Business Organizational Forms in Self-organizing Multiagent Systems

Diplomarbeit

von
Tore Knabe

nach einem Thema von Prof. Dr. Jörg H. Siekmann
im Fachbereich 6.2, Informatik, der Universität des Saarlandes

October 9, 2002

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich diese Arbeit selbständig verfaßt, nur die im Literaturverzeichnis zitierten Quellen benutzt und sie noch keinem anderen Prüfungsamt vorgelegt habe.

Saarbrücken, im Juli 2002 Tore Knabe

Acknowledgments

I would like to thank Prof. Siekmann for the opportunity to write this thesis at his chair. I am also grateful to Michael Schillo, whose support, supervision, and encouragement made this work possible. For important input on the computational side I thank Sven Jacobi and Rainer Siedle, and for the sociological side I thank Bettina Fley, Michael Florian, Frank Hillebrandt, and Daniela Hinck.

Abstract

This thesis investigates whether organizational forms derived from sociological models of real-world business organizations can increase the performance of a market-based multiagent system. The market consists of customer agents who have tasks to assign and provider agents who have the resources to complete the tasks. The tasks can be complex, so that completing a task requires several provider agents to work together. The multiagent system's efficency is measured by several performance measures. We hypothesize that the efficiency can be increased if repetitive patterns in the customer tasks or their subtasks are mirrored by corresponding structures on provider side in the form of organizations. We further hypothesize that provider agents can be programmed to increase the system's efficiency on their own by self-organizing, i.e., by changing their organizational structure based on information locally available to them.

We specify a number of selected organizational forms derived from sociology and specify a mechanism for self-organization. We also develop new communication protocols that help to structure the coordination processes of task assignment and self-organization among the agents. The concepts proposed in this thesis have been implemented in a testbed that has been used for the experimental evaluation of the ideas of this thesis and is planed to serve for further research in the interdisciplinary field *socionics*.

Our evaluation showed that the concepts of organizations and self-organization can improve the efficiency of multiagent systems. We describe which organizations are suited for which circumstances, when self-organization is advisable, and give examples of possible applications of these concepts in other systems.

Contents

List of Figures

Chapter 1

Introduction

This thesis is a product of the recently increasing cooperation of the fields artificial intelligence and sociology. Although it draws on both disciplines, the emphasis is strongly on the side of artificial intelligence. Our goal is to investigate the application of sociological concepts to computational systems. The first section describes the computational systems we have in mind and the concepts derived from sociology. The second section reports the results of our experimental evaluation of applying the concepts to the system. The third section gives an overview over the structure of this thesis.

1.1 Problem Description

The present thesis deals with the application of models of business organizations derived from sociological theories to self-organizing multiagent systems. The central setting for this work is a market of two kinds of agents, customers and providers. The customer agents have tasks they need to be done by provider agents. These tasks can be complex and may go beyond the capacities of any single agent, so that they require the working together of several providers. The customers try to find providers who can complete the task to be assigned by auction mechanisms. In order to find out how efficient this market works and to measure the changes in performance that result from application of sociological concepts, we define a number of performance measures. If the system contains many agents or the tasks to be assigned require the collaboration of many providers, assigning the tasks with auction systems becomes increasingly complex and methods to improve the performance of the system as measured by these performance measures become more important. We hypothesize that the system performance increases if repetitive structures or substructures in customer demand are mirrored by organizational structures on provider side. Since customer demand can usually not be predicted by the providers, we develop a mechanisms for them to change their organizational structure and adapt it to

new order situations. We implemented the proposed concepts in a testbed and experimentally evaluated the effect of organizations and self-organization on the performance measures.

1.2 Main Results

Our experimental evaluation showed that organizations and self-organization can increase the efficiency of multiagent systems. Whether to pre-design the organizations or use self-organization depends on the specific conditions of the scenario. Our results suggest that if the order situation does not change and it is important to have a low rate of failed task assignments and few messages used for the auctions, it is advisable to either put all agents into organizations such that no single agents are left or, if single agents have to be present, make all organizations of one of the forms *group* or *corporation*. If the order situation does change, then starting all agents as single agents and allowing them to self-organize will still provide the system with the advantage of using fewer messages, without increasing the number of failed task assignments.

1.3 Structure of the Thesis

The second chapter gives an overview of the research field underlying the thesis. Since this thesis is based on the interdisciplinary field *socionics*, some of the research fields are rooted in sociology, while others are rooted in computer science. The third chapter gives a more detailed problem description, states the research goals of the thesis and offers some examples for applications of the proposed concepts. We give a formal specification for the system in the fourth chapter, and describe a testbed implementing these specifications in the fifth. The sixth chapter presents the experimental plan and the results of the experimental evaluation. Finally, the seventh chapter concludes the thesis by giving a summarizing overview and offering ideas for future work. The appendix contains the experimental configurations in tabular form.

Chapter 2

Background and Related Work

This thesis is rooted in the field *socionics* and is an attempt to contribute to the understanding of presently not thoroughly investigated issues in this new research area. Socionics is a recently created field between sociology and artificial intelligence [Malsch, 2001]. The name's similarity to bionics reflects the common idea shared by both fields: just as biological phenomena inspire technical innovations in bionics, socionics looks for concepts in sociology that can be adopted in computer systems to increase the efficiency of implementations of artificial intelligence. Studying these implementations feeds back on sociology in that simulations of societies of artificial, social entities can serve as a testbed for social theories that are difficult to evaluate in real human societies. In socionics, scientists from both fields cooperate to investigate three major issues of interest in this interdisciplinary area:

1. Modern society has a large inventory of concepts that multiagent systems could be modeled on, for example cultural values, social roles, norms and conventions, shifts and institutions, or structures of power and authority. Researchers working in the field hope that applying these concepts will give multiagent systems some of the adaptivity, robustness, scalability, and reflexivity of social systems.

2. On the other hand, sociology might profit from artificial intelligence by using multiagent systems as simulation tool for testing and elaborating its own concepts, models, and theories. Of special interest are dynamic interactions between the micro- , meso- and macro-level on which societies can be described. The micro-level studies interactions between individual actors. The macro-level is concerned with phenomena of social systems at large, like cultures, and the meso-level lies between the two; it studies structures like organizations.

3. Finally, socionics deals with the study of hybrid societies, which may contain both software programs and humans as agents and which many believe to

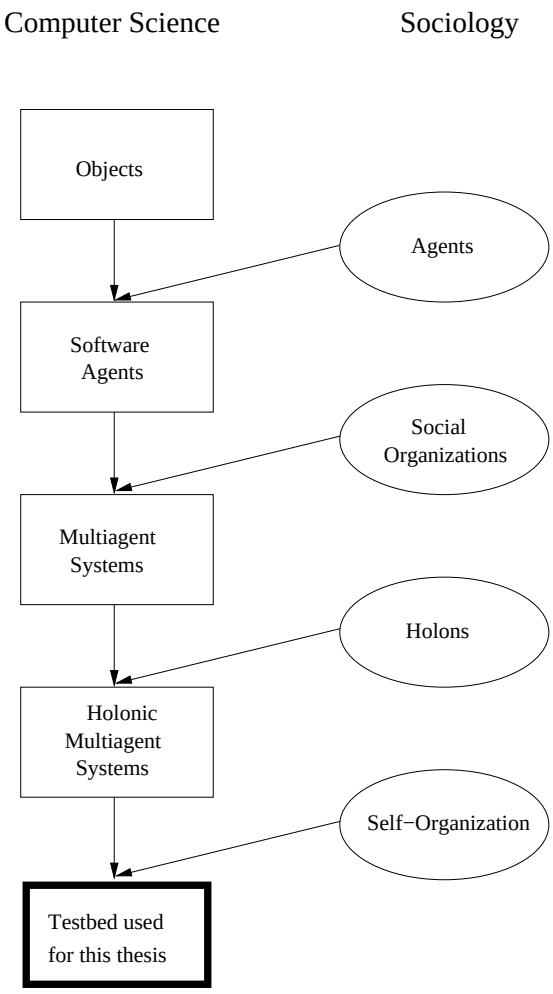

Figure 2.1: Relationships between fields in sociology and computer science.

play a larger role in the future. An important research question in hybrid societies is how these societies influence our interaction with technology and with each other.

This chapter gives an overview of the fields of computer science and sociology that are relevant for understanding the motivation behind the concepts proposed in this thesis and for developing the testbed that we used to evaluate them. Figure 2.1 shows the fields and the relationship between them. The left column consists of concepts in the field of computer science. Each concept developed from a previous one, indicated by arrows connecting upper, "earlier" concepts to lower, "later" ones. The right column consists of sociological concepts that had an influence on the development of the fields of computer science. Except for the uppermost box and the uppermost oval, each box and oval is described in its own section.

