Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Technical
Memo
TM-97-02

Scalability of Multi-Agent Systems

Christian Gerber

February 1997

Deutsches Forschungszentrum für Künstliche Intelligenz
GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Scalability of Multi-Agent Systems

Proposal for a Dissertation

Christian Gerber

DFKI-TM-97-02

This work has been supported by Siemens AG Germany.

ISSN 0946-0071

Scalability of Multi-Agent Systems

Proposal for a Dissertation

Christian Gerber
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
gerber@dfki.de

February 1997

Abstract

This work proposes a generic approach for achieving scalability of multi-agent-systems (MAS). The key to reach that goal is to introduce a self-organization mechanism allowing systems to configure themselves to any application scale and nature. Here, I shall outline such a mechanism, which can be introduced to any multi-agent system. I focus on two systems, InteRRaP and MECCA, and on two applications, the transportation domain, realized using InteRRaP agents, and a traffic telematics application modelled with MECCA agents. All systems and applications are briefly characterized in this work and relations to other fields of research are pointed out

Contents

1 General Overview

The goal of this dissertation is the examination of scalability of multi-agent systems (MAS) and the development of a mechanism which allows self-organization of multi-agent applications, in particular, of applications with a very large number of agents.

1.1 Motivation and Goal of this Thesis

The aspect of scalability is crucial for all kinds of software systems: a system running efficiently in a small environment may fail in a large one if for instance its algorithms' running time is somehow exponential to environment size. For instance, it is well known that programming in the large may differ significantly from small or medium size programming. Hence, a thorough examination of efficiency and problems accompanying programming in the large is inevitable. Unfortunately, in many cases the control flow of very large systems can become highly complex so that it can hardly be pressed into an exact mathematical framework. Using results from traditional complexity theory may not always be appropriate as classification into complexity classes may be of a granularity which is too coarse to be useful. So, another, more empirically oriented approach has to be pursued.
Investigating very large scale multi-agent applications is not only of scientific, but also of commercial interest: Multi-agent software is about to become mature enough to be introduced into the marketplace for many applications. The issue of whether the new software is in fact suited for the demands of the "real world" gains high relevance; even if a system runs perfectly in some test suites, it is not a priori clear that it will perform well in practice:

- Test suites may be built up by idealizing the environment and by making assumptions about it.
- The size of a test bed may be significantly smaller than the environment where the system will be used later on in practice. This may lead to the problems stated above.

This work addresses mainly the latter point: *How can a multi-agent system be organized to make it flexible enough to cope with applications of any scale?* This question raises a second aspect: Not only the size of an application has to be taken into account, but also other properties characterizing its nature. Hence, the goal of this work is to provide a mechanism which automatically adjusts a multi-agent system to any environment. To do so, I will examine scalability from a theoretical point of view as well as from a practical one.

Theoretical Aspects: In my work I will regard the task addressed above as an optimization problem by characterizing a search space and an objective function to be optimized. The objective function has to denote the system's performance while a multi-dimensional search space must describe the set of possible configurations of the system: Each modifiable property of the system reflects one dimension of the search space, ranging from one extreme distinction to the other. Finding a point in this search space which corresponds to an *optimal* configuration (i.e., a configuration with the highest performance) in reasonable time is an intractable task in general. However, finding a high local optimum will also suffice, provided it can be achieved rather easily. Thus, one goal of my thesis will be to derive a suitable optimization mechanism.
The search space dimensions, i.e., scalable parameters of a multi-agent application, can be classified into three, partially interfering groups:

- Scaling the agent society structure
- Scaling communication/cooperation of agents
- Scaling inside the agent model

I will outline these dimensions in some more detail in section 3. Another goal of my work will be to find general rules or heuristics to achieve a good configuration more easily.

Practical Aspects: Empirical experience is a second basis for the envisaged goal: It is intended to evaluate the performance of parameterized multi-agent applications and to use this information as feedback during an optimization process. Therefore, I shall present some very large-scale test suites which will be used for validating this approach.

The goal of this work is not only to derive a mechanism for finding an efficient system configuration, but also to provide a mechanism to *maintain system efficiency*. Usually, multi-agent applications are intended to work ad infinitum: there is no final goal state since the system reacts dynamically on new inputs. This leads to the fact that a system, which was originally adjusted to work at a very high performance level might lose its performance over time as the environment changes. Maintenance of high performance is therefore another main goal of this dissertation: applying some sort of (local) optimality conditions derived from theoretical and practical observations to the current situation of the application will be used to detect suboptimalities. I will outline an architecture layout for a self adapting system.

Having sketched the main goals of my work I will describe in the remainder of this introductory section how this work is classified into fields of research and how it is formally embedded into research environments.

1.2 Classification of this Thesis into Scientific Fields of Research

This thesis addresses problems somewhere in the intersection of a wide range of research areas beside its origin distributed artificial intelligence (DAI). Results of research pursued in other fields may be incorporated, but there may also be the chance that this work will be of some use for these disciplines, for instance for simulating behavior of large human groups.

Distributed Artificial Intelligence Research on multi-agent systems has become an increasingly important area of research in artificial intelligence. The rather young community of researchers in Distributed Artificial Intelligence was founded in the early 1980's. The first international workshop was held in Boston in 1980, while the first larger journal release was a special issue of IEEE Transaction on systems [IEE81] in 1981. It was almost one decade later before the first workshop in DAI was held in Europe (titled "Modeling Autonomous Agents in a Multi-Agent World", and held in 1989.) It took another year before the first German workshop was held.
In artificial intelligence, entities in a distributed environment are called *agents*. Russell and Norvig define in [RN95] an agent as follows:

> An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

This definition is, however, rather vague; agents can be robots, Internet agents, softbots, traffic guiding agents, etc., ranging from primitive entities to sophisticated ones. Research in DAI addresses how the core of "intelligent" agents (e.g. their ability to interact with the environment and other agents, to represent and infer knowledge and to make decisions) has to be modeled in order to construct universal agents which can be used in as many domains as possible.
A second aspect of DAI research focuses on the development of decentralized problem solving approaches by using agents as local decision making and executing units. Although in general global optimality cannot be guaranteed anymore, these methods have been proven useful: Even during the execution of a computed plan it is possible for the system to accept task changes and to adjust the current plan accordingly. Another aspect is robustness: If one participating unit fails another one may take its task, possibly with only small extra effort.

As mentioned above, parallels to other fields of research may be drawn. Therefore, I will shortly present points of contact to these disciplines. Examining them in depth may lead to further insights, for instance heuristics for an efficient system configuration may be found. Section 5 gives some more details on these points of contact.

Management Theory In business organization, consultants have to structure very large companies. Business administration has thus developed basic organizational forms for companies. (A survey can be found in [Wöh81].) These forms only provide a collection of "patterns", so organization managers still have to decide which form to choose for a certain division of the company in question. One point of interest for my work will be to investigate whether rules designed for building company structures can be applied to the modeling of artificial agent societies. If so, one would like to know how they must be modified in order to be applicable.

Distributed Computer Science In recent years much effort has been spent on the study of distributed systems, such as distributed operating systems [Tan95], database systems [AMO93], programming languages [SSR95], etc. Results in these divisions can help to clarify questions such as

- Robustness guarantee: The complete system should not go down if one single participating unit fails.
- Resource adaptiveness: Adding or removing resources to the system should lead to an automated re-balancing of the capacity of the system units.
- Network transparency: The semantics of computations should not depend on the site they are executed on.
- Deadlock handling: Communication or cooperative action between units should be designed in a fashion that always enables at least one participant to proceed with its part of the communication or action.

System Theory and Simulation Theory Results from system theory may be used to introduce a formal model of a whole society of agents. So far, effort has only been spent to characterize *one* agent formally. (An approach can be found in [Mül96a].) However, the behavior of a whole society of agents can hardly be characterized completely, or efficiently, by putting together descriptions of all agents. A more sophisticated model has to take into account group dynamic effects and synergy effects. Therefore, a model which abstracts from the notion

of a single agent, but regards groups of agents as one entity or the whole society as one system may be more appropriate.
Furthermore, if agent behavior is observed in this more abstract framework, features of individual agents may be blurred as characteristics of the whole society become visible, just as sociology abstracts from the psychology of each individual member of a society. Therefore, computer simulation theory might provide methods for describing this kind of behavior. M. Pidd describes in [Pid92] simulation techniques which might be exploited.

Psychology The formation and development of groups is one issue that social psychologists work on. Here much research has been pursued to find the optimal size of a certain group (see for instance [Nas88], [Sch89] or [CML93]). Therefore, it has to be examined in this work whether criteria for an optimal human group may be adapted for artificial agents. Furthermore, researchers in social psychology are interested in reasons why people form or join certain groups. ([TJ77] shows some aspects.) It will be investigated whether such fuzzy patterns of behavior might be of use for modeling decisions of artificial agents to join or leave an organizational unit.

Biology The examination of the sociology of animal societies, in particular of phenomena such as emerging functionality in insect societies, may lead to interesting insights into how a goal for such a society is split into subgoals and how they are achieved. Furthermore, feedback loops regulating the size and structure of insect societies are worth looking at in order to introduce similar mechanisms into artificial societies. Interesting models can be found in [Fre87].

Communication Network Technology Not only the field of computer science deals with the realization of large networks. Other technical disciplines such as electrical engineering or computer engineering have to cope with similar problems. For instance, a telephone network also has to be kept working, even if some of its cells fail or get overloaded. Therefore, techniques have been developed in order to overcome these difficulties. I will examine whether these techniques can be adapted for my work and if so, how they have to be modified. Furthermore, an examination of the world's largest network, the Internet [LR93] will lead to further insights on realization of these issues.

Putting it all together, one major part of the work will be the extraction of fruitful stimuli and results from these related scientific fields. So far I have given a brief overview over research areas this work will be related to. In the next subsection I describe the formal environment into which it is embedded.

1.3 Research Environment for the Intended Work

This work will be carried out in the environments of two German research groups: the project CoMMA-MAPS (Cooperative Man-Machine Architectures: Multi-Agent Planning and Scheduling) supervised by Prof. J. Siekmann and the Siemens project AiV (Agenten im Verkehr; Agents in Transportation).

CoMMA-MAPS This research group is located at the German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) in Saarbrücken. The DFKI is associated with the Universität des Saarlandes, also located in Saarbrücken. In its five-year history, CoMMA-MAPS has developed the multi-agent architecture InteRRaP [Mül96b]. Future goals are to further develop InteRRaP's system architecture towards a more cognitive and social agent architecture by incorporating components such as a more sophisticated knowledge representing and inference component, a learning component, resource bounded planning components, etc.
Furthermore, collaboration with several shipping companies has been established where multi-agent technology is used for optimizing cargo transportation. Here the question of scalability is crucial for a successful application, since such companies own hundreds of trucks; each of which shall be represented by an agent. Therefore, the work of this dissertation will be linked naturally into this project. In particular, the head of the group, Dr. K. Fischer is the adviser for this part of my work.

InteRRaP has been developed in close cooperation with the Programming Systems Lab headed by Prof. G. Smolka at DFKI. This group has developed Oz [SSR95], a concurrent constraint programming language. Using Oz has lead to some computational advantages, since some features of the agent architecture could be realized already at the programming language level.
Furthermore, CoMMA-MAPS has cooperated with other DFKI projects, such as the RAP group [SB95] where planning mechanisms are developed which might be integrated into InteRRaP's agent architecture. Cooperation with the COSMA (Cooperative Schedule Management Agent; [BOH+94]) group has lead to the development of a natural language understanding user interface for cooperation between MAS and user.
Cooperation with the Universität des Saarlandes has also been proven fruitful: the Institute for Information Systems (Institut für Wirtschaftsinformatik; IWI [Sch95]) has set its goals in pursuing application-oriented research in computer-oriented business. New methods in process engineering and work flow management could be evolved by combining their technologies with an agent-oriented approach.

