relationships between agents. For example a district-organisation can change the size goal of its buildings agent to allow them to preserve their difference. Thus some building agents will have to reach the 300m² value while others will have to reach bigger values. In such a case the organisation is an informant.

3.2.2 Geographical organisation needs

The way to construct organisations from Geo-Agents depends on specific requirement which will not be described hereafter. Nevertheless, mechanisms rely on specific groupings of objects governed by metrics and semantics. For this purpose specific methods such as Minimum Spanning Tree, Delaunay Triangulation, Cluster Analysis and Voronoi Diagram will be used. Construction mechanism are based on an initial filtering of information to build them, on the use of such spatial analysis tools and on the manipulation of others such as by grouping or segmenting. One can find such mechanisms in [Monier 97] [Ruas 98] [Regnauld 98]. The result is the creation of meso object, which is an agent organisation:

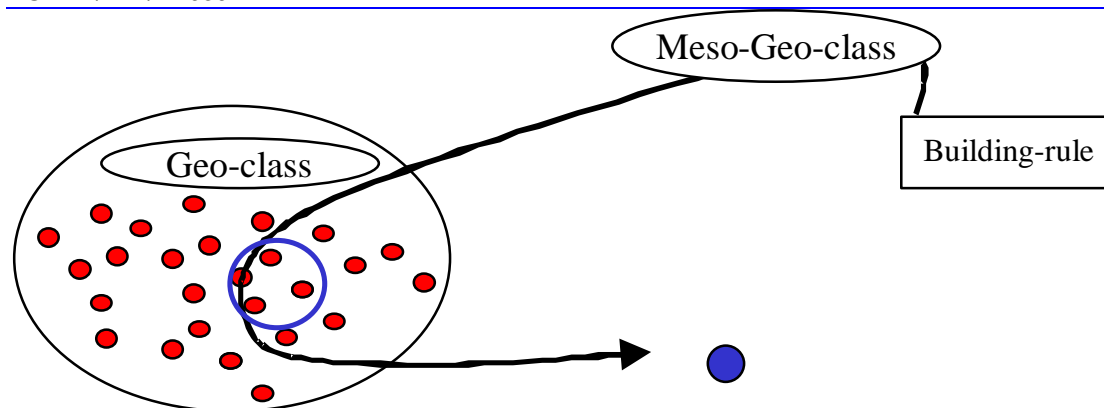


Figure 18: Creation of meso-objects

The construction of an organisation always uses a set of building rules that contain semantic and spatial aspects.

- For *exogene organisations* the rules are predefined, as we know that a specific kind of organisation will be necessary for generalisation purposes. This is the case with districts, which are computed from street partitions which owns buildings inside.

For *endogene organisations* the rules are more adaptive. The existence of an organisation depends on the existence of a specific configuration of agents. The construction is then not systematic but depends on spatial and semantic configurations. The construction of an organisation is mainly driven by constraints. If spatial structure maintenance is given as a constraint at meso level, a mechanism will try to identify such structures (such as houses alignment) and then create an organisation of which goal will be to preserve itself. Such organisations are very useful to preserve gestalt constraints (see A2) by identifying exceptions and invariants that need to be maintained during the process.

Endogene and exogene organisations, as well as their building mechanisms, will be specified in A4 tasks. The concepts of recursion and hierarchy will be compared with the necessary groups of agents for generalisation purposes. It seems that a hierarchical approach should be more frequent than a recursive one.

3.2.3 Choice of an Hybrid approach

The approach proposed from a MAS point of view, meaning, a combination of the two approaches (hierarchical and recursive), usually fits with any requirement. So, we indeed propose that the MAS for generalisation will be organised in several parts that will either hierarchical or recursive. We would like to refer to deliverable B1 for more details on how these two approaches will be combined into one single system, and to see how the actual system modelling will be done.

4 Conclusion

We have introduced the AEIO methodology, developed in the direction of Multi-Agent Oriented Programming, that we will use. This approach identifies agents, environment, interactions, and organisations as the basic bricks of our MAS.

We have decomposed the generalisation problem and identified that the geographical information should be modelled as agents. More specifically, from a geographer's point of view, Geographical entities, or objects, will be the Agents, and we will further call them Geographical agents. They correspond to the micro-level of analysis.

We will follow a flexible and combined approach involving intrinsic and extrinsic decomposition, according to the types of geographical information which are handled at a given time.