The *object-oriented programming* paradigm of computer science that emphasizes the importance of abstraction and encapsulation becomes a new paradigm under the influence of the notion of autonomous, reactive, proactive and social *agents* as observed in nature: the paradigm of *software agents*. Individual software agents vary widely in how complex their implementation is. Instead of increasing the level of sophistication of individual agents, some researchers take their inspirations from *social organizations*, and study the emergent phenomena that appear through interaction of many simple units in *multiagent systems*. Natural complex systems often appear to have a recursive structure; structural units are elements of larger structures and at the same time are made up of more fine-grained elements themselves. Applying this concept of *holons* to multiagent systems results in *holonic multiagent systems*, where groups of agents can build structures that look and behave like single agents in the system. If these agents change these structures by grouping with other agents or changing their roles and functions inside the group they form a system that is capable of *self-organization*, of adapting to environmental changes in order to increase its performance or that of individual agents. We have implemented such a system in a testbed to evaluate some models of social organizations in self-organizing holonic multiagent systems.

2.1 Social Organizations

The ability to cooperate might be one of the most important factors responsible for the current position of humans in Earth's ecology. Other species do also show forms of cooperation, as seen in insect states, swarms of fish or wolf packs. However, these forms are much more limited in range of interaction between the cooperating individuals as well as changing memberships. They are, therefore, not as big an improvement over the capabilities of an individual and less flexible to changing circumstances when compared to cooperations as seen in human societies. Human beings can choose how and whom to interact with depending on the configuration of one's own and others' goals, resources, abilities as well as the current and possible future situations. Different such configurations call for different forms of cooperation. Sociologists have studied these forms, ranging from short term interactions between strangers to big, long lasting organizations like multi-national enterprises. This section gives an overview of some of their insights that promise to be applicable to a scenario of agents in a market-based system.

The most important factors identified by sociologists that decide what form of cooperation will suit its members' interests are: goals, abilities, and performance (Figure 2.2).

Each individual's goals and abilities determine his actions, his 'local' performance. The performance of all individuals taken together in turn determines the group's performance, the 'global' performance. An individual is motivated to

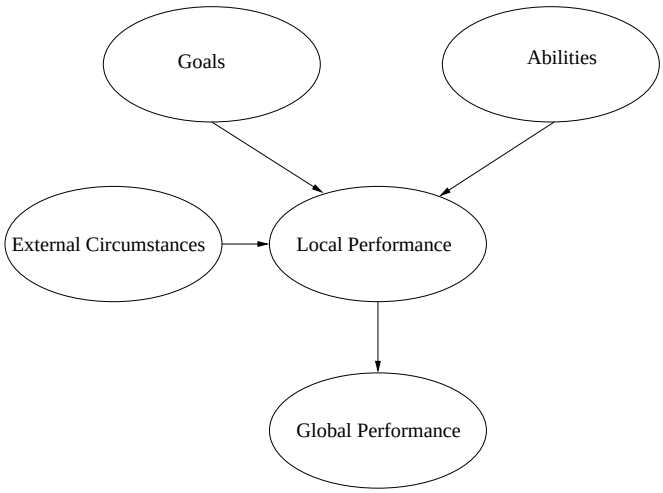

Figure 2.2: Causal relationships between the social factors influencing the form of cooperation.

participate in the group if this global performance satisfies his goals more than he could by not participating. In the words of March and Simon: "An organization will exist so long as it can offer its members inducements which exceed the contributions it asks of them." [March and Simon, 1958]

According to Mayo and Barnard, the fundamental problem of cooperation stems from the fact that individuals have only partially overlapping goals [Mayo, 1945, Barnard, 1968]. Further complications arise from an individual's true goals and abilities not being directly visible to others, and from external circumstances not always being predictable. Participating in a group can therefore pose a risk to the individuals, especially if it requires them to invest some of their resources without the guarantee of a certain pay-off.

2.1.1 Principles of Interaction

Several mechanisms have evolved to enable cooperation in spite of these uncertainties. Three major principles of these mechanisms are economic exchange, authority, and gift exchange. Williamson saw economic exchange and authority as two alternative modes of organizing economic activities, manifested in the extreme forms of markets and hierarchies [Williamson, 1975]. Gift exchange can be seen as a dimension orthogonal to the first two.

Economic Exchange

Economic exchange is the foundation of markets: an individual gives some of his resources and receives others in return. Implicit or explicit in this exchange is a contract that states what is exchanged for what. Therefore, market transactions

consist of contractual relationships.

Each party is bound only to deliver what is specified, so the contract must specify who must deliver what under every possible state of nature. A simple example is buying bread, where a set amount of money is exchanged for a set number of bread loafs. Economic exchange works well in cases like these when there is low uncertainty and the explicit or implicit contracts can be made to cover all possible contingencies concerning the exchange. Under such circumstances, auctions provide a very, if not the most, efficient means of finding suitable trading partners.

However, auctions fail when the nature of the exchange makes it difficult to objectively specify all aspects in a contract. In the bread example above, it would be difficult to describe the expected product quality in an auction's call for proposals.

Incomplete specifications leave many opportunities for economic actors to maximize their own profit at the expense of those they deal with. For example, the bread might be made from minor quality ingredients.

This uncertainty makes it risky to acquire certain goods or services on the market. Although there are reputation mechanisms that help to increase trust and reduce risk, they cannot completely eliminate it. Furthermore, there are costs bound to them; economists call them 'transaction costs': "A transaction cost is any activity which is engaged in to satisfy each party to an exchange that the value given and received is in accord with his or her expectations. Transaction costs arise principally when it is difficult to determine the value of the goods or service. Such difficulties can arise from the underlying nature of the goods or service or from a lack of trust between the parties" [Ouchi, 1980]. Jarillo sees lack of trust as the quintessential cause of transactional costs [Jarillo, 1988], similar to Williamson, who finds opportunism a central concept in the study of transaction costs [Williamson, 1975]. If transaction costs are too high, economic exchange needs to be replaced by other mechanisms, such as authority.

Authority

The mechanism of authority is found in the hierarchical structures of organizations. It is less flexible and less efficient than an optimal solution that could in principle be reached with a market mechanism if there were no transaction costs. "An organization such as a corporation exists because it can mediate economic transactions between its members at lower costs than a market mechanism can" [Ouchi, 1980]. In the absence of transaction costs, there would be no firms at all, just a market of separate economic atomistic units. "In this sense, every bureaucratic organization constitutes an example of market failure" [Ouchi, 1980].

Economic units can get together to form a hierarchy, with some units agreeing to take orders from units higher in this hierarchy. The economic exchange

Motivation	*Aim*
Altruism	making others happy
Egoism I	exchange
Egoism II	warm glow, social approval
Strategical	signaling, building trust
Fairness	norms, reducing inequity
Survival	selection

Table 2.1: Theories of gift giving [van de Ven, 2000].

for every single transaction has been replaced by employment relationships. The employment relation is an incomplete contract; by agreeing to be employed, a worker accepts the right of superiors to direct and monitor his work activities. The direction, surveillance, and evaluation made possible in a bureaucratic organization lowers the uncertainty and opportunism that lead to high transaction costs in market mechanisms, but they lower the workers' autonomy as well.

In summary, the autonomy in market systems is given up in hierarchical organizations. Many organizational forms can be placed somewhere between the extremes of market and hierarchy. The degree to which autonomy is reduced by authority depends on the nature of the tasks the organization is dealing with. Different tasks call for different forms of organizations, with different positions on the market/hierarchy dimension.

Gift Giving

The third mechanism, much less well known than economic exchange and authority, is gift giving. Economic agents might accept transactions which benefit another, without receiving an appropriate compensation or being forced to this transaction by authority. They are said to have given a "gift" to the other. Gift giving is part of current business practice; "Managers admit that informal cooperation between competitors exists for specified tasks with little profit and that subcontractors are deliberately given jobs in times of low income to let them survive economically to the next period of many customer requests, where the subcontracting will again be beneficial" [Schillo et al., 2001a]. Other examples of gift giving can be seen in the open source contribution of programmers and the scientific community. van de Ven discusses various theories that try to explain the motivation for giving [van de Ven, 2000]. Table 2.1.1 gives an overview of these theories.

In the context of this thesis, gift giving will be interpreted as a signaling device. Giving a gift, for example by accepting an economic dissimilarity in a transaction, the giver signals that he is interested in a longer relationship based on mutual trust. The underlying principle is the expectation of reciprocity. According to

Gouldner, reciprocity is one of only two social agreements that have been found to be universal among societies across time and cultures (the other is the incest taboo) [Gouldner, 1961]. Gift giving and authority might be seen as two different ways of dealing with the lack of trust inherent in a system of actors with different interests.

2.1.2 The ADICO Grammar

Modeling institutions requires identifying their basic characteristics and implementing them in an algorithmic form. The political science literature offers a formalism that can act as an interface between the real world of institutions and the world of precise computer models: the ADICO grammar.