AiV This group is classified as a subdivision of the Central Department of Research and Development (Zentralbereich Forschung und Entwicklung, ZFE) at Siemens, Munich. Here another multi-agent system has been developed: MECCA [Ste92], based on the language MAI^{2}L [SBKL93], also developed in this group through its subsidiary at DFKI Kaiserslautern.
Much emphasis has been put on the communication interface between an artificial agent and a human user (see [Lux95] for details). In particular this approach considers humans to be participants in the multi-agent system. Currently, a great deal of manpower has been invested in a project sponsored by the German Federal Department of Education and Research (Bundesministierium für Bildung und Forschung; BMBF): MOTIV (Mobilität und Transport im intermodalen Verkehr; Mobility and Transportation in Inter-modal Traffic) [Sie95]. This project is a united effort of the major German car companies (e.g. BMW, Daimler-Benz and Volkswagen) and supplier companies (Siemens, Bosch, etc.) to develop a travel planning support system for the next millennium. Agents will be used to represent traffic participants and service suppliers. Again, the question of scalability is most important for the overall success of the project as the envisaged goal of this project is to derive a system to be used (in the best case) by every traffic participant. Working with this project gives me the chance to gain industrial experience which can be used for finding scalability criteria of practical relevance. For this part I will be advised by Dr. D. Steiner and by Dr. B. Specker.

These two bases - DFKI and university research on the one hand and industrial application on the other hand - provide a very fruitful environment for the envisioned goal. Furthermore, using two different multi-agent approaches and different test suites may lead to the broadness and generality neccessary for this dissertation. Finally, being advised by several experts will lead and already has lead to diverse sources of inspiration.

1.4 Structure of this Proposal

Section 2 gives more detailed information on the research area of multi-agent systems in general and of the InteRRaP and MECCA systems, in particular, since they serve as underlying systems for this piece of work. Section 3 represents the core of this proposal: Here a discussion of scalability of multi-agent systems is made and a more formal view of the problem is given. Furthermore, starting points for a scaling approach are discussed: Modifiable system parameters are regarded as dimensions of the search space of that problem. Finally, a sketch of a possible layout of an envisioned dynamically self-adapting system is given. Section 4 describes two very large scale scenarios - one for each proposed MAS. They will be used in the future to gain practical experience on the various scaling possibilities. Furthermore, a possible layout of a unifying agent society structure is given. In section 5 related fields of research are discussed along with an examination of these fields' influences on my work, for instance by motivating system layout heuristics. The final section summarizes the aim of this dissertation and gives a detailed time schedule for both, practical and theoretical goals of this work. In particular, the workload is partitioned and for each piece of work I state how to operationalize it. Possible cooperations with other researchers or students are addressed.

2 Multi-Agent Systems

In this section I give a brief introduction to the research area of multi-agent systems. First general issues will be discussed while the subsequent sections characterize the two systems of concern: InteRRaP and MECCA.

2.1 General Properties

Currently, the term *agent* is used to denote very different kinds of entities, ranging from members in insect societies to various types of technological systems (such as objects in an object-oriented language, but also robots or air planes). Even in the more narrow field of DAI a definition which is generally agreed upon, is not yet in sight. However, many researchers have characterized their individual points of view on how to define an agent.
A description of what an agent is, has been put forward by Wooldridge and Jennings [WJ95]: They characterize an agent by the following traits:

- Autonomy: Agents control their actions and internal states to enable them to operate without the direct intervention of humans or others.
- Social Ability: Agents communicate with other agents by using some kind of agent communication language.
- Reactivity: Agents respond to changes in their environment which they can perceive.
- Pro-Activeness: Agents are able to perform goal-directed behavior in addition to reaction to their environment.

A simple concept of an agent is as e.g. a UNIX process with the above mentioned properties that runs continuously and independently, concurrent to its user's other processes. The agent's effectors would be commands which change the external environment state, and its sensors would be commands providing information.

Many AI Researchers have adopted stronger and more specific definitions of agents, usually in the context of agents they have modeled and/or implemented. P. Cohen and H. Levesque [CL90], define their agent with human mentalistic notions, such as knowledge, belief, intention and goals. They orient this architecture after M. Bratman's well-known requirements of an intelligent agent's mental capabilities [Bra87]. Other AI researchers have even considered emotional agents with human-like mental state (e.g. J. Bates [Bat94]).
A very crucial question a DAI system designer must address is how to model agents and objects in a computational environment. Whereas objects can well be described using an object-oriented programming paradigm, an appropriate definition of agents needs additionally the inclusion of their mentalistic notions. The result of such a treatment is a paradigm called the *agent-oriented programming (AOP)* paradigm due to Y. Shoham [Sho91]. It can be viewed as an extension to the object-oriented programming paradigm.
So far a discussion is going on how to realize that paradigm. In contrast to the paradigm of object-oriented programming, currently no standard has been found. As a consequence, agent models vary greatly in capabilities and functionalities as well as in underlying architecture principles. In the remainder of this section I will shortly describe two systems, the InteRRaP system and the MECCA system.

2.2 The System InteRRaP

A striking feature of the InteRRaP agent architecture [Mül96a] is the combination of reactivity with goal-directed behavior. This is achieved by modeling these two types of behavior in different layers.
In InteRRaP an agent consists of its *World Interface*, where its perception and action is modeled, a *Knowledge Base* (KB) and a three-layered control unit. This unit consists of the *Behavior-Based Layer* (BBL), the *Local Planning Layer* (LPL) and the *Cooperative Planning Layer* (CPL). Figure 1 shows the correlation of these units.
According to these three layers, the agent's knowledge base is partitioned into three units: a *world model*, a *mental model* and a *social model*. The BBL is designed to give the agent the ability to react quickly to exceptional situations and to cope with routine situations. Such reaction is triggered by processing information from its world interface and world model. The purpose of the LPL is to enable the agent to create long-term plans. This is achieved not only by using the agent's world model, but also by its mental models which reflect its intentions and goals. The CPL is responsible for creating so-called *joint plans* with other agents. To do so, the CPL uses not only information on its world model and its mental model; it also processes information about other agents' goals, skills and commitments stored in its social model. Figure 1 visualizes the overall structure

of an InteRRaP agent.

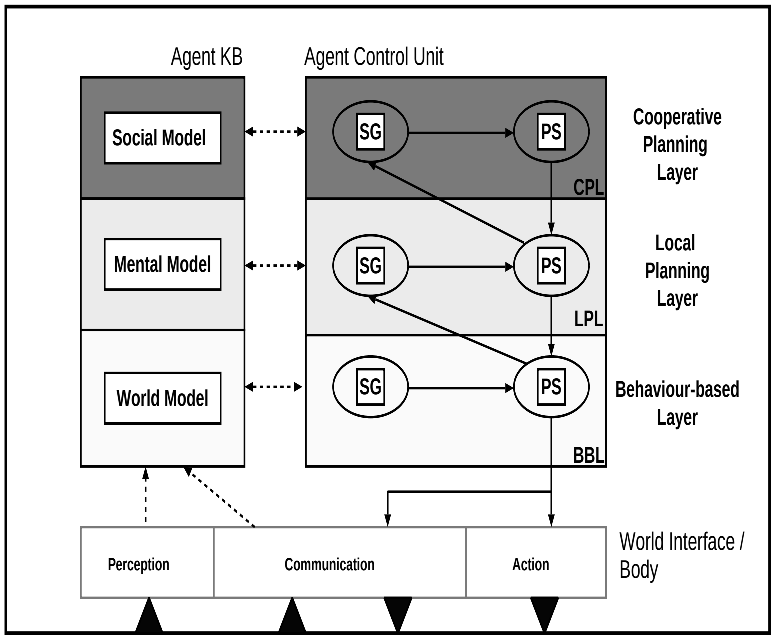

Figure 1: The InteRRaP Agent Architecture

Mental states of an InteRRaP agent are described by the following components: the current *perception* ($\mathcal{P}$) of the agent, a set of *beliefs* ($\mathcal{B}$) describing its informational state, a set of *goals* ($\mathcal{G}$) and a set of *intentions* ($\mathcal{I}$) defining some target states of an agent: If such a state is reached, a given goal is fulfilled. Furthermore, in InteRRaP intentions define which *action* is performed next. Using these concepts, three basic functions for updating the mental states can be defined:

- *BR:* $\mathcal{P} \times \mathcal{B} \mapsto \mathcal{B}$ is a *belief revision and knowledge abstraction function*, deriving new beliefs from perception and old beliefs.
- *SG:* $\mathcal{B} \times \mathcal{G} \mapsto \mathcal{G}$ is a *situation recognition and goal activation function*, mapping an agent's goals and beliefs into a new set of goals.
- *PS*: $\mathcal{B} \times \mathcal{G} \times \mathcal{I} \mapsto \mathcal{I}$ is a *planning and scheduling function*, deriving new intentions from current beliefs, goals and intentions.

Figure 1 shows the flow of control inside the model. The functions *SG* and *PS*, evaluated in a certain layer, influence activities on that layer as well as on neighboring layers: The *situation recognition and goal activation* process results in the creation of new goals. These goals may trigger *planning and scheduling* processes on the same layer. Such a process generates new intentions by planning the steps which have to be taken to achieve a focused goal. After execution of these steps, the situation and goals relevant for the *SG* function of the next upper layer may have changed. Hence, the *SG* function on that level has to be executed. Applying these techniques leads to two basic flows of control:

- Upward Activation Requests: If a particular layer is not competent for solving a task (i.e., it cannot find a plan or schedule to achieve a given goal) it sends a request to the next upper layer. This layer may be able to solve the task since it has more access to the knowledge base and has keener planning facilities. The upper layer reports the result of its *PS* process to the lower one.

- Downward Commitment Posting: Whereas activation of layers is done bottom-up, acting is organized in a top-down fashion: partial plans, derived at a given layer are posted to the underlying layer which has to integrate them into its schedule.

This leads to several problem-solving strategies: In a *reactive path* the situation is recognized in the BBL and directly addressed in that layer (example: avoiding a collision of two agents). A *local planning path* denotes a situation where a solution could not be found in the BBL, so that the LPL has to be activated in order to find a solution executed in the LPL (example: planning a transportation order). Finally, in a *cooperative planning path*, first the LPL had to be addressed by the BBL and the CPL had to be activated by the LPL. Once a joint plan has been found it is processed in the LPL into a local plan which is then executed in the BBL (example: negotiation on a joint plan solving a blocking conflict).
The next section gives a brief overview of the second MAS of interest, MECCA whose capabilities concentrate on a different issue.

2.3 The System MECCA

One major focus in the development of MECCA [Ste92] was to create a system which integrates human and artificial agents. Such an approach is classified under the paradigm of *Human Computer Cooperative Work (HCCW).* The realization of this paradigm has lead to unique features in MECCA's system architecture.
A main trait of the HCCW paradigm is to regard human and artificial agents as entities of equal rights. Thus, the main challenge to such a unifying system is to provide adequate user interfaces and powerful and flexible cooperation methods. These requirements have lead to the MECCA system architecture: A MECCA agent consists of three elements: The *body*, the *head* and the *communicator.* The body of an agent is used to model its basic problem solving capabilities.

These capabilities are application-dependent and can be pursued by the agent without integration into any cooperation structure. The head enables an agent to participate in cooperative processes. It plans at a high level of abstraction and interacts and negotiates with other agents. In order to communicate with other agents an agent needs to have access to communication channels and information about other agents, such as their addresses, etc. The communicator provides these channels and information to the agent's head.
In contrast to the InteRRaP model, where reactivity is a possible kind of behavior, a MECCA agent behaves strictly goal-oriented; problem-solving is realized in a four-step manner, visualized in figure 2.