The environment will correspond to the geographical database or geographical space.

The interactions will correspond to the generalisation algorithms.

The organisations, restricted to groups, will be used as a control structure in order to ensure co-ordination and co-operation between agents and for the ordering of the geographical object types. More specifically, from a geographer's point of view, Phenomena or a set of geographical objects will be modelled as Organisations. They correspond to the meso-level of analysis.

In addition to this, we have identified that the Agents and the Interactions will be the main actors of the generalisation problem solving.

This report mainly covered the static part (description) of these actors; only a view of their role of co-ordination and communication was given. Report B1 will complete this report by analysing more precisely the functions of the organisations, and the relations between the micro and meso levels.

5 Bibliography

- [Alvares 98] Alvares, L., Menezes, P. & Demazeau, Y. Problem Decomposition: an Essential Step for Multi-Agent Systems, 10th International Conference on Systems Research, Informatics and Cybernetics, ICSRIC'98, Baden-Baden, August 1998.
- [Armstrong 98] Armstrong A., Durfee E. Mixing and Memory: Emergent Cooperation in an Information Marketplace. In ICMAS 98, 3rd International Conference on Multi-Agent Systems. Paris, France, 1998.
- [Baeijs 95] Baeijs, C. & Demazeau, Y. Les organisations dans les systèmes multi-agents, 4^{ème} journée nationale du PRC-IA sur les systèmes multi-agents, Toulouse, 1995. (in French)
- [Baeijs 95b] Baeijs, C., Demazeau, Y. & Alvares, L. Application des Systèmes Multi-Agents à la Généralisation Cartographique, 3^{èmes} Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents, AFCET & AFIA, Chambéry, 1995. (in French)
- [Baeijs 98] Baeijs, C. Fonctionnalité émergente dans une Société d 'Agents Autonomes. Thèse de Doctorat de INPG, Grenoble, 1998 (to appear) (in French)
- [Berlioux 83] Berlioux, P. & Bizard, P. Algorithmique: construction, preuve et évaluation des programmes. Dunod, France, 1983.
- [Berthet 92] Berthet, S., Demazeau, Y. & Boissier, O. Knowing Each Other Better, 11th International Workshop on Distributed Artificial Intelligence, Glen Arbor, 1992.
- [Bijnens 94] Bijnens, S., Joosen, W. & Verbaeten, P. Language Constructs for Co-ordination in Agent Space, 6th MAAMAW workshop, Demazeau, Perram & Müller eds., University of Odense, 1994.
- [Boissier 92] Boissier, O. & Demazeau, Y. A Distributed Artificial Intelligence View on General Purpose Vision Systems, in Decentralized A.I. 3, Demazeau & Werner, eds., North-Holland, Elsevier, Amsterdam, 1992.
- [Boissier 93] Boissier, O. Problèmes du contrôle dans un système intégré de vision, utilisation d'un système multi-agents. PhD Thesis, LIFIA-IMAG, 1993. (in French)
- [Boissier 94a] Boissier, O., Demazeau, Y., Masini, G. & Skaf, H. Une architecture multi-agents pour l'implémentation du bas niveau d'un système de compréhension de scènes, 2^{èmes} Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents, AFCET & AFIA, Voiron, 1994. (in French)
- [Boissier 94b] Boissier, O. & Demazeau, Y. ASIC: An Architecture for Social and Individual Control and its Application to Computer Vision, 6th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Odense, 1994.
- [Boissier 94c] Boissier, O. & Demazeau, Y. MAVI: A Multi-Agent system for Visual Integration, 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Las Vegas, 1994.
- [Bond 88] Bond, A. & Gasser, L. An analysis of problems and research in DAI. A. Bond and Les Gasser , eds, Readings in distributed artificial intelligence. Morgan Kaufmann Publishers, Palo Alto, 1988.
- [Brazier 95] Brazier, F. , Dunin, B. , Jennings, N. & Treur, J. Formal Specification of Multi-Agent Systems: a Real-World Case. In ICMAS 95. 1st International Conference on Multi-Agent Systems. San Francisco, California, USA, 1995.
- [Brauer 98] Brauer W., Weiß G. Multi-Machine Scheduling - A Multi-Agent Learning Approach. In ICMAS 98, 3rd International Conference on Multi-Agent Systems. Paris, France, 1998.