The ADICO grammar has been proposed in 1995 by Crawford and Ostrom to facilitate the analysis of institutions and cooperations in game theory and behavior research [Crawford and Ostrom, 1995]. Crawford and Ostrom view institutions as "enduring regularities of human action in situations structured by rules, norms, and shared strategies, as well as by the physical world."

According to their view, the basic principles characterizing and distinguishing institutions can be expressed in linguistic statements, all of which can be assigned to one of the categories mentioned above: rules, norms, and shared strategies. An important assumption underlying this view is that even implicit and tacit agreements can be expressed in this form.

The ADICO grammar offers a way of decomposing such linguistic statements into five components: `ATTRIBUTES`, `DEONTIC`, `AIM`, `CONDITIONS`, and `OR ELSE`. The ADICO acronym is derived from letters of these components. They have the following meaning:

A `ATTRIBUTES` is a holder for any value of a participant-level variable that distinguishes to whom the institutional statement applies (e.g., 18 years of age, female, college-educated, 1-year experience, or a specific position, such as employee or supervisor).

D `DEONTIC` is a holder for the three modal verbs using deontic logic: *permitted* (P), *obliged* (O), and *forbidden* (F).

I `AIM` is a holder that describes particular actions or outcomes to which the deontic is assigned.

C `CONDITIONS` is a holder for those variables which define when, where, how, and to what extent an AIM is permitted, obligatory, or forbidden.

O `OR ELSE` is a holder for those variables which define the sanctions to be imposed for not following a rule.

All shared strategies can be written as

`[ATTRIBUTES]` `[AIM]` `[CONDITIONS]` (AIC);

all norms can be written as

`[ATTRIBUTES]` `[DEONTIC]` `[AIM]` `[CONDITIONS]` (ADIC),

and all rules can be written as:

`[ATTRIBUTES]` `[DEONTIC]` `[AIM]` `[CONDITIONS]` `[OR ELSE]` (ADICO).

Norms can be seen to subsume shared strategies, and rules to subsume norms.

Crawford and Ostrom offer an example for the transformation of a linguistic rule underlying the cooperative behavior of a group of people to ADICO grammar:

> *All villagers must not let their animals trample the irrigation channels, or else the villager who owns the livestock will be levied a fine.*

A	all villagers
D	F
I	irrigation channel trampled by their animals
C	at all times
O	fine

2.1.3 Selected Organizational Forms

Existing business organizations can be classified by the degree to which they use the three mechanisms above: rules, norms, and shared strategies. Five forms of organizations will be introduced with regard to their accepted norms and applied mechanisms: the cooperation, the virtual enterprise, the strategic network, the group, and the corporation.

Virtual Enterprise

The definition of virtual enterprises varies widely. The present thesis adopts the definition of Hales: "The virtual enterprise is a network of independent firms in a synergistic relationship with a central executive entity operationally linked by extensive ICT [information and communications technologies] to achieve a common goal" [Hales and Barker, 2000]. Virtual enterprises have attracted a lot of interest in recent times, the main reasons for this being globalization and single sourcing [Kemmner and Gillessen, 2000]. Globalization has exposed enterprises to an environment of much stronger competition, increasing the pressure for faster, more efficient production and better services. The term 'single sourcing' describes the tendency to reduce the number of suppliers in order to lower complexity in the production chain and increase the price pressure through larger supply volumes.

Current trends favor producers who have large know-how bases, produce in large numbers and at the best locations, but who can still adopt quickly to changing technologies and demands. Virtual enterprises promise to offer the best of both worlds, flexibility and economy of scale. They are networks of legally and economically independent enterprises, each concentrating on its core competencies and out-sourcing the rest, modeled on the best-of-everything organization. The enterprises work together in vertical or horizontal structures and coordinate their actions with the help of information and communication technology. Working in vertical structures means the output of one enterprise is the input of the other; enterprises working in horizontal structures produce on the same level of the supply chain. The virtual enterprise appears and acts like a single enterprise to the outside. There is no physical institutionalization of central management functions. The contracts defining the relationships between the participating enterprises are deliberately left loose, in order to facilitate quick formation and greater flexibility in reorganization. Risks and costs tend to be distributed among all partners.

Cooperation

A cooperation can be defined by the working together of a group of independent, equally positioned enterprises with comparable financial power. The basis is a relationship of 'equals'. They differ from virtual enterprise in that the relationships between their members is based on long-term contracts and in that there is a central organ. The function of this central organ is restricted to operative management. Individual enterprises can be members of several cooperations. They retain the option to exit any time. The emphasis of inter-organizational coordination is on trust-building mechanisms like gift-giving.

Strategic Network

Strategic networks differ from cooperations in that they use stronger legal contracts, and feature a *hub firm* that "sets up the network, and takes a pro-active attitude in the care of it" [Jarillo, 1988]. The hub firm in a strategic network has more authority than the central organ of a cooperation and is usually significantly larger than the other members of the network. It coordinates activities in the strategic network, but the members retain their legal independence and autonomy. This network arrangement allows a participating firm to specialize in those activities of the value chain that are essential to its competitive advantage, reaping all the benefits of specialization, focus and, possibly, size. Membership in several strategic networks is possible, and firms have the right to leave the network. The time frame and financial volume are usually larger than in the case of virtual enterprises and cooperations.

Group

Groups are formed from enterprises that retain their legal independence, but are bound by contract to the authority of the central firm [Freichel, 1992]. In contrast to the strategic network, no multiple memberships are allowed, and there usually is no exit option for subordinate firms. All economic activities are focused on the group and subject to directions from the head enterprise. The interdependency between the firms is found in an authoritative hierarchy, whereas in strategic networks, it is based on economic relationships.

Corporation

The corporation is the result of the complete inclusion of all legal and economic aspects of the original companies into a new entity. This organizational form is at the hierarchical end of the spectrum market-hierarchy. Companies merging into a corporation give up all of their autonomy. The process is usually not reversible; once inside a corporation, the former status can not be regained. In the business world, the process of merging usually happens when a large company assimilates a much smaller one.

2.2 Holons

2.2.1 Historical Background

The term "holon" has been introduced by Koestler in 1968 in his book "The Ghost in the Machine" [Koestler, 1967]. Koestler observed that complex systems, be they natural or man-made, are hierarchically composed of less complex parts. The reason for this he sees in that any system needs to be able to recover from disturbances in the course of its development and further existence. Errors in monolithic structures affect the whole system, making recovery very costly. The potential for such disturbances increases with the system's complexity, hence more complex systems tend to be built from stable intermediate forms.

The components of a system have a functional stability of their own and can be decomposed into parts of lesser complexity. Therefore, they can be seen as parts and wholes at the same time. Most entities are such "inner nodes" in trees of hierarchical structures: human cells, insects in a colony, or departments in an institution. Koestler calls these inner nodes "holons", from the Greek word "holos" for whole and suffix "on" for part.

The two roles of the holon require it to balance two contradictory forces: the force of separation and the force of cohesion. This balancing property can be found on multiple resolution levels of the system which the holon is a part of, as it replicates in self-similar structures throughout the hierarchy. Interaction between different holons is constrained to paths along the hierarchical structure.

For example, workers of different departments do not communicate directly with each other, but via their respective departments' heads.

The properties exhibited by holons are the reason why Koestler's terminology is used today in several fields of technology: the importance of balance between autonomy and cooperation, tractable complexity through self-similar structures at different levels, and efficient functioning through controlled interaction has been pointed out before.

2.2.2 Holons Today

One of the most promising applications of the holonic concept is in the area of manufacturing. Increasingly, the requirements for modern manufacturing systems demand flexibility in reacting to changing consumer demand, resilience to unexpected disturbances, and efficiency in production. The idea is to organize manufacturing units into recursive, reconfigurable hierarchies of holons. Every holon has a certain degree of autonomy in that it can build and execute its own plans, but it can also cooperate with other holons to form a superholon and let the superholon accomplish tasks that none of its constituting subholons could do alone. A flexible organization like this is called a holarchy.

In order to make holarchies a viable alternative to traditional manufacturing systems, the Holonic Manufacturing Systems (HMS) consortium has been created in reaction to [Suda, 1989, Suda, 1990], who first proposed the application of holons to manufacturing. Its task is to do the necessary research and set the required standards to integrate machines and humans in autonomous, self-reliant units, which interact together to build a holarchy.

Other applications of holons are in the field of business interaction, for example the coordination of supply webs [Gerber et al., 2001] or the creation of virtual enterprises [Ulieru et al., 2001]. Just like manufacturers, businesses face quicker change in consumer demands and stronger competition requiring higher efficiency. In light of increasingly complex networks of suppliers and customers new mechanisms of cooperation are called for, to answer the challenges created by the rising use of new technologies like the Internet. Modeling these new mechanisms after the example of biological and social systems in the forms of holarchies might provide the needed flexibility and efficiency.