- Initialization: During this step, a goal is established for one or several agents. Negotiation between agents may be neccessary to achieve this goal.
- Planning: During the planning phase, alternative action sequences are developed and evaluated in order to determine the optimal one.
- Execution: While performing and monitoring the planned actions they are monitored in order to recognize differences between the expected and current outcomes.
- Evaluation: Evaluating the result of a plan execution helps to determine weaknesses of current plans and this may lead to new goals.

As mentioned above, one major focus of MECCA is to provide an integrating system for both, human and artificial agents. Hence, a sophisticated *cooperation model* is required. Based on speech act theory [Aus62], MECCA's cooperation model consists of two levels: *cooperation primitives* and *cooperation methods.*
Cooperation primitives are structured messages sent from one agent to another. A cooperation primitive consists of a keyword and a message contents where the keyword defines how to interpret the contents. Examples of keywords are `propose`, `accept`, `reject` or `refine`. Primitives are treated by the agents' planning component as actions, i.e., their semantics can be described by preconditions and effects. This enables the planner to reason about communication with other agents.
Cooperation methods are built up by composing cooperation primitives to larger pieces. Methods then serve as well-defined communication protocols between agents. Using cooperation primitives allows to model well-known cooperation methods such as *master-slave cooperation*, *contract net cooperation*, *negotiation* etc. In MECCA, pre-built cooperation methods are stored in a library and may be activated by any agent. Furthermore, it is possible for human agents to define new cooperation methods dynamically. This ensures the flexibility required for a successful integration of human agents.

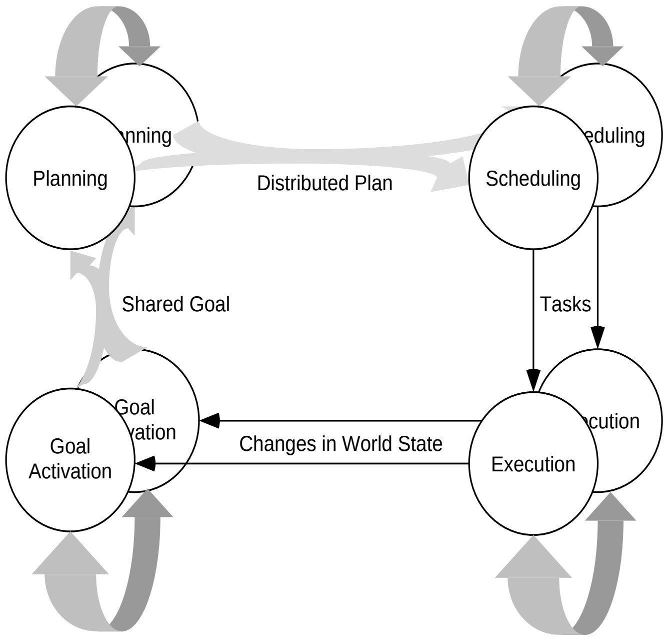

Figure 2: Problem solving with MECCA

3 Theoretical Aspects of Scalability

This section presents the core of the proposal: theoretical aspects of scalability, which is one focus of the intended dissertation. First, I will try to characterize the term "scalability" and its correlation to very large scale multi-agent applications in order to present a more formal description of the problem. In the following subsections I will discuss various scaling opportunities of a multi-agent application, and finally, I will sketch a layout of a future self-organizing system.

3.1 Definition

In [BB89], a scale is characterized as follows:

A classification curve to assign numbers to certain entities.

The authors of [BB89] distinguish between three different types of scales:

- A *nominal scale* is used to simply distinguish between entities by assigning different numbers to them.
- An *ordinal scale* is a nominal scale where an order is introduced between the numbers representing entities. This order allows to sort entities with respect to a certain property.

- A *cardinal scale* is an ordinal scale where additionally a metric is introduced which allows to determine *how far* two entities differ according to the property in question.

Furthermore, the authors distinguish between *discrete* and *continuous* scaling dimensions: In a discrete dimension an entity can only be assigned to a value out of a discrete set of numbers whereas in a continuous dimension all possible values between two extreme points may be assigned.
Thus, the concept *scalability* denotes the possiblity to exactly up- and down-size an object. In the field of software systems the degree of scalability of a system architecture can be used to describe how its problem solving behavior reacts on resource modifications.[1] This behavior can be measured by the introduction of a *performance function.*[2] One may achieve *optimal scalability* if performance is directly proportional to the use of resources. However, optimal scalability can hardly be achieved in complex applications: in general, scaling up one or several resource quantities by some factor n does not imply a performance improvement by the factor $c * n$ for some constant c.
However, investigations can be made to discover an *optimal resource distribution* which can be viewed as the optimum of the performance function in an m-dimensional search space where m denotes the number of scalable quantities and each point in that search space corresponds to one particular system configuration: Each scalable quantity (a list will be presented in subsequent subsections) is represented by a dimension in the search space, whose domain ranges from one extreme point of that quantity to the other.
Figure 3 shows a simple, two-dimensional example: here performance depends only on the number of agents and the usage of a sophisticated knowledge representation (KR) component: Whereas the agent number is represented by a discrete dimension (whose domain ranges from 0 to possibly infinity) the other dimension is modeled as a continuous one, its domain ranging from 0 to 1: a number between those extremes indicates how much percentage of an agent's

[1]In this work I refer the notion of a *resource* mainly to computation time and space; other resources, for instance physical resources of some real-existing robot, have to be considered in the future.

[2]This function has to be defined for each application from scratch since it depends on several factors such as operating time, quality of the result, etc. The application designer has to combine them by e.g. building a weighted sum. Designing this performance function depends strongly on the preference structure of the application designer. Hence, general claims can hardly be made about it.

computational time can be used in the KR component.[3] [4]

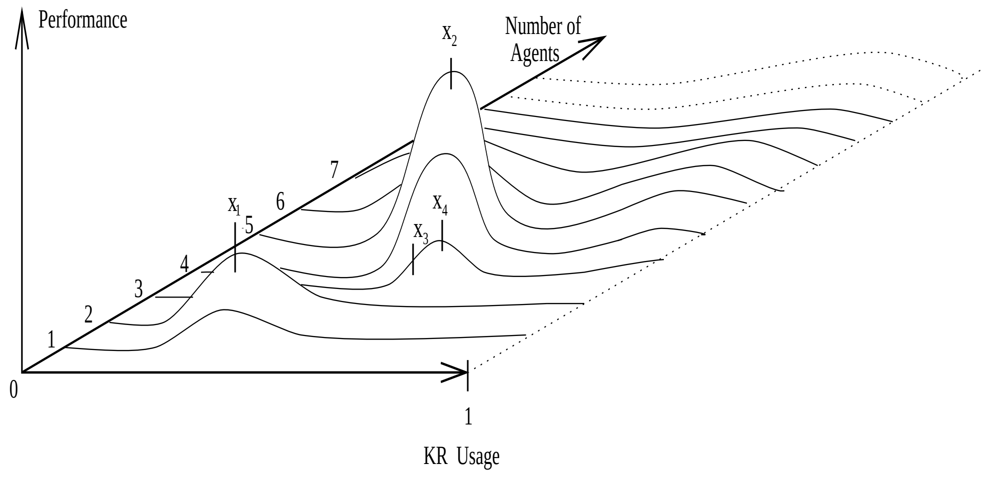

Figure 3: A Simple Example of a System Performance Relation

Following the methodology of operations research (OR), the problem can be characterized formally as follows:

max	Overall Performance
subject to	Distribution of resources on scalability quantities
	Minimal and maximal boundaries for the use of each resource
	Maximal boundary for the overall use of resources

Finding a global optimum is a very hard problem whose solution can hardly (if at all) be found in reasonable time. Furthermore, a *high enough local optimum* will do as well considering the fact that the list of tasks the system has to perform, may change rapidly: current configurations may lose high performance and hence, have to be discarded quite frequently.
The point of concern is to find for each quantity a value with a high *marginal utility value*, i.e., a point where the investment for one more resource unit leads to a high performance *improvement*. By making assumptions, which I will discuss below, a solution may be achieved by applying a variation of optimization procedures well-known in OR or AI.

[3]For the sake of simplicity, this dimension is scaled equally for all agents; in future applications it may well be scaled separately for each agent. This, of course, can lead to an explosion of the number of scaling dimensions. I will discuss later how to address this problem.

[4]Another way to scale a KR component would result in a discrete dimension, provided, several KR algorithms of possibly different complexity are at the agent's disposal: every point in that dimension would denote the usage of a certain combination of these algorithms. An order can be introduced by comparing the complexity of these combinations.

For instance, a modification of G. Zoutendijk's steepest ascent method [Zou76] may be used: An optimum configuration (i.e., an optimal point in the search space) can be found by moving from some arbitrary starting point in the search space iteratively to the optimal point. In every step, first the direction is determined on which to move. In case the relation of performance towards resources can be expressed by some differentiable function, the best direction can be determined using partial differentials. Otherwise the direction has to be determined empirically. One way to do so is to restrict possible directions by moving only along the given dimensions: iteratively a *marginal step*[5] in both directions of each dimension is made, the performance difference is measured, and the step is undone. Then a step in the direction with the highest performance gain will be made. If no performance gain could be found, the algorithm has reached an optimum.
In the example, a starting point may be x_3. A maximal performance gain will be achieved by adding more resources to the KR component: x_4 may be achieved. By increasing the number of agents to five, the optimal configuration (represented by x_2) will be found.
As stated above, some assumptions have to be made in order to use that simple algorithm:

- *The performance function P is concave*, i.e., for arbitrary points x and y in the search space and for $0 \leq \lambda \leq 1 : P(\lambda x + (1 - \lambda)y) \geq \lambda P(x) + (1 - \lambda)P(y)$. If this assumption is violated, the algorithm may run to a local optimum: for instance in the previous example, by beginning from another starting point the algorithm might return x_1 as a result. Solutions to such dilemmas are provided by algorithms such as simulated annealing [KGV83], where random jumps are made from time to time. Although, in general, this approach cannot guarantee to find an optimal solution, in practice it has proven to be efficient. It has to be investigated how to model such techniques for my purpose.

- *During the search for optimization neither the search space changes, nor does the dependence of the performance function to scalable or non-scalable quantities.* This assumption cannot hold in general, as e.g. new tasks may be incorporated anytime which may change for instance the optimal number of agents. Furthermore, such changes may occur already during the

[5]If the search space is not *convex* (i.e., all possible points between two arbitrary points in the search space do also belong to the search space), a "marginal step" denotes the smallest step possible to reach again a point in the search space. For instance, in the above example the discrete dimension "Number of Agents" induces non-convexity. A marginal step along that dimension can be achieved by increasing or decreasing the agent number by one.

computation of the next step to perform in the search space. I will propose in section 3.5 how to cope with this problem.

- *Performance impacts of the resource quantities are independent from each other.* Again, this is an assumption which doesn't hold in the general case. For example, it is easily possible that making knowledge base reasoning more sophisticated will also influence the performance impact of a learning component: This component may work more effectively if powerful inference mechanisms are provided. Generally speaking, interaction of these quantities have to be examined. Still, assuming no interdependence between these quantities can be used to find an approximated solution. Further research has to be performed to decide whether such an non-interdependence assumption can be made without too much loss of generality.

- *All dimensions are scaled cardinally or at least ordinally*, i.e., a partial order can be introduced. This assumption is crucial for the algorithm above to decide in which direction to move next. However, it also cannot hold in general; for instance, only a nominal scale was introduced at the second possibility to define the dimension of KR component usage. In such cases an order has to be introduced artificially (e.g., by using some heuristics) such that not every single point of that dimension has to be tested.