[Burmeister 93] Burmeister, B., Haddadi, A. & Sundermeyer, K. Generic Configurable Cooperation Protocols for Multi-Agent Systems, 5th MAAMAW workshop, Ghedira and Sprumont eds., University of Neuch, tel, 1993.

[Bussmann 98] Bussmann S. Agent-Oriented Programming of Manufacturing Control Tasks. In ICMAS 98, 3rd International Conference on Multi-Agent Systems. Paris, France, 1998.

[Campbell 92] Campbell, J. & d'Inverno, M. Knowledge Interchange Protocols, in Decentralized A.I., Demazeau & Müller, eds., North-Holland, Elsevier, Amsterdam, 1990.

[Cardozo 93] Cardozo, E., Sichman, J., & Demazeau, Y. Using the Active Object Model to Implement Multi-Agent System, 5th IEEE International Conference on Tools with Artificial Intelligence, Boston, 1993.

[Chaib-Draa 96] Chaib-Draa, B. Interaction between agents in routine, familiar and unfamiliar situation, International journal of cooperative information systems, vol. 5, n 1, 1996

[Chang 92] Chang, M. & Woo, C. SNAP: a communication level protocol for negotiations, in Decentralized A.I. 3, Demazeau & Werner, eds., North-Holland, Elsevier, Amsterdam, 1992.

[Demazeau 90] Demazeau, Y. & Müller, J.-P. Decentralized A.I., in Decentralized A.I., Demazeau & Müller, eds., North-Holland, Elsevier, Amsterdam, 1990.

[Demazeau 91a] Demazeau, Y. Co-ordination Patterns in Multi-Agent Worlds: Application to Robotics and Computer Vision, IEE Colloquium on Intelligent Agents, IEE, London, 1991.

[Demazeau 91b] Demazeau, Y. & Müller, J.-P. From Reactive to Intentional Agents, in Decentralized A.I. 2, Demazeau & Müller, eds., North-Holland, Elsevier, Amsterdam, 1991.

[Demazeau 93] Demazeau, Y. La Plate-forme PACO et ses Applications, 2^{ème} Journée Nationale du PRC-IA sur les Systèmes Multi-Agents, PRC-IA, Montpellier, 1993. (in French)

[Demazeau 94a] Demazeau, Y., Boissier, O. & Koning, J.-L. Interaction Protocols in Distributed Artificial Intelligence and their Use to Control Robot Vision Systems, 6th International Conference on Artificial Intelligence and Information-Control Systems, Smolenice, Plander ed., World Scientific Publishing, 1994.

[Demazeau 94b] Demazeau, Y. O. Boissier & Koning, J.-L. Using Interaction Protocols to Control Vision Systems, 1994 IEEE International Conference on Systems, Man and Cybernetics, San Antonio, 1994.

[Demazeau 95] Demazeau, Y. From Interactions to Collective Behavior in Agent-Based Systems. In: 1st European Conference on Cognitive Sciences, St. Malo, France, 1995.

[Demazeau 96] Demazeau, Y., & Rocha Costa, A. Populations and Organizations in Open Multi-Agent Systems", 1st National Symposium on Parallel and Distributed AI, PDAI'96, Hyderabad, July 1996.

[Demazeau 97] Demazeau, Y. Steps towards Multi-Agent Oriented Programming, 1st International Workshop on Multi-Agent Systems, IWMAS 97, Boston, 1997. (slides)

[Demazeau 98a] Demazeau, Y. Preface, ICMAS 98, 3rd International Conference on Multi-Agent Systems, Paris, 1998.

[Demazeau 98b] Demazeau, Y. & Ferber, J. Introduction to Multi-Agent Systems. 3rd International Conference on Multi-Agent Systems, Paris, 1998. (slides) (tutorial)

[Durfee 90] Durfee, E. & Montgomery, T. A Hierarchical Protocol for Coordinating Multiagent Behaviours. In: Eighth National Conference on Artificial Intelligence, 1990.

[ElFallah 96] El Fallah Seghrouchni, A. & Haddad, S. A Recursive Model for Distributed Planning. In: ICMAS 96, 2nd International Conference on Multi-Agent Systems, Kyoto, Japon, 1996.

[Ferber 91] Ferber, J. & Jacopin, E. The framework of Eco-Problem-Solving in Decentralized A.I. 2, Demazeau & Müller, eds, North-Holland, Elsevier, Amsterdam, 1991.

[Ferber 95] Ferber, J. Les systèmes multi-agents. Interéditions, 1995.