2.3 Self-Organization

According to Parunak, in the ultimate agent vision "the application developer simply identifies the agents desired in the final application, and they organize themselves to perform the required functionality" [Parunak, 1997]. Self-organization is clearly a promising field for computer scientists working with multiagent systems, but we will see in this section that its understanding is based on an intuitive no-

tion, as it still lacks a clear definition. Self-organizing systems have a number of desirable properties that are behind the motivation of building self-organization into software systems to help coping with the increasing complexity of software. We will describe some of these properties, and conclude this section with related work that tries to capture some of these properties in multiagent systems.

2.3.1 Definition

The term *self-organization* was introduced in 1947 by the psychiatrist and cybernetics researcher Ross Ashby [Ashby, 1947], but to this day there is no officially accepted definition of what constitutes a self-organizing system. One reason for this is that the intuitive understanding of the term–that self-organizing systems are systems that appear to organize themselves without external direction, manipulation, or control–is difficult to formalize without running into contradictions. For example, insect societies like ant colonies are widely regarded as self-organizing systems, but their structure and behavior is clearly dependent on external circumstances; the ant colony adapts to its environment to maximize its chances of survival. A definition that is suitable for systems like ant colonies in that it expresses self-organization in terms of goals and performance is proposed by Klir [Klir, 1991]:

Definition 1 (Self-organizing System according to Klir, 1991)
A **self-organizing system** *is a system that tends to improve its performance in the course of time by making its elements better organized for achieving the goal.*

The problem with this definition is that it is not clear what the concepts goal and performance mean in the context of physical or geological systems where the term self-organization has been applied as well. However, this thesis deals with multiagent systems that are inspired by sociological and biological concepts, for which the given definition is sufficient. A more generally applicable definition would emphasize that the system structure often appears without explicit pressure or involvement from outside the system and that its constraints are internal to the system, resulting from the interaction among the components.

2.3.2 Characteristics of Self-Organizing Systems

The characteristic that is most strongly associated with self-organizing systems is *emergence*. Emergence is the formation of a coherent pattern that arises out of interactions of the system's components. This pattern can be quite complex and useful to the system, even though the individual components act according to very simple rules. The system's behavior looks like it was designed in a top-down fashion, whereas it really originates bottom-up from many simple interactions. Insect states, especially ant colonies, are the prototypical example for systems

with emerging properties. Each individual ant has a very simple "program", yet the colony as a whole is capable of adapting to a wide range of environmental circumstances. For example, it is important for the efficient functioning of the colony that ants find nearby food sources and lay out short paths between their nest and the food. The colony is actually able to build minimum spanning trees that connect the nest to the food by each ant following this simple program [Parunak, 1997]:

- Avoid obstacles
- Wander randomly, or towards the general direction of nearby pheromones if any.
- If holding food, drop pheromone at constant rate.
- If food and not holding any then pick it up.
- If at nest and holding food then drop it.

Emergence is often a desirable property when engineering complex systems, because designing a template for a simple unit and creating a system from interacting units built from this template is much easier than hard-coding complex behavior into the system in a top-down fashion. Two further desirable properties of self-organizing systems are *homeostasis*, which is the capability to return to a steady state after disturbance, and *homeorhesis*, the ability to seek out new developmental pathways through successive instabilities [Sahal, 1979].

2.3.3 Self-organization in Multiagent Systems

The characteristics of self-organizing systems described above suggest that self-organization might be a useful paradigm for building software systems that are applicable to a wide range of complex tasks. One argument for multiagent systems is that agents might be just at the right level of complexity to constitute the interacting elements of self-organizing systems. André et al. have built a centralized MAS that orders its agents in a hierarchic graph of service decomposition and that is capable of dynamic reorganization by changing the decomposition depending on the performance of the current structure [André et al., 1990]. Turner and Jennings argue that multiagent systems need to be both self-building and adaptive if they have to be scalable and if the number of agents can change during runtime [Turner and Jennings, 2000]. By self-building they mean the agents' ability to determine the most appropriate organizational structure for the themselves at runtime, and by adaptive they mean the ability to change this structure as their environment changes. Brooks and Durfee show that congregation, the self-organization of the system into smaller groups of agents, can help allocating problems scale to large populations by allowing agents to interact locally [Brooks and Durfee, 2002].

2.4 Software Agents

Agent technology has been gaining increasing popularity among researchers in the last decade. One reason for this is that many see it as a new step in the evolution of programming approaches, promising to let us develop software systems that are too complex to handle with earlier paradigms, like object-oriented programming (OOP). We introduce the concept of agents by describing it in terms of such an evolutionary step in the first section. The second section presents arguments for the use of agent technology. Agents are certainly not always the best way to implement all kinds of software systems, but they are a promising approach for certain classes of problems. The third section deals with a more formal definition of agents, but can only concentrate on a few suggestions used in the literature, as there still is no universally accepted definition of the term "agent". Finally, the fourth section gives an overview over some popular architectures for software agents, showing that agents vary widely in their level of sophistication.

2.4.1 Comparison with Object-Oriented Programming

In our overview picture at the beginning of this chapter we suggested that software agents are the product of applying concepts from real-world agents to object technology. A good way to introduce agents is therefore to compare them to objects and show what they have in common and where they differ. According to Parunak, both programming paradigms emphasize a modular unit behavior and an internal unit state [Parunak, 1997]. The crucial difference is that objects are invoked by external messages, whereas agents can be invoked internally by rules or goals. In this sense, agents are *autonomous*: they can react not only to specific method invocations, but also to observable events within the environment. They can actually poll the environment for events and other messages to determine what action they should take. This property makes them *proactive*, in contrast to objects, which are conventionally passive, with their methods being invoked under a caller's thread of control [Odell, 2000b].

A further important difference between agents and objects is that in order to program with objects, the programmer needs no know in advance the details of the objects' interfaces. In contrast, agents can employ other mechanisms, such as advertising their services and capabilities in directories that other software can query. This advantage in interoperability is one of the major arguments for using agents in the next section.

2.4.2 Motivation

Due to the increasing number of people using information technology and rising difficulty of tasks required from programs, software systems need to become more and more complex and require the working together of different modules.

Jennings argues that conventional technologies are not well suited to deal with such complex, distributed systems, because (i) basic building blocks are too fine grained, (ii) interactions are too rigidly defined, and (iii) insufficient mechanisms are available for dealing with organizational structure [Jennings, 1999].

Genesereth sees the major problem in that there is an increasing number of programs that need to exchange information and services with other programs to solve the tasks assigned to them; they need to inter-operate [Genesereth and Ketchpel, 1997]. However, programs are written by different people with different applications in mind. Their interfaces are therefore heterogenous and each program can only communicate with a very specific class of other programs.

Apart from being designed with different interfaces, programs are the product of different stakeholders with different aims. Hence another barrier for interoperability is the need for programs to cooperate even if their represented interests do not completely coincide [Panzarasa and Jennings, 2001]. Access to information and operations might be restricted to local entities, requiring distributed, non-centralized structures of encapsulated units with weak links connecting each other.

Agent technology promises to alleviate these difficulties by copying the solutions nature has come up with to solve similar problems: agents can represent interests and goals explicitly and therefore take their own interests and that of other agents into account when deciding what to do; they communicate via standardized messages that still leave enough flexibility to exchange information about all required tasks; finally, they encapsulate services of the right granularity to represent flexible building blocks for dynamic complex systems.

2.4.3 Definition

There is currently no universally agreed-on definition of *agent*, partly because the term has been used outside the field of computer science in many different contexts. Russell and Norvig's widely cited definition reflects this wide applicability of the term [Russell and Norvig, 1995]:

Definition 2 (Agent according to Russel & Norvig, 1995)
An **agent** *is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.*

A definition of agency that is more specific and more useful for the purposes of this thesis is provided by Wooldridge and Jennings. A computer system satisfies their **weak notion of agency** if it has the following properties [Wooldridge and Jennings, 1994]:

Autonomy The system should be able to act without the direct intervention of humans (or other agents), and should have control over its own actions and internal state.

Reactivity The system should perceive its environment, and respond to changes in it to reach its goals.

Sociality The system should be able to interact with other systems exhibiting agency, in order to complete its own problem solving and to help others with their activities.

Proactivity The system should not just act in response to the environment but should be able to exhibit opportunistic, goal-directed behavior and take the initiative where appropriate.

These characteristics are exhibited by most systems that are called agents, but other authors have suggested further properties that, although not necessary to call a system an agent, are strongly connected with many agent applications today [Odell, 2000a]: adaptivity, mobility, rationality, unpredictability, credibility, transparency and accountability, coordinativity, cooperativity, competitiveness, robustness towards errors and incomplete data, and trustworthiness.