This procedure is of course not the only applicable method. Another promising idea is to use some variation of J. Holland's Genetic Algorithms approach [Hol75]. However, I will have to see in detail if (and how) this method can be modified for my purposes.
It may not be reasonable to introduce only one such optimization procedure as too many scaling dimensions may occur. For instance, as noted above, the usage of a KR component may as well be scaled for each agent individually which would lead to an additional scaling dimension for each agent running in the system. Therefore, it may be wiser to distinguish between scaling dimensions which characterize general system properties (which I call *global dimensions*), dimensions characterizing smaller groups of agents (*local dimensions*) and dimensions characterizing one particular agent (*individual dimensions*). Hence, one optimization process for global dimensions, several processes for local dimensions have to be performed as well as a process for each agent, addressing individual dimensions. Distinguishing between these three dimension types leads to some sort of abstraction hierarchy.

Three different architecture levels can be distinguished for scalability examination: The *agent society level*, the *communication level* and the *agent model level*. In the following paragraphs of the next three subsections I briefly characterize

each scalable quantity by describing how it can be modeled to a dimension in the search space: The search space will be characterized as nominal, ordinal or cardinal, as discrete or continuous, and as global, local or individual. If possible, two extreme points on the domain of that dimension will be marked.
Note that scaling on the society level is the most relevant issue; it influences scalability on the communication level. Scaling on the agent model level is of the lowest interest for general examination as most of the factors to be scaled may vary greatly from one MAS architecture to another.[6]

3.2 Scaling Agent Societies

In this section I discuss various possibilities of scaling an agent society. The dimensions mentioned below interact highly. Hence, a separated scalability examination may not be reasonable.

Number of Agents This quantity is perhaps the most obvious: the more resources are available, the more agents may be added to the system. However, assigning too many agents to a certain task may lead to communication overheads and hence, this may actually decrease performance.[7] Furthermore, impact on performance is greatly dependent on other quantities such as listed below.
As shown in the example, the corresponding dimension is discrete and cardinal; its domain ranges from 1 to infinity. This dimension can be addressed globally or locally for a certain substructure which gets defined by the following dimension modeling the society structure.

Organizational Forms Similar to organizational forms built up in business or nature (which I will discuss in section 5.1), very large scale multi-agent applications need some form of organizational structure. This structure can be used to determine communication and cooperation partners, it may define a command hierarchy, etc. Several different models are available by applying organizational forms presented in section 5.1 to the MAS domain. In the big picture, the organizational structure of a multi-agent application determines a degree of decentralization since centralized and more decentralized organizational forms are available.
In principle, each possible combination of organizational forms may be represented as one point in a discrete and nominal dimension. As this leads to exponentially many points, it may be more reasonable to represent this quantity by

[6]However, this is an important issue for the further development of the InteRRaP architecture and will be included into the work schedule of CoMMA-MAPS for the next three years.

[7]This corresponds to a quote by F. Brooks [FPB75] on up-sizing groups of human agents:“Adding more manpower to a late project makes it even more late.”

several dimensions: a global dimension describes the overall structure; units in that structure may be single agents or substructures which are represented on their own by local dimensions.
In addition, research has to be pursued in order to find a similarity relation which serves as an order so that the dimensions are at least ordinal.

Agent Specialization Heterogeneous societies (i.e., societies with different agents) may have wider ranges of expertise at their disposal which may enable them to perform better; on the other hand, such a society may tend to be susceptible to failure since a specialist may not be replaced easily.
A scaling dimension introduced to model skill distribution defines how scaling dimensions at the agent architecture level (which will be presented in section 3.4) are addressed: globally - all agents are treated equally; locally - only agents belonging to the same substructure are treated equally; or individually - the scaling dimensions on the agent architecture are used for the optimization process of each agent on its own. Therefore, the agent specialization dimension can be treated as a global, discrete and ordinal dimension: {*global ; local ; individual*}.

These issues have a strong implicit influence on the now following items; however, they can also be scaled explicitly.

Migration In a traditional approach, communication between geographically distributed agents is performed over the network. This can be rather time consuming in case the network is heavily loaded and the communication process consists of some complex negotiation procedures. In the *migration approach* an agent is transmitted over the net in order to communicate with its partner on the local server. This can be fast if the receiving server has an accurate model of the traveling agent. In this case, only data describing the agent's mental states have to be transmitted; a copy of the agent will be generated locally.
A scaling dimension can be realized in various ways: For instance, one might define a threshold determining a minimal *distance* (represented by e.g. signal passing time) between two agents to allow migration. Such a realization would lead to a continuous and cardinal dimension which may be optimized globally or locally. Its domain ranges from 0 to infinity.

Explicit Resource Control *Decentralized control* of resources (where agents have to trade on resources) may lead to efficient resource distribution, on the other hand, quite some communication overhead may occur. A *centralized resource control* (e.g. an agent which administrates memory and CPU time) may not requires so much communication, but there is the danger that the adminis-

trator may misjudge the importance of some agent activity, which may lead to an inefficient resource distribution.
This scaling quantity can be modeled by two dimensions: first it has to be determined if (and how much) explicit resource control shall be used at all. If so, a second dimension would characterize the trade-off between centralized and decentralized control. Both dimensions may be realized continuously, cardinally (their domains ranging from 0 to 1: a number in this interval denotes how much of the resources will be controlled explicitly or centrally resp.) and either globally or locally.

3.3 Scaling Communication/Cooperation

In this section I discuss scaling approaches which cope with agent communication and cooperation. The quantities presented here depend highly on quantities mentioned in section 3.2. In particular, it is assumed that the agent society is structured into units by some sort of organizational form.

Communication/Cooperation in and between Units Depending on the organizational form, communication and cooperation within a certain unit may be restricted strongly or weakly: For instance, in a hierarchical structure members of a unit may only communicate with the head of the unit; in other organizational forms communication between unit members may also be allowed. Furthermore, communication and cooperation *between units* has to be specified: It has to be determined whether all unit members may communicate with neighbor unit members or only selected agents (e.g. the head of the units) may do so.
Communication between units may be modeled either as a global or local dimension, depending on the overall size of the application; communication inside a unit may be best modeled in a local manner. Both quantities may be realized as discrete, nominal dimensions, where each possible communication type denotes one point in that dimension. However, it may be possible to introduce an order which leads to ordinal dimensions.

Communication/Cooperation Through Official or Unofficial Channels
Using only official channels leads to communication and cooperation induced by the organizational form. However, it is well possible to introduce direct ways of communication between distant units in order to facilitate cooperation.
Representing this quantity can be done by defining only a limited number of "unofficial communication channels" which serve as shortcuts: By counting communication acts between agents a ranking can be set up: A direct communication channel will be introduced between agents that communicate often with one another. The number of channels can then be scaled. This induces a discrete,

ordinal dimension, whose domain ranges from 0 to infinity. This dimension can be realized locally or globally.

Complexity of the Communication Process Communication between artificial agents is usually performed in a structured form, e.g. by using speech act protocols. However, several protocol types are possible, ranging from fast master-slave communication over various types of auctions to complex negotiation protocols which may be quite time consuming if applied widely.
This quantity can be modeled as a discrete, ordinal dimension (each communication type is represented by one point in that dimension) which will be used in either global or local optimization.

Communication and cooperation between equally or differently ranked agents The form of the organization may define a ranking between agents. This ranking influences the communication pattern: Whereas a higher ranked agent may order actions or request information from lower ranked agents equally ranked agents may only cooperate if it is in both agents' interest.
A discrete and nominal dimension may be modeled either globally or locally: Each point in that dimension denotes one possible combination of communication type and ranking relation. Again, it has to be examined how to introduce an order.

3.4 Scaling Inside the Agent Model

Finally, I discuss scaling possibilities at the agent architecture level. As already mentioned, it will be hard to find general properties which hold for any multi-agent system since these systems may vary greatly. For instance, the usage of a learning component can only be scaled if an agent actually has such a component at its disposal. During my work I will therefore focus on the InteRRaP agent architecture since it is going to be extended by modules presented in the subsequent paragraphs.
The issue whether these dimensions shall be treated globally, locally or individually has already been discussed in section 3.2 where the scaling dimension "Agent Specialization" has been defined in order to let the optimization algorithm decide how to treat the quantities below.

Explicit Resource Control Similar to the situation on the agent society level, explicit control over resource (mainly memory and CPU time) distribution can be scaled on the agent architecture level: On the one hand, there may be a central module inside the agent that controls and distributes resources in a centralized fashion, on the other hand, different modules may compete for resource usage.

Similar to administering explicit resource control for agent societies, this quantity can be realized by two continuous and cardinal dimensions whose domains range form 0 to 1: a value on the first dimension denotes what percentage of resources should be controlled at all; a value on the second dimension describes how much of that percentage shall be administered centrally.

All of the following scaling quantities can be realized as continuous and cardinal dimensions: a number between 0 and 1 denotes a percentage of resources allocated centrally to that quantity. A second realization may be possible if *a priori* an explicit collection of sophisticated algorithms for the various modules are at the agent's disposal: A scaling dimension then can be realized discretely where each point denotes a combination of algorithms to be used. However, some examination has to be performed to find an order to make these dimensions ordinal. In the following, I give a brief description of the scaling dimensions.

Complexity of Knowledge Representation and Inference Capabilities Efficient integration of these capabilities is a crucial aspect of an agent design. A powerful inference mechanism enables the agent to draw logical conclusions. However, complexity theory has shown that powerful reasoning algorithms easily become intractable. A scaling process has to determine when the usage of an inference component is reasonable.

Usage of Learning Algorithms Learning enables an agent to improve its performance by using experience gained from previously performances of similar tasks. Again, a trade-off has to be taken into account, since the learning process itself can lead to a severe overhead.

Usage of Resource-Bounded Algorithms and Anytime Planning Algorithms These algorithms produce results for planning tasks which are suboptimal if time is limited, but can be improved to more and more optimal solutions if further processing time is available.

Complexity of Perception Functionality If an agent has a perception module at its disposal, its performance can be influenced by introducing sophisticated methods which allow the agent not only to perceive passively its environment, but also to focus actively on certain circumstances. If such a functionality is given the question has to be addressed on how extensively it should be used.

Complexity of Communication between Modules As the various modules of an agent have to cooperate to achieve an agent's goals, communication between modules can be scaled as well. However, in many agent architectures communication channels between modules are defined directly as a part of the architecture. Thus, scaling opportunities are rather limited.

To summarize, I treat “Agent specialization” and the overall “Organizational Form” as global dimensions. “Number of Agents”, “Organizational Form” of subunits, “Migration”, “Explicit Resource Control” and all communication and cooperation scaling quantities may be modeled as global or local dimensions: A final decision on how to exactly model these quantities can only be made after some experimentation. Eventually, scaling dimensions addressing the agent model will be treated according to the “Agent Specialization” dimension.
Scaling dimensions on the society level, especially global ones, are of the highest relevance, as they influence communication dimensions and local dimensions. General claims about scaling on the agent model level can hardly be made as they rely heavily on the agent architecture in use. Hence, such scaling quantities are of little interest for *general* issues.
Note, this enumeration is not claimed to be complete, it is subject to change. As mentioned above, it is a goal of this dissertation to use information gained by examining scaling quantities in order to derive a self-organizing system. The next section will shortly sketch some aspects of such a system.