[Ferber 98] Ferber J., Gutknecht O. A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems. In ICMAS 98, 3rd International Conference on Multi-Agent Systems. Paris, France, 1998.

[Ferguson 92] Ferguson, I. Toward an architecture for adaptative, rational, mobile agents. In Decentralized Artificial Intelligence 3, Werner and Demazeau, Elsevier, North-Holland, 1992.

[Ferrand 94] Ferrand, N. & Demazeau, Y. Multi-Agents et Multi-Décision en Aménagement du Territoire, 2^{èmes} Journées du PIR Environnement du CNRS, Montpellier, 1994. (in French)

[Finin 94] Finin, T., Fritzson, R., McKay, D. & McEntire, R. KQML as an Agent Communication Language, 3rd International Conference on Information and Knowledge Management, ACM, 1994.

[Fox 81] Fox, M. An organizational view of distributed systems IEEE Trans. Syst. Man and Cybern., SMC-11:70-80, 1981.

[Genesereth 94] Genesereth, M. & Ketchpel, S. Software Agents, Communications of the ACM, vol. 37, 1994.

[Gimenez 98] Gimenez E., Godo L., Rodriguez-Aguilar J., Garcia-Calves P. Designing Bidding Strategies for Trading Agents in Electronic Auctions. In ICMAS 98, 3rd International Conference on Multi-Agent Systems. Paris, France, 1998.

[Gmytrasiewicz 95] Gmytrasiewicz, P. & Durfee, E. A rigorous, operational formalization of recursive modeling. In: ICMAS 95, 1st International Conference on Multi-Agent Systems, San Francisco, California, USA, 1995.

[Hassoun 92] Hassoun, M., Demazeau, Y. & Laugier, C. Motion control for a car-like robot: potential field and multi-agent approaches, IEEE International Conference on Intelligent Robots and Systems, IROS 92, Raleigh, 1992.

[Itoh 98] Itoh F., Ueda T., Ikeda Y. Example-Based Frame Mapping for Heterogeneous Information Agents. In ICMAS 98, 3rd International Conference on Multi-Agent Systems. Paris, France, 1998.

[Koning 95] Koning, J.-L., Demazeau, Y. Esfandiari, B., Quinqueton, J., Interaction entre Agents pour la Supervision de Réseaux en Télécommunications, 3^{èmes} Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents, AFCET & AFIA, Chambéry, 1995.

[Lesser 83] Lesser, V. & Corkill, D. The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving networks. AI Magazine, 1983.

[Malone 87] Malone, T. Modeling co-ordination in organizations and markets. Management Science, 33(10), 1987.

[Malville 98] Malville E., Bourdon T. Task Allocation: A Group Self Design Approach. In ICMAS 98, 3rd International Conference on Multi-Agent Systems. Paris, France, 1998.

[Menezes 96] Menezes P. & Costa J. Systems for System Implementation, Advances in Modeling of Anticipative Systems, Lasker et al. eds, International Institute for Advanced Studies in Systems Research and Cybernetics, Canada, 1996.

[Mesarovic 70] Mesarovic, M., Macko, D. & Takahara, Y. Theory of hierarchical, multi-level systems Academic Press, 1970.

[Monier 97] Monier, P. Caractérisation du relief en vue de son traitement numérique. Application à la généralisation de l'orographie. Thèse de Doctorat de l'Université Louis Pasteur, Strasbourg, 1997 (in French)

[Moulin 96] Moulin, B. & Chaib-Draa, B. An Overview of Distributed Artificial Intelligence. In: Foundations of Distributed Artificial Intelligence, O'Hare and Jennings eds, Wiley, 1996.

[Muller 94] Muller, J., Pischel, M., Thiel, M. Modeling reactive behaviour in vertically layered agent architectures. In Intelligent Agents, ECAI 94 Workshop, M. Wooldridge and N. Jennings, eds, LNAI 890, Springer Verlag, 1994.

[Ocelllo 94] Ocelllo, M. & Demazeau, Y. Building Real Time Agents using Parallel Blackboards and its use for Mobile Robotics, 1994 IEEE International Conference on Systems, Man and Cybernetics, San Antonio, 1994.