2.4.4 Architectures

Having discussed what agents are and why we might need them, we now turn to the question of how to construct agents that have the required properties that we expect from them. What software or hardware structures are appropriate, in other words, what architecture should we use for the agents? There are three different approaches to this question: reactive agents, who have a very tight coupling of perception and action, deliberate agents, who tend to do a significant amount of reasoning before acting, and hybrid agents, who fall between the first two approaches.

Reactive Agents

Reactive agents have at most a very simple internal representation of the world. Advocates of this approach argue that intelligence is a product of interaction between the agent and its environment and that a direct mapping of perceptual input to action output is preferable to abstract reasoning. A prominent example of a reactive architecture is Brook's subsumption architecture [Brooks, 1986]. A major problem with this approach is the difficulty of building large systems that have enough flexibility to handle new situations which the designers did not explicitly hand-code behavior for.

Deliberate Agents

Deliberate agents build a model of the world from their sensory input, reason inside this model how best to reach their goals, create a plan as a result from this

reasoning, and base their actions on this plan. An advantage of this approach when compared to the reactive agents is that the range of behavior is far larger than what has been explicitely coded into the system. Examples are Shoham's Agent-Oriented Programming [Shoham, 1993] and the BDI architecture [Bratman et al., 1988]. BDI agents represent their state by three structures: their beliefs, their desires, and their intentions. An agent's belief is its model of the domain, its desires order the possible states of the world according to the agent's preferences, and its intentions are the actions the agent has decided to do in order to satisfy its desires. Major problems of agents with deliberate architectures are the time delay between perception and action and the complexity of the world, which makes it difficult to plan successfully outside a laboratory environment.

Hybrid Agents

Hybrid architectures have been proposed that combine reactive and deliberate behavior to avoid the problems mentioned above. These architectures usually consist of several subsystems, some of which function as deliberate components that reason symbolically about the world and create plans, and others which react quickly to the environment without complex processing. An example of such a system architecture is InteRRaP [Müller and Pischel, 1993]. This architecture has been created for agents that are members of complex agent societies, who need both to be able to react quickly to environmental changes as well as to reason about other agents and coordinate their actions with them. Its planning component contains therefore an explicit coordinative planning layer that complements the local planning layer and the behavior-based layer. The InteRRaP architecture will be introduced in more detail in the section on Holonic Multiagent Systems.

2.5 Multiagent Systems

Multiagent systems (MAS) have historically been one of two parts of distributed artificial intelligence, the other being distributed problem solving (DPS) [Durfee et al., 1989]. The understanding was that the agents in a DPS cooperate to solve a computational problem. All have the same goal and the same interests, perhaps because they were designed by the same source. In contrast, MAS can consist of interacting entities that represent parties with different interests, interactions include competition as well as cooperation. An example of a scenario where agents cooperate with members of their own team and compete with those of the other team is RoboCup [RoboCup, 2002]. Today, the terms have changed in that distributed artificial intelligence is more or less synonymous with MAS, and DPS has become a subfield of multiagent systems. Jennings et al. define MAS as systems that have four properties [Jennings et al., 1998]:

Definition 3 (Multiagent System according to Jennigs, 1998)
A **multiagent system** *is a system that has these four properties:*

- *each agent has incomplete capabilities to solve a problem*
- *there is no global system control*
- *data is decentralized*
- *computation is asynchronous*

Environments that support such systems usually provide an infrastructure specifying communication and interaction protocols. A definition that emphasizes the sociological aspect of MAS is given by Panzarasa and Jennings: "A MAS is a social and cognitive entity, with a relatively identifiable boundary, that functions on a relatively continuous basis through the coordination of loosely interdependent, cognitive and autonomous agents" [Panzarasa and Jennings, 2001]. We will return to the two approaches to this field, the computational approach and the social approach, in the first section where we motivate the use of multiagent systems. The second section deals with the question of interaction between the entities constitution such a system, presenting an overview over some mechanisms used by agents to coordinate their activities to reach their goals. The third sections describes the problems agents have to deal with when relationships between them go beyond short-term interactions and form the basis of coalitions. Section four investigates the communication requirements that an agent platform needs to fulfill to make all these interactions possible. Finally, the fifth section describes the FIPA standard of agent communication, the standard that has been used in the experimental system for evaluation of the concepts proposed by this thesis.

2.5.1 Motivation

According to Weiß, there are two main reasons which drive forces behind the growth of the MAS paradigm in recent years: one related to the field of computers, the other related to the field of sociology [Weiß, 1999a].

The first has to do with the fact that modern computing platforms and information environments are distributed, large, open, and heterogenous. Computers have become closely connected both with each other and their users, they are no longer stand-alone systems. MAS promise to provide technology that is more suitable to the demands of the Internet, telecommunications, e-commerce, etc.

The second reason is the potential of MAS to model human societies and test theories about human interactions. Examining the results of experiments of social simulations is an increasingly popular way to investigate the validity of assumptions on which social models are based.

Distributed systems provide several advantages to computational applications and human societies: they can enable the cooperation of entities whose knowledge, capabilites or services are encapsulated for reasons like spatial separation or privacy and intellectual ownership. The modular creation of structures provides the system with more flexibility, redundancy, modifiability, and extensibility. It can also increase the system's speed and efficiency, as central units in non-distributed systems often present a bottleneck to the system's performance.

2.5.2 Interaction

Agents may need to coordinate their actions in cooperative as well as in competitive settings. Parunak et al. see as the basis for analyzing the interactions of agents the *correlation*, which is defined as non-zero joint information over a population of agents [Parunak and Brueckner, 2002]. This measure is a purely behavioral notion, taking into account only the actions of agents. A derived measure that emphasizes information is *coordination*, which is correlation with a focus on the information flow that enables it. If the agent's inner states are taken into account, the system can be analyzed in terms of *cooperation*, the correlation modulated by the intent of individual agents. The authors also define *congruence* as the degree to which an agent system aligns with a system-level goal.

The above mentioned information flow that is the basis of coordination between agents can be interpreted as negotiations. The result of negotiations is a promise of the agents involved to try to reach a certain goal state. Cohen and Levesque called this promise *commitment* and defined it as as relative and persistent goal [Cohen and Levesque, 1987]. Agents want other agents to commit if they want to *delegate* a task to others (which means that the other agent promises to do the task delegated to him) or if they want others to *adopt* some of their own goals [Castelfranchi and Falcone, 1998].

There are several mechanisms to delegate via negotiation, which mechanism is most suitable for a given situation depends on the scenario. A common scenario that is applicable to many settings like e-commerce on the Internet or task assignment and resource allocation is the market, where some agents are *buyers*, others are *sellers*. Guttman and Maes have proposed a classification scheme for such scenarios (Figure 2.3) [Guttman and Maes, 1998].

In this thesis, we are interested in applying the market framework to a system for task assignment. Some agents have a task they need to be done by other agents and they want to delegate their task to the agents who can do it cheapest. They can find those agents by starting auctions, as the market framework suggests, or they can use a contract net protocol that has been developed for task assignments. We defer a discussion of the contract net protocol to chapter Four, where it is described in more detail in the context of developing the protocols used for this thesis. Here we will restrict ourselves to giving a short overview of auctions.

Some of the most common auction mechanisms are the English auction, the

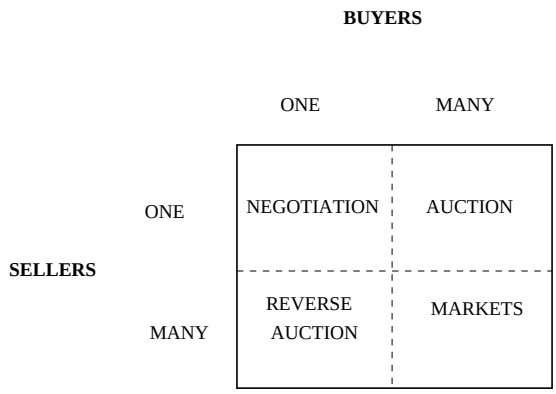

Figure 2.3: Market framework [Guttman and Maes, 1998].

Dutch auction, and the Vickrey auction. In the English auction, each bidder is free to raise his bid. If no more bidder wants to raise, the auction ends with the highest bidder winning the item at the price of his last bid. In the Dutch auction, the auctioneer continuously lowers the price until a bidder takes the item at the current price. Finally, in the Vickrey auction each bidder submits a bid without knowing the others' bids. The highest bidder wins the item at the highest losing price.