3.5 A Possible Layout of a Self-adapting System

Realizing an efficient scenario layout consists of two tasks: First, it has to be built up and second, it has to be maintained on a high performance level.
Building up such a complex scenario from scratch is a difficult task - manually as well as automated; achieving high performance instantly is almost impossible. Therefore, the idea is to generate a first, unoptimized version which then can be optimized iteratively. A further goal of this thesis is to create a framework which allows the system to optimize the structure of an application to the requirements of the current environment.
In this work I distinguish between the *virtual* and the *actual* structure of a multi-agent application: The *actual structure* is characterized by the number of agents, their grouping, communication, architecture, etc., basically by all quantities mentioned in sections 3.2, 3.3 and 3.4. The *virtual structure*, on the other hand, describes the view a user has of the system. This structure should be specified by the scenario designer and then left fixed. The actual structure and the mechanism by which the virtual structure is mapped to the actual structure should not be visible to the user, just as in a distributed environment the actual distribution of a computation is hidden from the user. (This paradigm is called *information hiding.*)

The distinction between virtual and actual structure has the following advantage: An application designer only modeling the virtual structure needs not to worry about bottlenecks in his structure as it is transfered to an optimized actual structure in the background. For instance, suppose, an agent A has to perform some huge workload of communication which prohibits it to do its other tasks, the optimization framework may add an additional agent to the system supporting agent A. However, this modification is irrelevant for the user and therefore should be hidden in the actual structure.
On the other hand, in some cases, the system designer may wish to represent some entities in the environment explicitly by agents. (e.g. in a shipping scenario each existing transportation device might be represented by one agent.) In such cases, agents in the virtual structure have to be modeled in the actual structure in a one-by-one fashion. Hence, the application designer must be able to distinguish *named* agents, substructures, etc., which must not be changed in the actual structure, from *anonymous* agents, substructures, etc. which may be represented differently in the actual structure.
The distinction between a virtual and an actual layout of an application allows the following approach which can be viewed as an anytime method:

1. The application designer models the virtual structure of the application.

2. This virtual structure is used as a first step for the actual structure.

3. The system detects bottlenecks in the actual structure and modifies it in order to remove these bottlenecks. This step may be performed in interaction with some human expert.

   Furthermore, the system recognizes server and agent failure and performs patching actions.

4. The third step will be repeated until the system is shut down.

This problem solving paradigm is usually called *local search*. For the realization of this algorithm two issues have to be addressed: *How to detect bottlenecks* and *how to monitor the actual structure.*
Bottlenecks will be recognized by applying *(local) optimality conditions* on the layout. These conditions will be retrieved through examination of scalability quantities described above in combination with performance evaluation. Furthermore, examination of related fields of research may lead to further insights: Some first approaches can be found in section 5: Organizational forms defined in organization theory can be used to model substructures of the layout; heuristics about the perfect size of a cooperating team may be taken from psychological group theory, etc. If a qualitative evaluation of bottlenecks is possible (i.e., estimations are available how far the overall performance would be improved by

removing a certain bottleneck) one could decide to always work on the worst bottleneck (i.e., the one with the highest performance improvement estimation) in step 3 of the algorithm. Such a strategy can then be regarded as a variation of the steepest ascent strategy, mentioned above. In order to avoid achieving low local optima, some random structure modifications can be made form time to time, e.g. according to principles of the simulated annealing method.
Second, the structure has to be controlled or *monitored* somehow in order to detect suboptimalities or failures (which are massive suboptimalities). As stated in section 3.1, a distinction between global, local and individual scaling dimensions has been made. This inspires to build not just one adaptation mechanism, but several: one for global dimensions, some for local dimensions and one for the individual dimensions of each agent. Clearly, the latter control structure has to be integrated into the agent architecture; it remains to be discussed how to built up the former mechanisms.
One possible approach is to control optimality conditions for global and local dimensions by some agent. Control agents can simply be agents inside the structure, ADS agents (described in the next section) or some additional agents whose sole functionality is to monitor. These control agents monitor optimality conditions either by applying them in a feedback loop, by controlling them in a demon-like fashion, or by exception handling.

4 Practical Aspects of Scalability

The previous section has given an overview over various approaches how to scale a multi-agent application. These approaches have to be assessed in some way. In many cases, systems are far too complex to allow the definition of a performance function in a mathematical way. So, performance has to be measured by applying the system to suitably large scale test suites. This section copes with practical aspects of scalability. First, I describe basics of a multi-agent development environment, then two applications are introduced which will work as test beds. Finally, a possible layout of a future unifying application architecture is presented.

4.1 Basics of a Multi-Agent Development Environment

It is almost impossible to make general claims on structural issues which hold for each and every multi-agent application. However, one technique has been carried through in many applications: the usage of an *Agent Directory Service* (ADS)

agent.[8]
An ADS agent is a global blackboard agent with some specialized functionality: Agents register themselves at an ADS in order to allow other agents to find them by requesting addresses. During the registration process an agent leaves its unique agent name, type, a description of special skills, resource accessabilities, etc. Intelligent ADS agents may be able to further reason on this information. So, agents looking for cooperation partners are enabled to put complex requests on other agents to the ADS.

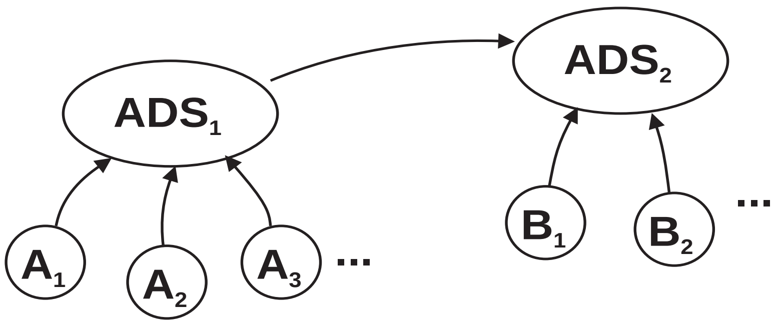

Figure 4: Agent Registration

In theory, ADS functionality may be added to any agent. In practice, however, most applications use "full time" ADS agents, i.e., agents whose only job is to provide other agents with information about possible cooperation partners. As in complex applications one ADS agent can barely administer *all* agents, several agents may perform ADS functionality: local ADS agents register themselves at more global ones. Figure 4 visualizes this approach.
The ADS functionality is a feature many multi-agent applications have in common. Other similarities cannot be stated in general, as various approaches may vary greatly.

4.2 Applications and Test Beds

In this section I will briefly present two main applications for the MAS introduced in section 2: The MARS scenario as an application for the InteRRaP system and the VIS scenario as an application for the MECCA system.

[8]Of course, the denotion such an agent may vary: For instance, in the PASHA II system [SS96] it is called *administrator*.

4.2.1 The MARS Scenario

MARS (Modeling Autonomous coopeRating Shipping companies [FMP95]) represents a collection of geographically distributed transportation companies. These companies carry out transportation tasks which are offered dynamically anytime during the run of the system. Companies have a limited set of trucks at their disposal. The goal of each company is to gain as much profit as possible, i.e., to accept as many promising offers as possible due to the restriction of capacity. Furthermore, optimal distribution of the load as well as optimal routes for each truck have to be found.
A multi-agent oriented approach to model this scenario provides not only plan generation, but also monitoring of plan execution: if a step of the plan fails or if some new task is incorporated during the execution phase, a decentralized replanning procedure is started on-line to adjust current plans to the new situation. Each company and each truck is realized by some agent that makes local decisions:

- An agent representing a company has to trade for offers (i.e., "buy" and "sell" them to other companies) and to allocate orders to the trucks of the company. This has to be done on-line (i.e., while the trucks are performing some job) and interactively (i.e., trucks evaluate their current situation and trade for delivery).

- An agent representing a truck has to generate a path from the origin of the freight to its destination. While performing the transportation it has to reconsider its plan dynamically in case traffic conditions change or another task has been assigned to the truck.

Figure 5 visualizes the organizational structure of the agents. Two types of agent communication and cooperation can be distinguished: *vertical* and *horizontal communication/cooperation*:
Interaction of agents within one shipping company is called *vertical cooperation.* This interaction is totally cooperative (as the prime goal of all these agents is to increase the profit of their company): A given truck agent may accept deals, even if this leads to some time delay for the delivery of its current freight. The job allocation procedure is realized by a modification of the well-known *Contract Net Procedure* [Smi80]: the company agent offers the task to all truck agents; the truck agents compute a utility value for accepting the job which is used as an offer to the company agent. This agent assigns the task to the truck whose corresponding agent has proposed the best offer.
A company agent's cooperation with other company agents is regarded as *horizontal cooperation.* Interaction between companies is *not* completely cooperative as their prior goals might conflict. As experiments in [FMP95] have shown, such

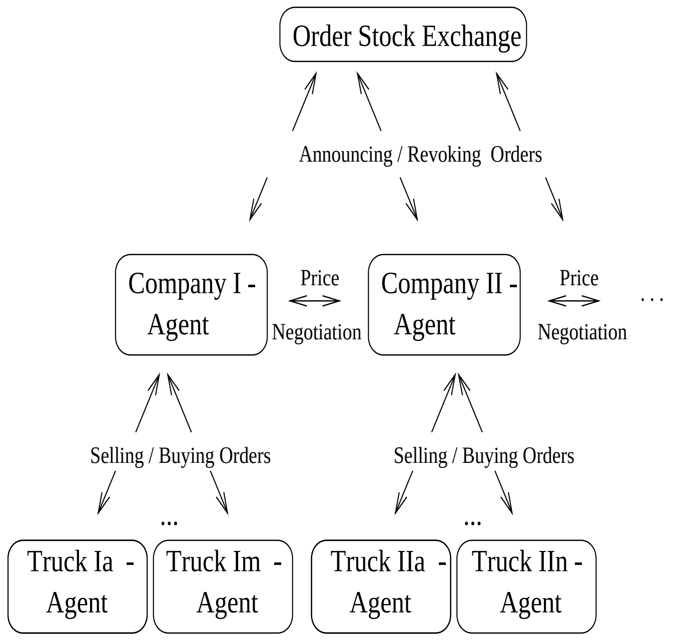

Figure 5: Organization of Agents in MARS

a cooperation can still be beneficial for all participants (a well-known law in economics). Cooperation between companies is realized by introducing a stock exchange for transportation orders. This stock exchange is organized as a blackboard where all company agents can post selling offers. If another company wants to buy the task, some bilateral negotiations are performed. The companies' decision making procedures are also based on information they obtain from their truck agents (i.e., how well that job might fit in their trucks' schedules). It has been shown that the overall performance of the system increases up to 30% if joint ventures between different companies are allowed.
One of the main applications for the MECCA system will be the VIS scenario, whose conceptualization will be presented next.

4.2.2 The VIS Scenario

VIS (Verkehrsinformationssystem; Traffic Information System)[9] is being developed for personal and individual support of travelers. Such an assistance includes route planning (for multiple means of transportation), accommodation and dining booking, reservation of tickets for cultural events, etc. During the actual trip the system assists a user not only for car navigation, it also accompanies him or

[9]Additional information can be found in [Fun96].

her while using ideally any transportation system.
Similar to the MARS scenario, a multi-agent approach has been chosen to model the scenario: During the execution of a plan, some of the circumstances may change any time which may lead to the process of dynamic re-planning. In VIS, each user is assisted by some *Personal Travel Manager* (PTM) agent. In addition, also each service source is modeled as an agent:

- A PTM has to perform the tasks mentioned above: planning of travel routes, information retrieval, booking, etc.
- An agent representing some kind of information source or service (e.g. some sort of travel service; TS) must provide this information or service to requesting agents.

The agents are distributed over many sites of some network (such as the Internet). So, communication between agents has to be performed over this net.
In order to guarantee flexibility and relevance to the current situation, the personal planning agent should always be available to the user. Furthermore, the guidance system should be as powerful as possible. Therefore, three types of user agents will be introduced: A notebook-like tool, called *mini-PTM* or *Personal Intelligent Communicator* (PIC), a system integrated in the user's car dashboard, both designed for mobile on-line usage, and a piece of software which is run on the user's PC for off-line planning. The system running on a PC is intended to have all neccessary capabilities for performing the tasks required to a PTM. The capacities of PICs and dashboard tools, on the other hand, are much too small to perform complex planning and negotiation processes. Therefore, these tools communicate over the network with some associated *broker agents* who will perform the task on behalf and report the result to the PTM. Figure 6 displays this approach graphically.
Once the PTM (or its representative broker agent) has received an order from the user it starts a planning procedure. During this procedure it has to retrieve information (e.g. from a travel service agent) and bring in offers from service provider agents. Furthermore, it has to negotiate with service providers in order to determine the "cheapest" offer.
The PTM does not only create a travel plan for its user, it also monitors the execution of the plan. The key idea here is that the PTM has always access to relevant information sources (such as traffic conditions) in order to receive the latest information: Whenever some of the assumed circumstances change, the PTM starts a re-planning process and advises the user to change his plan in case a better one has been found.