[Ocelllo 97] Ocelllo, M. & Demazeau, Y. Vers une approche de conception et de description réursive en univers multi-agents. In Actes des 5^{èmes} Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents, Edition Hermes. La Colle sur Loup, France, 1997. (in French)

[Picault 98] Picault S., Collinot A. Designing Social Cognition Models for MAS through Simulating Primate Societies. In ICMAS 98, 3rd International Conference on Multi-Agent Systems. Paris, France, 1998.

[Pleiad 92] Pùle PLEIAD iVers une Taxinomie du Vocabulaire pour les SystÈmes Multi-Agents, 1Ère JournÈe Nationale du PRC-IA sur les SystÈmes Multi-Agents, PRC-IA, Nancy, 1992. (in French)

[Populaire 93] Populaire, P., Demazeau, Y., Boissier, O. & Sichman, J. iDescription et ImplÉmentation de Protocoles de Communication en Univers Multi-Agents, 1Ères JournÈes Francophones sur l'Intelligence Artificielle DistribuÈe et les SystÈmes Multi-Agents, AFCET & AFIA, Toulouse, 1993. (in French)

[Populaire 93] Populaire, P., Demazeau, Y., Boissier, O. & Sichman, J. Description et implÉmentation de protocoles de communication en univers multi-agents. In: 1Ères Journees Francophones sur l'Intelligence Artificielle DistribuÈe et les SystÈmes Multi-Agents, Toulouse, France, 1993. (in French)

[Regnauld 98] Regnauld, N. Généralisation du bâti: structure spatiale de type graphe et représentation cartographique, Thèse de Doctorat de l'Université de Montpellier, 1998. (in French).

[Ruas 98] Ruas, A. Strategies de généralisation de données géographiques à base d'autonomie et de contraintes, Thèse de Doctorat de l'Université de Marne-La-Vallée, Saint-Mandé, 1998 (to appear) (in French)

[Searle 69] Searle, J. Speech Acts, Cambridge University Press, 1969.

[Shoham 92] Shoham, Y. Agent Oriented Programming, Artificial Intelligence, 1992.

[Sian 91] Sian, S. Adaptation based on Cooperative Learning in Multi-Agent Systems, in Decentralized A.I. 2, Demazeau & M,ller, eds., North-Holland, Elsevier, Amsterdam, 1991.

[Sichman 94] Sichman, J., Conte, R., Demazeau, Y. & Castelfranchi, C. A Social Reasoning Mechanism based on Dependence Networks, 12th European Conference on Artificial Intelligence, ECAI 94, Wiley, Amsterdam, 1994.

[Smith 80] Smith, R. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, IEEE transactions on Computers, Vol. 29, IEEE, 1980.

[StÉfanini 93] StÉfanini, M.-H. & Demazeau, Y. Vers une architecture multi-agents pour le traitement automatique des langues naturelles, 1Ères JournÈes Francophones sur l'Intelligence Artificielle DistribuÈe et les SystÈmes Multi-Agents, AFCET & AFIA, Toulouse, avril 1993. (in French)

[Tambe 95] Tambe, M. Recursive agent and agent-group Tracking in a Real-time, Dynamic Environment. In: ICMAS 95, 1st International Conferences on Multi-Agent Systems, San Francisco, California, USA, 1995.

[Uma 93] Uma, G., Prasad, B. & Kumari, O. Distributed Intelligent Systems, Issues, Perspectives and Approaches. Knowledge based Systems, Vol. 6, No. 2, 1993.

[Van Aeken 98] Van Aeken, F. & Demazeau, Y. Minimal Multi-Agent Systems. In ICMAS 98, 3rd International Conference on Multi-Agent Systems. Paris, France, 1998.

[Van Aeken 99] Van Aeken, F. Minimal Multi-Agent Systems. Thèse de Doctorat de l'INPG, 1999. (to appear) (in French)

[Vanderveken 94] Vanderveken, D. The Logic of Speech Acts, Cambridge University Press, 1994

[Vidal 95] Vidal, J. & Durfee, E. Recursive Agent Modeling using Limited Rationality. In: ICMAS'95, 1st International Conference on Multi-Agent Systems, San Francisco, California, USA, 1995.

[Wegner 90] Wegner, P. Concepts and Paradigms of Object-Oriented Programming, OOPS Messenger, Vol. 1, N. 1, ACM Press, 1990.

[Wooldridge 95] Wooldridge, M. & Jennings, N. eds. Intelligent Agents. ECAI 94 Workshop on Agent Theories, Architectures, and Languages, LNAI 890, Springer Verlag, 1994.