2.5.3 Coalitions

The last section dealt with the problem of coordinating actions with other agents for a single task or problem. Agents can increase their efficiency or profit beyond such short-term cooperations by forming coalitions with other agents. Coalitions are characterized by long-term involvement; agents commit not to single actions but to participation in solving a whole class of problems or tasks. This is a different problem from allocating tasks, because the value of having an agent participate in a coalition might be more difficult to assess than the value of a single service or resource. Sandholm has identified three questions that need to be tackled in systems forming coalitions: [Sandholm, 1996]

Coalition Structure Generation Which agents form a coalition together? Sandholm examines the case of exhaustive and disjoint coalitions. In such scenarios, agents coordinate their actions only with agents inside their own coalition, but not with other coalitions. We do not require this condition in our system, but Sandholm's questions still apply.

Solving the Optimization Problem of each Coalition Given a coalition consisting of a set of agents and a task to be solved, how is the task to be divided among the members of the coalition such that it can be completed successfully and most efficiently? In most settings, efficiently means with the greatest profit, measured in monetary value.

Divide the Profit If a task has been completed and the coalition has received a profit from it, how is the profit to be distributed among its agents? This question has a direct bearing on the other two, because when forming coalitions, agents consider the expected cost and income from participating in a proposed coalition.

Coalition formation has been widely studied (see Sandholm's dissertation for references [Sandholm, 1996]), but mostly under the condition that agents are rational and have complete information about the system. In our setting, agents have only very limited knowledge of other agents, so most mechanisms of finding the best coalitions cannot be applied in their full form. We will describe how agents find other agents to form long-term relationships with later in chapter Four.

2.5.4 Communication

One of the motivations for using agent technology mentioned in this chapter was that agents facilitate the interoperability between software systems built by different parties. However, this interoperability requires that the agents share the same means of communication, a common standard for information exchange. In other words, there must be a standard language for agents. There have been several efforts to design an agent communication language (ACL) that enables agents from different designers and on different platforms to exchange information, without restricting the type of content of this information. This is done by building ACLs as *wrapper languages* which implement a knowledge-level protocol that is unaware of the choice of content language and ontology specification mechanism. Most ACLs today are inspired by the linguistic theory of speech acts developed by Searle [Searle, 1969].

Speech act theory provides language primitives that are categorized depending on, among other things, the intent of the speaker and the effect on the listener. An example of a category of speech acts are *commissives*, whose meaning is to commit the speaker to a future course of action. Adapting a human knowledge-level communication protocol like speech acts to agent communication was motivated by the similarity of software agent to real agents and by the possibility of real humans taking the role of agents in agent systems. Two popular standard ACLs based on speech act theory are KQML [Finin et al., 1994] and FIPA ACL [FIPA, 2002].

KQML, which stands for Knowledge Query and Manipulation Language, is a language and set of conventions that support network programming specifically for knowledge-based systems and agents. FIPA ACL, with FIPA standing for Foundation for Intelligent Physical Agents, is a newer ACL which differs from KQML mainly in that its semantics does not allow agents to directly manipulate the knowledge base of other agents (they can no longer insert facts into oth-

ers' databases by sending a performative "tell(Agent5, Fact3)"), and that many functions of the wrapper language have been delegated to the content language.

Both ACLs provide three levels of protocol: the communication level, the message level, and the content level. The communication level specifies information like sender, receiver, content language, and ontology. This makes it possible to route the message to the right recipient and lets him process it with the right syntax rules (specified by the content language) and domain knowledge (specified by the ontology). The message level contains the kind of performative that tells the agent whether the message is a request, query, promise, etc. Finally, the content level contains the specific facts requested or promised.

The two ACLs also have in common the structuring of interactions in protocols. Protocols are patterns of interaction that are formally defined and abstracted from any particular sequence of execution steps. They specify what performatives are allowed and in what order they must occur. We will give an example of an interaction protocol in the next section, where we describe the FIPA standard in more detail, which is the standard our implementation is based upon.

2.5.5 FIPA

FIPA stands for *Foundation for Intelligent Physical Agents*, but the initial vision of physical agents has been replaced by the focus on software agents, communication and interoperability between agents, specification of external behavior, and use in heterogenous environments. This shift is reflected in the new in-official meaning of the acronym: Foundation for InteroPerable Agents. FIPA is a joint effort of several organizations and companies started in 1995 to provide a standard for agent communication that allows interoperability of agents based on different platforms. The FIPA organization publishes specifications that prescribe the interfaces that agents must implement in order to operate on FIPA-compliant agent systems. The specifications provide standards in the areas of agent communication, agent management, and agent-software integration.

Agent Communication

FIPA has specified the agent communication language FIPA ACL to support the negotiation, cooperation and information exchange between agents. The structure of a FIPA ACL message consists of five levels:

Protocol The protocol defines the social rules for structuring the communication between agents. An example is the contract net protocol for task assignment described in the chapter Specification.

Communicative Act The communicative act defines the performative of the message, for example request, query, or agree.

```
(inform
   :sender (agent−identifier :name i)
   :receiver (set (agent−identifier : name j))
   :content
      "weather (today, raining)"
   :language Prolog)
```

Figure 2.4: Example of a FIPA ACL message.

Messaging This level contains meta-information about the message including the identity of the sending agent and the receiving agent.

Content Language This specifies the gammer and semantics of the language the message's content is expressed in, for example, LISP.

Ontology The ontology defines the vocabulary and meaning of the terms and concepts used in the content.

.

Figure 2.4 gives an example of a FIPA ACL message by which agent i informs agent j that it is raining today.

Conversations and dialogues between agents are usually structured according to predefined patterns of messages, the interaction protocols. This facilitates the design of agents in that it allows the anticipation of the type of next message in a conversation. Figure 4.7 shows the *Contract Net Protocol* used for task assignment.

Agent designers can build their own protocols or even let their agents communicate without any protocols at all, but for the benefit of better interoperability they are encouraged to use the standard protocols.

Agent Management

The agent management specified by FIPA consists of the agent management system (AMS), the directory facilitator (DF), and the agent communication channel (ACC).

The agent management system keeps an index for all agents currently registered on its platform. This index includes, among other things, the identifier of its current platform and its identifier within that platform. The AMS is actually an agent that supervises the platform by controlling the creation, deletion, registration of agents and their migration to other platforms.

The directory facilitator is another agent that provides a "yellow pages" service to other agents. Agents may register their services at the DF or query the DF for agents providing specific services.

The agent communication channel is an agent that routes messages between agents on the same platform and between its platform and other platforms. The minimum requirement for FIPA compliance is the support of the Internet inter-orb protocol (IIOP) between different agent platforms.

Agent-Software Integration

This part of the FIPA specification describes a common way to encapsulate non-agent software and allow agents to access it as agents. This is done by creating wrapper agents that act as interface between the software and the agents on the FIPA platform.

2.6 Holonic Multiagent Systems

In this section we apply the concept of holons to agent technology. There have been proposals to merge ideas from holonic manufacturing systems and multiagent systems [Bussmann, 1998, Fischer, 1999a]. Holonic multiagent systems (HMAS) can be seen as the realization of this idea. We first outline the notion of a holonic agent, then describe systems consisting of such entities, continue with an architecture proposed for implementation of holonic agents, and finally give an example for an application using holonic multiagent systems.

2.6.1 Holonic Agents

A holonic agent consists of a group of several agents that together retain the properties of a single agent [C. Gerber, 1999]. These agents can be holons themselves, in which case they are called *subholons* of the original holon. The concept of holon can therefore be applied to build recursive structures in agent groups. Holonic agents have, like normal agents, a unique identification that makes it possible to communicate with them with the normal agent communication methods, for example protocols [Fischer, 1999b]. Holonic agents need to be addressable as single agents. One way of realizing this is to let all communication to the outside run through one of the agents forming the holon. Such an agent is called *head* of the holon or *mediator* [Ulieru et al., 2001]. The binding force that keeps agents in a holon together can be understood as commitments [Singh, 1997]. The agents stay together to cooperate and work towards a common goal. Holons can therefore be seen as the recursive application of the concept of coalitions described in the section on multiagent systems.