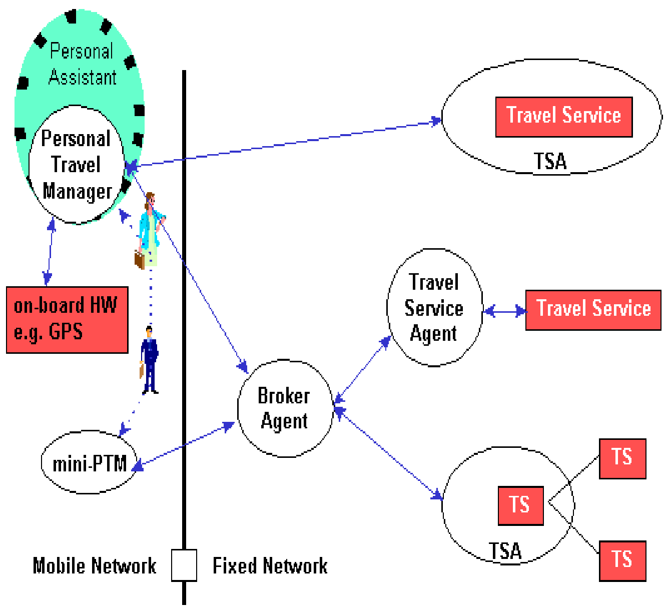

Figure 6: Organization of Agents in VIS

4.2.3 A Unifying Layout for both Applications in the Large

In the previous subsections I have presented two applications without taking their size into account: In the MARS scenario, a company agent may possibly communicate with any other company agent; in the VIS scenario communication between PTM agents and broker agents and service agents is not specified yet. However, it is obvious that in both cases some sophisticated mechanisms are needed in order to allow efficient communication and cooperation *in a very large scale scenario.*
Although these two applications differ in many perspectives, they share a common background: *geographically distributed entities are represented by agents; some of these entities are traffic participants that need traveling assistance.* Therefore, designing one common virtual scenario layout (into which an actual layout can be generated) appears to be reasonable. In this section, I shall outline a virtual application architecture which units both scenarios.

The main observation is that agents in both scenarios can be grouped together according to two criteria: Their *geographical position* and *the topic of their task.* Among these criteria virtual structures (distributed over some network, such as the Internet) can be set up. An agent can possibly be allocated into both structures (as it typically has some kind of position and has to perform some kind of

task) .
Using a structure which is sorted only according to one criterion, has severe drawbacks: agents cannot be characterized exactly. A topic-oriented structure may distinguish agent tasks at arbitrary fine granularity; agents located all over the domain can possibly be allocated into the same substructure (e.g. an ADS administering taxi agents might have to control registrations of taxi agents from all over the world.) On the other hand, a geographically oriented substructure would have to administer agents of any type that are currently located at that place. (e.g., an ADS representing the town Munich would have to administrate all agents representing entities in town.)
Thus, the question arises how to model an efficient interaction between these two structures. One idea is to introduce a two dimensional lattice, a so-called *bilattice* (presented in [Mes93]). Using a bilattice leads to some sort of *matrix organizational form*, presented in section 5.1. However, I have not pursued this idea very far yet; further investigations on how to incorporate this idea have to be made. Another approach is to build two (more or less) independent structures; each of which distinguishes at the top-level over one criterion and at the lower levels over the other criterion. This approach follows the idea of a *divisional organization*, also described in section 5.1. In the following I describe these two structures; I begin by sketching the structure with geographic location as top-level criterion.

Geographically Oriented Structure In this structure each town of the domain is represented by some ADS. Figure 7 illustrates the sample structure for the realization of the town Munich. All agents representing entities currently staying in town are registered at the *Munich-ADS* (or one of its subordinated ADS agents resp., which are sorted in a topic-oriented fashion). The town-ADS itself has a unique process name which can always be retrieved as described below. Hence, an even more global structure administering all town-ADS agents is not required. An agent registered in another town can use this unique town-ADS name in order to contact this ADS and get passed to the requested communication partner.

Topic-oriented Structure This structure is characterized in figure 8. A universal *topic-oriented ADS* is placed on top of that hierarchy. This ADS also has a unique address which has to be known by all agents. Similar to WWW search engines, it is designed to guide a topic-oriented search for communication partners by forwarding communication requests to ADS agents that administer addresses still domain-wide, but sorted by topics, such as travel, accommodation, etc. Nation-wide operating enterprises may be interested to be registered not only at their home town-ADS, but also at a national topic-oriented ADS. Re-

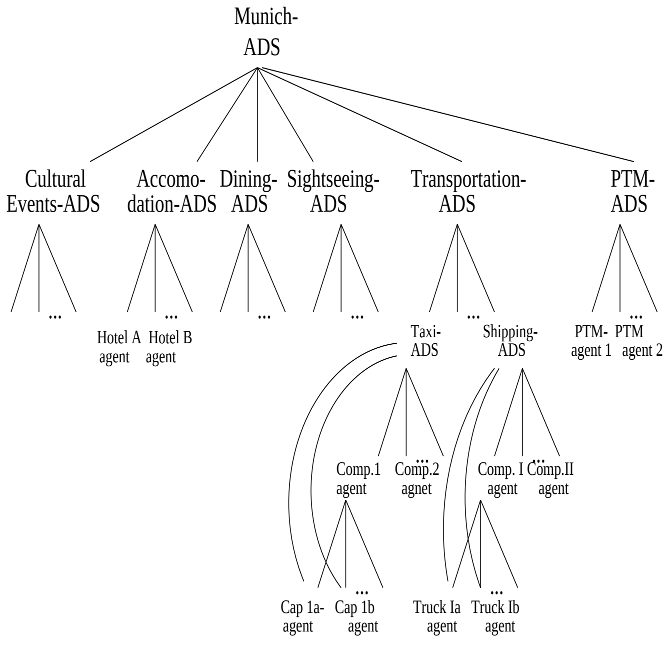

Figure 7: Layout of a Geographically Oriented Structure

gional enterprises may only be interested in being registered in the regional ADS of their topic (which may be cheaper).
Additionally, the universal topic-oriented ADS can provide links to search engine and universal broker agents. If some agent wants to retrieve the net address of some town-ADS or any other agent, it is guided to the *search agent*. This search agent retrieves the physical net address of an agent, given its virtual generic address (e.g. *ads.munich.bavaria.germany*). Another agent subordinated to the universal ADS is a universal broker agent with the functionality described in section 4.2.2. Note that ADS agents too can occur in both structures. For instance, the *Munich Accommodation-ADS* is registered at the *Munich-ADS* (as shown in figure 7) and at the *Bavaria Accommodation-ADS* (as shown in figure 8).
The structure described above is only the virtual one: It is clear that having only one top-level agent for a town or for the whole topic-oriented structure would lead to an incredible bottleneck. Hence, an actual layout must model such single agents somehow by whole agent collections in order to guarantee the required performance and robustness.
In order to illustrate this concept I will present two examples, one modeling a typical MARS application and one describing a typical VIS application.

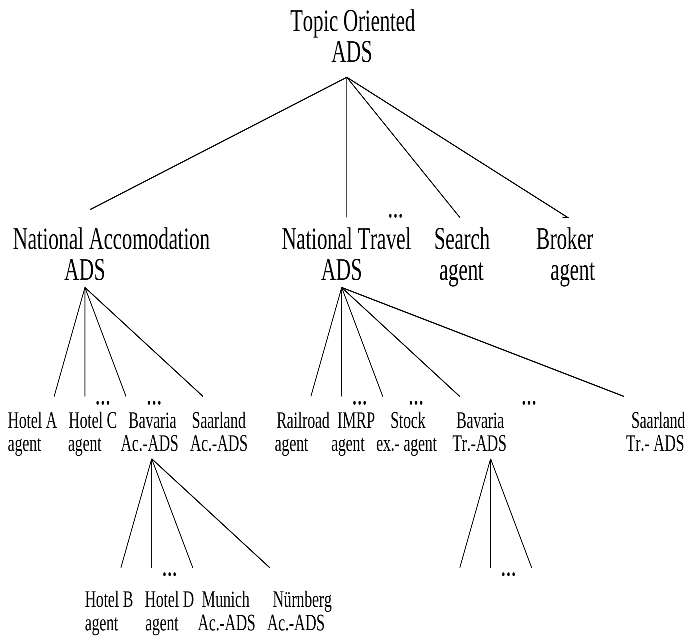

Figure 8: Layout of a Topic-Oriented Structure

Example 1: A MARS Application Suppose shipping company *II* has an order it cannot carry out and therefore wants to sell it. Using the virtual scenario layout presented above, it first contacts the nation wide operating stock exchange agent. In case the company agent doesn't know the stock exchange address it can either request it from the search agent or it can be guided to the National Travel ADS who can also provide the address of the stock exchange agent since it is registered there.
Having contacted the stock exchange agent, the company can post its offer and its physical address. Other shipping company agents who have contacted the exchange agent in the same way can then communicate with company *II* directly over the net. In case one of them accepts to take over the task it can communicate locally with its truck agents over their dashboard tools.

Example 2: A VIS Application A traveling salesman has to stay in a town, say Munich, for a couple of days: First he has to decide how to reach that town: His PTM agent contacts the nation wide *Inter-modal Routing Planner* (IMRP) agent (either directly, if the PTM agent knows already the address or via the universal topic-oriented ADS and national travel ADS). The IMRP agent returns a route so that the PTM can book traffic means, for instance, by contacting the railroad agent. Furthermore, suppose, the user wants to stay in a hotel of a certain nation wide operating hotel chain A. By requesting information from the nation wide accommodation ADS or the Munich accommodation ADS the PTM agent receives the address of the corresponding hotel agent. Thus, room booking can then be performed.

These two examples demonstrate the principle by which agents are intended to cooperate. It has to be mentioned explicitly that often *several* information retrieval processes may be successful to find some cooperation partner as, depending on the position of the requesting agent, the optimal partner retrieval strategy may vary.[10] Of course, in such a case, the requesting agent has first to reason which strategy to apply. Finally, it has to be made clear that this structure is only a first draft version. Much more examination has to be pursued.

5 Related Fields of Research

In this section I shall briefly outline how research pursued in other fields may be incorporated into this work. In particular, I hope to find inspirations to derive heuristics how to achieve and maintain high system performance.
So far, I have not done an extensive literature search and hence, the following sections are intended to give a first impression of possibilities how to incorporate results from other disciplines and to present a collection of interesting references which so far I have only partly exploited.

5.1 Business Administration: Organization Theory

The examination of this field is motivated by the idea that organizational forms developed in business administration may be used to model suitable substructures in a complex agent society. Wöhe defines in [Wöh81] a collection of possible organizational forms:

Single-Line System In a classical straight-line organization (as shown in figure 9) every unit is only subordinated to *one* higher unit, as it is typical for example in military or in plan-based economies. Thus, tasks can be distributed vertically over the hierarchy.
The advantages of such a structure are clear order of authority, transparency of the line of command and clear delineation of powers. Disadvantages lie in the length of that line of command, its inflexibility and in the danger of communication overload of intermediate and higher units.

[10]This is similar to the fact that WWW search engines may return the same WWW address on different search patterns.

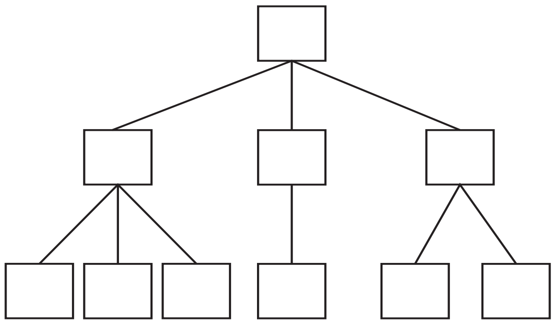

Figure 9: Single-Line System

Multi-Line System In this functional organization (displayed in figure 10) a unit is subordinated to several higher units.

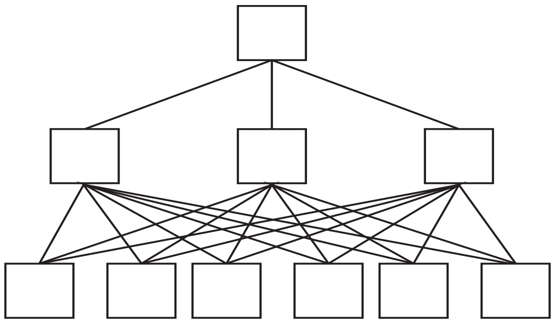

Figure 10: Multi-Line System

The pros of this approach lie in a fast and short information flow between the units and the possiblity of specialization since this structure can be used to subordinate units to *several specialists*, each of whom, however, is only entitled to give directives restricted to his sphere of authority. The cons lie mainly in the unclarity of lines of authority and responsibility separation.

Staff-Line System This system is a modification of the single-line system: decision making units are augmented with a panel of experts who have consulting functionality but no decision making competence. Figure 11 models this system.

On the one hand, competence areas and the order of authority are clearly defined, on the other hand, problems may occur since the panel has strong influence on decisions it doesn't have to be responsible of.

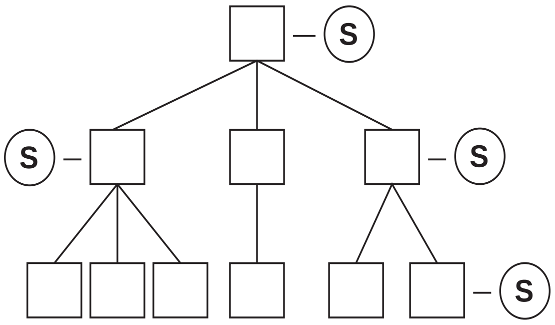

Figure 11: Staff-Line System

Divisional Organization In this type of organization a company is organized according to two classification criteria: business lines and subdivisions. Two single-line systems are combined: On the highest level the company is split according the business line criterion, on the lower levels according to the subdivision criteria. Figure 12 displays this approach.

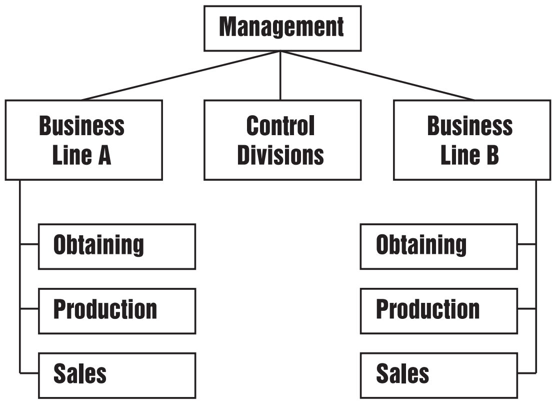

Figure 12: Divisional Organization

Technically speaking, this model is a single-line system. Hence, it inherits all properties of such a system. In practice, however, for each product almost a whole independent enterprise has to be built up. This leads to decentralization, accompanied by the effect that subdivisions have to be constructed multiple times: e.g. in figure 12 a sales department has to be built up twice. On the other hand, a company is represented in a clear fashion as no interweavings occur. This model is mainly realized in very large enterprises which can plan over a great product variety.

Matrix Organization This organizational form is also designed for integrating two organization criteria, business lines and subdivisions. In contrast to the model above, this organizational form is a version of a multi-line system: every unit is subordinated into two hierarchies, one for each criterion. Figure 13 shows the correlations. As an instance of a multi-line system, it inherits all its proper-

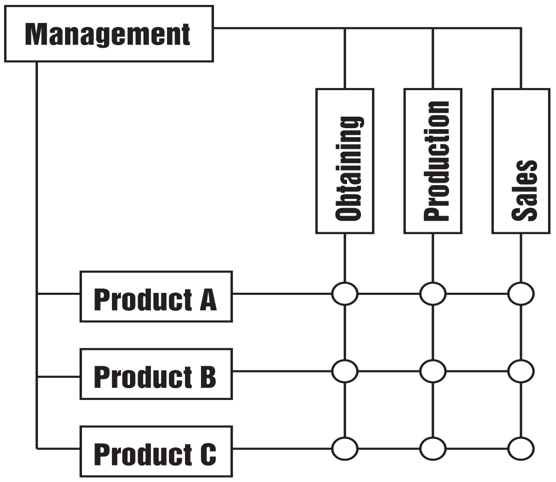

Figure 13: Matrix Organization

ties. In contrast to a divisional organization, no subdivision has to be built up twice which reduces structural overhead but may lead to competence confusion. After being en vogue in the 1980's as a favorite organizational form for very large enterprises, currently it is rejected since a clear order of authority was not visible to employees.

These organizational forms provide a collection of possible structure elements an enterprise consultant has at his disposal in order to form a company. As the pros and cons of each structure can only be characterized rather vaguely and generally, research is currently pursued in order to give some formal more rules how to model an enterprise. A. Picot describes in [Pic93] four additional organizational forms and characterizes their usage according to two properties of the task to be performed: Specificy and changeability of the task. E. Frese examines in [Fre93] variations of single-line structures and multi-line structures under the aspect how to realize several dimensions in these structures.
Other organization theorists (e.g. Kieser in [Kie94]) put much emphasis on examining informal quantities, such as the chemistry between employees, informal communication channels, etc. Traditionally, these aspects have been ignored; they were even considered as distracting. Kieser on the other hand, tries to ex-

ploit such aspects by incorporating them into his structure model. Similarly, Sydow [Syd93] proposes some promising methods how certain properties of employees can be used for the creation of a more optimal organization structure. It has to be examined how far these results can be used for modeling artificial agent societies where such aspects are of little concern at first glance.

5.2 Computer Science: Distributed Systems

The field of computer science, particular research on distributed systems, offers a wide range of perspectives for points of contact to my work. I mention only briefly some interesting aspects and describe how they can be incorporated.
A. Tanenbaum [Tan95] and F. Mattern [Mat89] describe principles of distributed operating systems. Here basics on network communication may be adapted for agent communication and migration. Observing the world's largest computer net, the Internet (a detailed description can be found in [LR93]), may lead to deeper understanding how very large scale distributed applications have to be designed. Very large scale distributed databases ([KS91] gives a good survey) are also of concern for this dissertation as synchronized communication models including locking techniques designed for systems with a very large number of net participants can be used in large multi-agent applications as well.
As mentioned in the introduction, distributed programming languages, such as Oz, provide techniques to achieve network transparency, robustness and resource adaptiveness. Of course, these techniques need not to be adapted in case a MAS is based on such a language. However, it is very unlikely that in heterogeneous multi-agent environments only one underlying programming language will be used. So, adapting the mentioned techniques in a general agent model may be inevitable for very large scale applications.

5.3 Biology: Sociology of Insect Societies

Insect life has always been fascinating to humans as insect behavior and intelligence is structured completely differently from those of mammals: Insects seem to act much more *socially*: The prosperity of an individual appears to carry much less weight than the prosperity of the society. A single insect has a neglectable intelligence; however, the whole colony shows remarkable traits of intelligence. Understanding its cause has been and still is a major goal for scientists working in this field.
Insect species can be classified by the degree of their social behavior: Wilson distinguishes in [Wil71] between six different types of social behavior, ranging from *solitary* life where adults do not care about their breed, do not live in a

common hive and do not share labor, up to *eusocial* behavior, where all three traits can be found.
Of course, insects of the latter type are of the most interest for this work. The insect probably most investigated is the *Apis mellifera*, the honeybee. (a survey can be found in [See85].) Typically, a honey bee colony consists of one queen, several thousand workers (between 5000 and sometimes more than 10 000) and drones whose population varies over the seasons, in summertime roughly 1500)
C. Starr explains bee behavior in [Sta79] as a result of their genetic relationship: individuals prefer cooperating with close relatives rather than cooperating with other colony members. He is able to explain several interesting traits of group behavior with his theory. An adaption of this idea has to be considered.
In the past some insect researchers (e.g., Wheeler [Whe11] have created the notion of a *super organism*: A colony is compared to a metazoan body. Analogies which have caused this analogy are replication, import and export of materials, control of the inner environment, such as temperature, and finally, response and orientation to the external environment. However, the super organism concept failed, mainly for the reason that in a metazoan body all cells are genetically identical which is not true for members of an insect colony.
Furthermore, it has been found that colony members have individual goals which partly contradict those of other members. Neither pursuing only one of these goals, nor adding them together would lead to a strategy allowing the colony to survive; the successful survival strategy is a result of a very complex communication and action behavior: Communication between queen and other colony members is realized via *pheromones*, chemicals, secreted by the queen and spread by workers as liquids or gas. Using pheromones, the queen influences the behavior of workers who might have goals inhibiting the queen's goals. This method, which can be represented by a feedback loop system, allows the colony as a whole to react on changes of the internal and external environment, such as death of the queen, critical size of the colony, etc. Figure 14 gives an overview; a detailed description can be found in [Fre87].
Such a feedback loop system has been proven very robust for achieving the colony's prime goal, to survive. This approach, where centralized control meets decentral decision finding in a fashion that turns out to be very satisfying for all colony members, has clearly inspired the introduction of the feedback loop algorithm proposed in section 3.5. Further investigations may lead to more insights in order to further improve this method.

5.4 Psychology: Group Behavior

Group theorists have been studying behavior of human groups that have to solve a given task. In particular, scientists are interested in determining for a certain

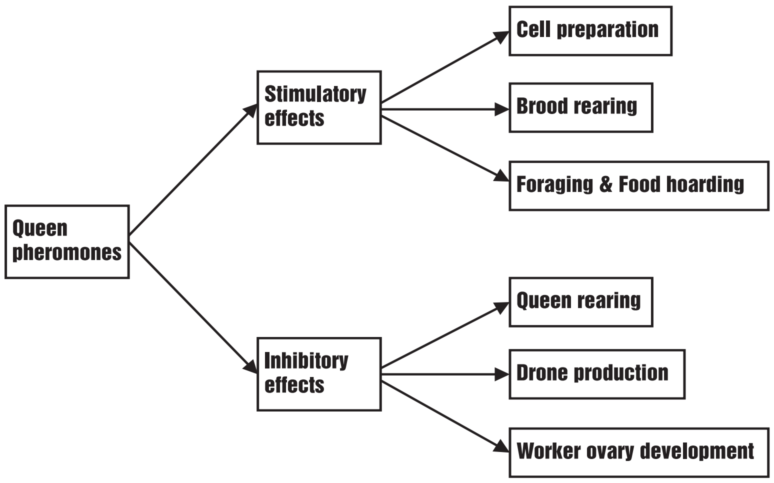

Figure 14: Effects of pheromone transmission

task the size and degree of diversity by which a group performs best. In my work I will examine if these results can be used for modeling efficiently cooperating artificial agent groups.