2.6.2 Holonic Structures in Agent Systems

As mentioned above, agents form holons by ensuring cooperation via commitments. By committing, they limit their possible future actions and therefore give

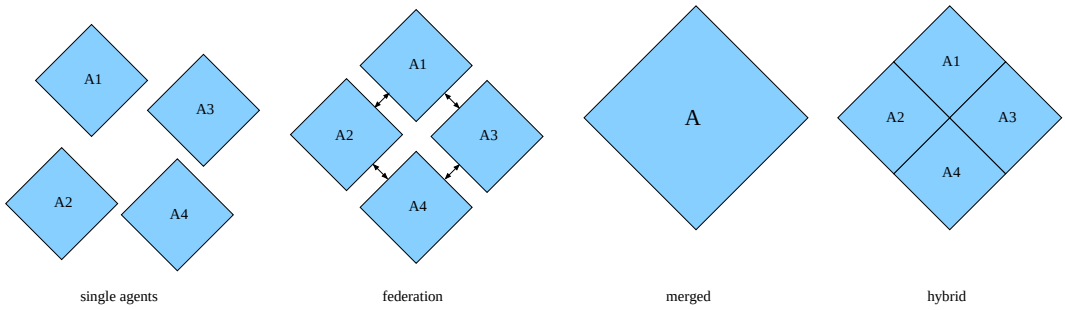

Figure 2.5: Different forms of holonic agents [C. Gerber, 1999].

up part of their autonomy. The degree to which they give up autonomy is not set in advance but depends on the circumstances and is subject to negotiation between the agents participating in the holon [Schillo et al., 2001b]. The least sacrifice of autonomy can be seen in holons forming a loose *federation* of agents (see Figure 2.5). The long-term commitment in this form is actually so low that agents need to negotiate their coordination on a case-by-case basis. The federation is at one end of the autonomy spectrum and does not differ significantly from conventional multiagent systems. At the other end of the spectrum, agents can give up all of their autonomy and merge into a single agent. How this merging is realized depends on the agents' architecture. In BDI agents, for example, the belief of the resulting holon is the union of the beliefs of the subholons, provided consistency is guaranteed. The same applies to its goals. Between the two extremes, there can be hybrid forms that let agents retain part of their autonomy, but not all of it. This thesis proposes a number of hybrid forms motivated by business organizational structures found in human societies and investigates their properties in self-organizing scenarios.

2.6.3 Architecture for Holonic Agents

Gerber et al. propose to implement holonic agents in Müller's three-layered InteRRaP architecture [C. Gerber, 1999]. As has been described in the section on software agents, InteRRaP is a hybrid architecture that combines properties of reactive and deliberate agents. The reactive component is realized by the behavior-based layer (BBL), which is responsible for the basic behaviors of the agent and its immediate reactions to simple stimuli from the environment within short time frames. The deliberate component is divided into the local planning layer (LPL) and the cooperative planning layer (CPL). The agent's knowledge base is structured similarly into three layers to correspond with the three control component layers just described. The LPL is responsible for planning tasks that only concern the agent itself and does not take into account interactions with other agents, which are handled by the CPL. The CPL contains the holonic

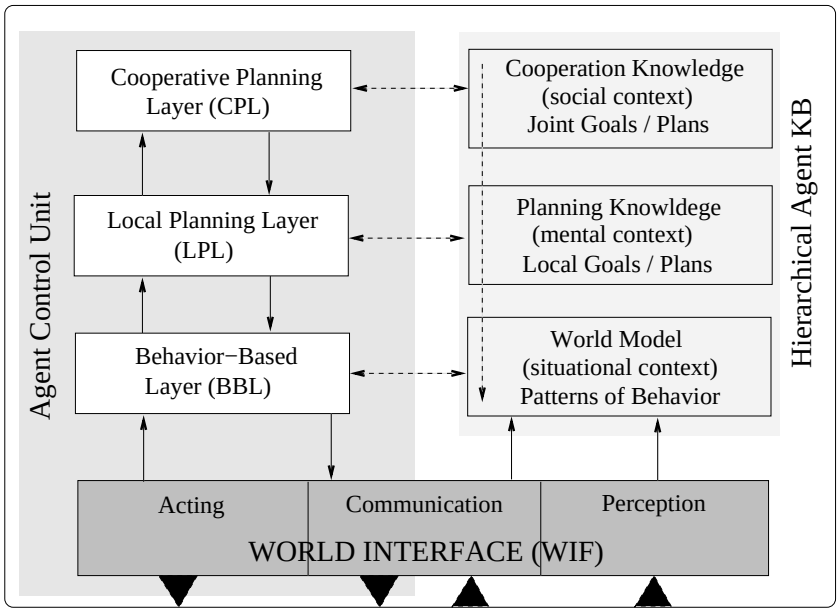

Figure 2.6: INTERRAP architecture [C. Gerber, 1999].

structure of the agent society, representing it as part of a directed graph of pointers, the complete graph being maintained in a distributed fashion by the CPLs of the member agents. The CPL is also responsible for coordinating the communication with other agents, either within the holon or between holons. The CPL of the head of a holon contains the information about the holon's resources and configurations, so it can take over administrative and representative functions. If the member agents of a holon are implemented on the same platform, they can share computational structures to increase the system's performance.

2.6.4 Example Application

Holonic multiagent systems have been applied in the area of logistics. The TELETRUCK system is an online scheduling system for truck fleets developed at the DFKI [Bürckert et al., 1998]. The transportation units in this system are divided into the categories drivers, trucks, trailers, and containers. Each unit can have its own objectives and is modeled as a single agent. In order to form a compound capable of performing a transportation task, a driver, a truck, a trailer, and containers can build a holon to form a complete vehicle. The different vehicles communicate via GPS with the Shipping Company, which uses a HMAS to manage the fleet schedules.

Chapter 3

Problem Description

In this chapter we describe the areas of research to which this thesis is contributing. The first section *Problem Statement* delineates the setting of the proposed concepts. In *Research Questions* we enumerate the three major research goals of this thesis. Finally, in *Applications*, we give a few examples of where the research results of this thesis might find application in real-world systems.

3.1 Problem Statement

This thesis is a contribution to the field of socionics. Of the three tenets of socionics, namely sociological reference (the use of computer models in sociological theorizing), computational reference (the development of new techniques and methods in distributed artificial intelligence and investigation of the role of sociological foundations in the construction of large-scale multiagent systems), and praxis reference (the examination of the social impact of hybrid artificial societies composed of both human beings and artificial agents), the thesis concentrates on the second tenet, computational reference. We are interested in the application of sociological models to multiagent systems.

The central setting for this work is a market of two kinds of agents, customers and providers. The customer agents have tasks they need to be done by provider agents. These tasks can be complex and may go beyond the capacities of any single agent, so that they require the collaboration of several providers. The customers try to find providers who can complete the tasks by auction mechanisms. The agents in this setting are self-interested entities that do not necessarily share a common goal. They can be designed and owned by different human parties, which limits the possibilities of global control of the task assignment scenario. Another important condition is that the market is not totally predictable; the provider agents do not know what the future orders of the customers will be, nor can they predict how the system will behave as a consequence of local interactions.

In order to find out how efficient this market works and to measure the changes

in performance that result from application of sociological concepts, we define a number of performance measures. These performance measures tell us what properties the system under examination has and which change in agent configuration and behavior has desirable consequences. Some of these performance measures are global measures, they measure the performance of the system as a whole, without looking at the level of individual agents. These measures are the number of messages sent in the system and the rate of failed task assignments. A system that uses few messages exchanged between agents for assigning the same number of tasks is considered to be more efficient than one that uses many messages. Similarly, a system is considered more efficient if it succeeds in assigning a larger number of tasks. The local performance measures are the profit per provider agent and the net income of groups of agents. These local measures evaluate the consequences of different actions from the point of view of single provider agents, and are therefore a valuable guideline for agent behavior.

If the system contains many agents or the tasks to be assigned require the collaboration of many providers, assigning the tasks with auction systems becomes increasingly complex and methods to improve the performance of the system as measured by the performance measures described above become more important. If the same type of task needs to be assigned several times or some parts of changing tasks are constant, the system might be more efficient if this repeating structure on the demand side was reflected by a similar repeating structure on the provider side, that is, if providers who are successful at completing a task or subtask together form relationships that facilitate long-term teamwork. This provider grouping can be formalized by the concept of organizations. Organizations are social structures that are nothing else but routines of conflict resolution resulting from previously resolved problems or conflicts [Gasser, 1991]. They institutionalize anticipated coordination.

Organizations can increase the efficiency of agents under certain conditions on the demand side. If these conditions change, organizations can actually worsen the performance of the system or of individual agents. This raises the problem of how to decide which agents should form what kind of organization. Pre-designing the organizational structure is often not possible because the designer does not know which demand will arise during run-time. We therefore investigate how the agents can decide themselves at run-time how to organize in the light of a given customer demand. They have to decide when to build new organizations, with whom to build them, and how to adapt them once they had a chance to learn about the efficieny of the organization.

3.2 Research Questions

In order to provide a contribution to the field of socionics with the concepts described above, three goals need to be met: we need to specify a number of

organizational forms and a mechanism for self-organization, we need to develop new protocols for the communication, and we need to implement a testbed and evaluate the effects of organizations and self-organization on the performance of the multiagent system.

3.2.1 Specification of Organizational Forms and a Mechanism for Self-organization

In order to apply sociological theories and models about organizations to multiagent systems, we need to select a number of organizational forms and give a detailed description of each that specifies the roles of the organization's member agents. The computational formalism we use to implement organizations is the concept of *holons*. Agents in a holon can have one of two roles: head or body agent. Each role has a number of rules that the agent assuming the role has to follow to ensure the efficient working of the organization. We provide a specification of the rules in the ADICO grammar, a sociological formalism for describing the rules and norms of organizations that is on the one hand flexible enough to express the rules required in our implementation and allow possible future enhancements, and on the other hand precise enough to be applicable in a computational system.