In [Mor96], R. Moreland summarizes research results where empirical investigations have been made by observing social interaction, by asking people and by artificially creating groups of different sizes and measuring their performance: Experiments have shown that the optimal size of a cooperating group varies from five up to a dozen members (see [Nas88], [Sch89] or [CML93] for details), depending on the task and on properties of the test group. These results are rather vague and they show that there is no simple way to determine the perfect size of a human group. Hence, a better approach might be to study some of the correlates of group size:

Moreland compares in [Mor96] characteristics of larger groups to those of smaller ones: Generally speaking, larger groups enjoy several advantages: They have access to more resources, including time, money, and expertise, they tend to be more diverse. On the other hand, larger groups suffer several disadvantages: they often experience coordination difficulties which may decrease problem solving performance. Furthermore, there is more conflict among members of larger groups. In general, they are less willing to cooperate.

A great diversity of a group can improve its problem solving performance as specialists might be available to perform a certain task. The main risk of a great diversity is that it can produce conflicts among members.

Moreland concludes that all these factors make it difficult to specify the optimal size or degree of diversity of a human group for pursuing a certain task. He

suggests rather than worrying about the best group size and degree of diversity, it might be wiser to maximize the advantages and minimize the disadvantages of a group, whatever size that group has reached. This is not very promising for adapting general rules to the world of artificial agents. Still, I plan to further study this field in order to find heuristics about good work size and degree of diversity of groups of artificial agents.
In [TJ77], B. Tuckman and M. Jenson examine the formation of groups. They regard group formation as a continuous process moving through a five stage life cycle: In the *forming stage*, members seek to orient themselves to the group. In the *storming stage*, members try to alter the group to satisfy their personal needs. In the *norming stage*, members endeavor to resolve the disagreements and tensions which threaten group success. In the *performing stage*, members attempt to maximize group performance and productivity. Finally, in the *adjourning stage*, members disengage from the group emotionally and behaviorally which leads to the end of the group life cycle. It will be an interesting task to examine if such a life cycle can be established for groups of artificial agents as well.

6 Conclusion and Time Schedule

Overall goal of the dissertation is to examine scalability of multi-agent systems and to derive a method which allows systems to self-configurate to any application scale and nature. This goal is motivated by the necessity to guarantee high performance of multi-agent systems.

6.1 Summary

In detail, I plan to achieve the following goals:

- Examination of scalability possibilities on three levels: the agent society level, the communication/cooperation level and the agent architecture level,
- Investigation of how to express these possibilities as search space dimensions for an performance optimization method,
- Development of an appropriate performance optimization method,
- Examination of related fields to find stimuli for the work on the above issues,
- Development of a large scale test suite,
- and evaluation of the obtained results.

These points are addressed throughout the proposal as follows: Two concrete multi-agent systems (presented in section 2) serve as bases for this dissertation. They can be used for gaining practical experience with quantities relevant for scaling. Scalability possibilities will be investigated in section 3. A method to derive self-configurating systems is also proposed in this section. In section 4 two very large scale applications (intended to serve as test suites) are presented. Results of research performed in related fields, (points of contacts were outlined in section 5) can be used to define possible structure elements (see organization theory), to find heuristics on an optimal structure (see psychology), to receive stimuli for defining the described feedback loop (see biology), etc.

6.2 Time Schedule

The first time phase of my work will be used for modeling the logical application layout in order to derive a test bed for applying and evaluating various scaling approaches. In addition, it shall be used for the continuing collecting of additional information on various related topics.
During a second phase emphasis will be put on comprehending the correlation of scaling quantities and the derivation of general rules or patterns of behavior. Investigating scalability on the levels of society structure and agent communication/cooperation might partially be realized in cooperation with students by assigning Master's Theses (Diplomarbeiten) on clearly defined subranges of these fields. Examining scalability on the agent architecture level can be done together with co-workers in CoMMA-MAPS and AiV who are focusing on topics such as efficient planning, knowledge representation and learning. At the end of the second phase the development of a self-adapting system and an evaluation of the method will be on focus.
In the third phase the results of the previous phases shall be collected and written down in the thesis. Furthermore, these results shall be integrated in releases of both, VIS and MARS applications.

The whole dissertation shall be completed in three years. Finishing the first phase is scheduled after nine months. The time consuming second phase is intended to be completed after additional 18 months which leaves the last nine months for the final third phase.

Acknowledgments

I would like to thank Prof. Siekmann for providing this opportunity and environment to me. The original motivation for this work came from a suggestion by Dr. Kleinschmidt. I am thankful to all members of the CoMMA-MAPS and AiV groups, especially Dr. Fischer and Dr. Steiner for their helpful ideas, and to Jay C. League and Thomas W. Bullock.

References

[AMO93] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows.* Prentice-Hall, 1993.

[Aus62] J. L. Austin. *How to do things with words.* Oxford University Press, 1962.

[Bat94] J. Bates. The role of emotions in believable agents. *Communications of the ACM*, 37(7):122–125, 1994.

[BB89] G. Bamberg and F. Baur. *Statistik.* Oldenbourg, München Wien, 6th edition, 1989.

[BOH$^+$94] Stephan Busemann, Stephan Oepen, Elizabeth Hinkelman, Günter Neumann, and Hans Uszkoreit. COSMA–multi-participant NL interaction for appointment scheduling. Research Report RR-94-34, German Research Center for Artificial Intelligence, Saarbrücken, October 1994.

[Bra87] M. E. Bratman. *Intention, Plans, and Practical Reason.* Harvard University Press, Cambridge, Mass., 1987.

[CL90] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. In *Artificial Intelligence*, volume 42, pages 213–261. 1990.

[CML93] M. Cini, R. Moreland, and J. Levine. Group staffing levels and responses to prospective and new group members. *Journal of Personality and Social Psychology*, 65:723–734, 1993.

[FMP95] K. Fischer, J. P. Müller, and M. Pischel. A model for cooperative transportation scheduling. In *Proceedings of the 1st International Conference on Multiagent Systems (ICMAS'95)*, pages 109–116, San Francisco, June 1995.

[FPB75] Jr. Frederick P. Brooks. *The Mythical Man-Month, Essays on Software Engineering.* Addison Wesley, 1975.

[Fre87] J. Free. *Pheromones of Social Bees.* Chapman and Hall, London, 1987.

[Fre93] E. Frese. Organisationsstrukturen, mehrdimensionale. In E. Frese, editor, *Handwörterbuch der Organisation*, pages 1670–1688. 3rd edition, 1993.

[Fun96] P. Funk. Entwurf eines Reiseszenarios. Internal Paper, December 1996.

[Hol75] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbour, MI, 1975.

[IEE81] IEEE. *Transactions on Systems, Man and Cybernetics*, volume 11, 1981.

[KGV83] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimalization by simulated annealing. *Science*, 220:671, 1983.

[Kie94] A. Kieser. Fremdorganisation, Selbstorganisation und evultionäres Management. *zfbf*, 46(3):199–227, 1994.

[KS91] H. Korth and A. Silberschatz. *Database Concepts.* McGraw Hill, 2nd edition, 1991.

[LR93] D. Lynch and M. Rose. *Internet System Handbook.* Addison-Wesley, 1993.

[Lux95] A. Lux. *Kooperative Mensch-Maschine Arbeit: Ein Modellierungsansatz und dessen Umsetzung im Rahmen des Systems MEKKA.* PhD thesis, Universität des Saarlandes, Saarbrücken, 1995.

[Mat89] F. Mattern. Verteilte Basisalgorithmen. In *Informatik-Fachberichte*, volume 228, pages H134 – H135. Springer-Verlag, 1989.

[Mes93] B. Messing. Integrating knowledge bases: Towards symbolic representation and confict managing. 1993.

[Mor96] R. Moreland. Creating the ideal group: Composition effects at work. In E. Witte and J. Davis, editors, *Understanding Groupd Behavior*, volume 2. Lawrence Erlbaum Associates, Publishers, Mahway, NJ, 1996.

[Mül96a] J. P. Müller. *An Architecture for Dynamically Interacting Agents.* PhD thesis, Universität des Saarlandes, Saarbrücken, 1996.

[Mül96b] J. P. Müller. A markovian model for interaction among behavior-based agents. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents — Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, volume 1037 of *Lecture Notes in Artificial Intelligence*, pages 376–391. Springer-Verlag, 1996.

[Nas88] D. Nasser. How to run a focus group. *Public Relations Journal*, 44:33–34, 1988.

[Pic93] A. Picot. Organisationsstruktur in der Wirtschaft und ihre Anforderungen an die Informations- und Kommunikationstechnik. In *Handbuch des Informationsmanagements, Aufgaben - Konzepte - Praxislösungen* , pages 49–68. A.W. Scheer, 1993.

[Pid92] M. Pidd. *Computer Simulation in Management Science.* Wiley, 3rd edition, 1992.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[SB95] Werner Stephan and Susanne Biundo. Deduction-Based Refinement Planning. Research Report RR-95-13, German Research Center for Artificial Intelligence, Saarbrücken, 1995.

[SBKL93] D. Steiner, A. Burt, M. Kolb, and C. Lerin. The Conceptual Framework of MAI^{2}L. In *Proceedings of the 5th European Workshop on Modelling Autonomous Agents and Multi Agent Worlds (MAAMAW'93)*, Neuchatel, Switzerland, August 1993.

[Sch89] A. Scharf. How to change seven rowdy people. *Industrial Management*, 31:20–22, 1989.

[Sch95] A. W. Scheer. *Wirtschaftsinformatik, Referenzmodelle für industrielle Geschftsprozesse, Studienausgabe.* Springer Verlag, Berlin, 6th edition, 1995.

[See85] T. Seeley. *Honeybee Ecology - A Study on Adaptation in Social Life.* Princeton University Press, Princeton, NJ, 1985.

[Sho91] Y. Shoham. Agent-oriented programming. In *Proc. of the 11th International Workshop on DAI*, pages 345–353, 1991.

[Sie95] Siemens AG, Zentralbereich Forschung und Entwicklung SN5 Mensch-Maschine-Kooperation. *Personal Trip Assistant: Dokumentation zur ZFE-Pressekonferenz Verkehr*, 1995.

[Smi80] R. G. Smith. The contract net protocol: High level communications and control in a distributed problem solver. *IEEE Transactions on Computers*, 29:1104–1113, 1980.

[SS96] S. Schmeier and A. Schupeta. PASHA II- Personal assistant for scheduling appointments. In *Proceedings of the first international conference of Practical Applications of Intelligent Agents and Multi-Agents: PAAM'96*, London, April 1996.

[SSR95] G. Smolka, C. Schulte, and P. Van Roy. PERDIO: Persistent and Distributed Programming in Oz. Technical report, German Research Center for Artificial Intelligence, Saarbrücken, March 1995.

[Sta79] C. Starr. Origin and evolution of insect sociality: A review of modern theory. In H. Herrmann, editor, *Social Insects*, volume 1, chapter 2, pages 35–80. Academic Press, 1979.

[Ste92] D. Steiner. MEKKA: Eine Entwicklungsumgebung zur Konstruktion kooperativer Anwendungen. In J. Müller and D. Steiner, editors, *Kooperierende Agenten*, number D-92-24, pages 17–21. Saarbrücken, 1992.

[Syd93] J. Sydow. *Strategische Netzwerke.* Gabler, Wiesbaden, 1993.

[Tan95] A. Tanenbaum. *Distributed Operating Systems.* Prentice Hall, 1995.

[TJ77] B. Tuckman and M. Jenson. Stages of small-group development revisited. *Group and Organization Studies*, 2:419–427, 1977.

[Whe11] W. Wheeler. The ant-colony as an organism. *Morphology.*, pages 307–325, 1911.

[Wil71] E. Wilson. *The Insect Societies.* The Belknap Press of Harvard University Press, Cambridge, MA, 1971.

[WJ95] M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[Wöh81] G. Wöhe. *Einführung in die Allgemeine Betriebswirtschaftslehre.* Verlag Wahlen, 14th edition, 1981.

[Zou76] G. Zoutendijk. *Mathematical Programming Methods.* North-Holland, Amsterdam, 1976.

TM-97-02
Technical Mem

Scalability of Multi-Agent Systems

Christian Gerber