The ADICO grammar has to specify the allowed and required behavior of agents depending on their role in the organization. For example, the head of the organizational form *cooperation* has to start a new internal auction for each order that matches the product the cooperation was built for. Another thing that needs to be specified is the profit distribution, which details how the income an organization receives is divided among its members.

Once the organizational forms are specified, we need a mechanism for building new organizations and adapting existing ones to changing circumstances. The decisions must be made by the agents themselves on the basis of information available to them, they are *local* decisions. The creation and change of organizations always involves several agents and existing organizations might restrict the creation and change options of other organizations. For example, some organizational forms do not allow their members to be members of certain other organizations at the same time. Agents need to have a way of coordinating the creation and updating process taking conflicting requirements into account.

3.2.2 Development of Protocols

We structure the agent communication in our market scenario with a number of protocols. These protocols are based on the FIPA standard, described in the second chapter. The protocols already existing in the standard do not meet all of our requirements, as they were developed with single agents in mind and are not

optimal when applied to organizations. We therefore have to develop a number of extensions to existing protocols for inter- and intra-organizational communication. Part of this communication is auction related, like the assigning of tasks to other agents. Another part of it is self-organization related, for example the process of coordinating the change of an existing organization. Most new protocols developed in this thesis are not completely new ones but rather modify standard protocols, most notably the *contract net protocol*. All new protocols conform to the FIPA standard in that they only use performatives defined in the standard and are implementable on all agents compatible with the standard. The last point is important to ensure the extensibility of the current work and the interoperability with other projects.

3.2.3 Experimental Evaluation

It is not self-evident that applying sociological models of organizations in the form suggested here is really advantageous for the performance of multiagent systems. The third goal of this thesis is therefore to implement a working market scenario based on the proposed setting where the provider agents can be members of organizations and have the capability of self-organization. The research on this implementation includes varying the organization parameters of this implementation and observing the effects on the defined performance measures.

The testbed for the thesis has been implemented twice, once in Java and once in C++. The Java implementation was made to conform to the wide-spread practice of using Java for multiagent systems and to facilitate the implementation of the concepts used in the testbed in other systems. At the start of the experiments, however, it turned out that the Java implementation, which used the freely available FIPA-OS library as basic architecture for agent communication, is not fast enough. The speed and memory requirements of our scenarios limited the number of experiments that could be done in the available time below the limit for statistical significance. We therefore developed a second implementation, this time in C++. This implementation is significantly faster and allows more experiments to be done. Since the implementations do not differ in the auction mechanisms and agent reasoning, the theoretical results should be independent of the implementation.

Our experimental plan is structured around a number of hypotheses that relate organizational variables manipulated by the designer to the performance measures. The hypotheses reflect the expectation that organizations and self-organization are advantageous for the performance of multiagent systems.

The hypotheses are:

1. In scenarios where all organizations are of the same form (*homogenous scenarios*), the rate of failed orders in the system will be lower for more hierarchically oriented organizational forms.

2. In homogenous scenarios, the number of messages in the system will be lower for more hierarchically oriented organizational forms.

3. In homogenous scenarios, a stricter message limit (number of *calls for proposals* allowed to be sent per auction initiator) will increase the rate of failed orders in scenarios with single agents more than in scenarios with other organizational forms.

4. In scenarios where different organizational forms can coexist (*heterogenous scenarios*), the more hierarchically oriented organizational forms will have a higher net income.

5. In heterogenous scenarios, the presence of single agents will lower the net income of other organizational forms.

6. In heterogenous scenarios, a stricter message limit will reduce the net income of single agents more than that of other organizational forms.

7. In scenarios where agents start out as single agents and can self-organize to form new organizations, the rate of failed orders of the system will be lower than if self-organization is not allowed.

8. In self-organization scenarios where agents start out as single agents, the number of messages per system will be lower than if self-organization is not allowed.

9. In self-organization scenarios where agents start out as single agents, the profit per agent will be higher than if self-organization is not allowed.

3.3 Applications

The concepts of organizations and self-organization as proposed in this thesis have potential applications in scenarios where many agents interact, the tasks to be assigned require the collaboration of several agents, and where the tasks have repetitive structures or substructures. The initial motivation for this work was the sector of transportation and logistics. The tasks in this scenario consist of finding a transportation route for a freight from a given starting point to a given destination. Since the route might span a large territory, the transport might require the services of several transportation firms. For example, Company 1 might be responsible for transporting the freight from A to B, and Company 2 for getting it from B to C. Several transportation firms might be available for providing the transporting along a given sub-route. The scenario therefore contains both competition and cooperation opportunities. Transportation firms whose home territories are adjacent to each other could form an organization

whose home territory is the two territories combined. This organization might be more efficient than if the firms had to negotiate each new transportation order anew with each other.

Another field for potential applications of the proposed concepts is the Internet. For example, another Diplomarbeit at our faculty at the time of this writing investigates the use of multiagent systems in distributed spam filtering [Metzger et al., 2002]. Agents share knowledge about the classification of mails as spam or no-spam. It is conceivable that agents owned by users who share a certain affiliation could be made to work better together if they formed an organization. For example, each employee of the DFKI could have a spam filtering agent, and these agents could form an organization. Members of the same organization could trust each other more and give each other access to more information and services. A classification originating from an organization partner is potentially more reliable than one originating from an external agent. It is also possible that organizations are formed not on the basis of their users working in the same company, but on their users having the same mail preferences. The head agent of such a spam filtering organization would be responsible for filtering the classification proposals from outside, thus its body agents could save computational resources.

Chapter 4

Specification

This chapter describes the requirements that our system needs to meet to answer the questions posed in the last chapter. Our system is based on a market scenario where customers try to assign complex tasks to providers. The performance of the system can be measured in several ways by monitoring the values of variables during the simulation. Provider agents can form organizations that are motivated by examples of business models in human societies. This organizational structure can be set to change depending on the local decisions of the providers. The market scenario is described in more detail in the first section. The second section specifies the performance measures we have selected to evaluate the concepts proposed in this thesis. The third section formalizes the multiagent equivalents of the organizational forms introduced in the second chapter. The specficiation of the implementation for self-organization are given in the fifth section. Finally, section Six contains the protocols used by these organizational forms for inter- and intraorganizational communication.

4.1 Market Scenario

The market consists of two groups of agents: customers and providers. Time is modeled as discrete rounds. Each round, customers have the option to announce a task that can consist of several subtasks and whose deadline can be the current or a future round. Elementary tasks, those that do not consist of subtasks, are termed by single letters: A, B, etc. They can be combined to form composed orders, termed by strings whose letters specify the subtasks they are composed of. For example, a task ABC can be completed by completing tasks A, B, and C. Each task has a deadline that specifies the maximum number of rounds the completion of the task may take.

Each new round, the customers announce their tasks to the providers by initiating a new auction for each task. An auction is started by sending orders to a number of provider agents asking for completion of the task. The round ends

when all auctions have terminated. How providers react to calls for proposals sent to them depends on whether they have formed an organization with other providers or not and on the form of this organization. For example, a head of a *strategic network* who receives an order AB might ask two of its subordinate agents whether they can complete orders A and B, respectively.

Customers do only interact with providers, but not with each other. Providers, on the other hand, are free to interact with other providers. When starting new auctions, not all providers may be sent a call for proposals. There is a message limit that limits the number of providers that may be asked per auction. Unlimited auctions are not realistic, especially if the number of agents becomes significantly large. The height of this message limit is an independent variable that will be varied in the experiments. Since not all providers can be asked, agents starting an auction need to choose whom to include in the auction. They do this by assigning a value to each provider and asking the providers with the highest value. The value they compute is dependent on the specific task of the auction: the value is the sum of the volumes of elementary resource types they have assigned to that agent in the past, summed up over all elementary resource types in the order of the auction they want to start. For example, if they want to send call for proposals for type AB, the value of a provider who they have assigned 5 units of type A and 2 units of type B to in total in the previous rounds is 7.

Agents who process an order as single agents and not as members of an organization check how much of the order they can do themselves. If they do not have the resources for all of the types in the order, they start a new auction asking other providers to complete the parts of the task they cannot do by themselves. They start only one auction for each order, that is, if the agent received an order ABC and can only do A, it will start an auction asking for BC. It will not start two auctions one asking for A and the other for B, because this splitting of orders into their elementary types would make the system less efficient, as the system would no longer benefit from the advantage of organizations being able to efficiently process complex orders. Agents do not start new auctions if they cannot complete any part of the task. If they did, the round would not be guaranteed to terminate.

The providers are profit oriented. In the case of a single agent, the price it asks for its resources is the cost of the resource multiplied by an agent-specific profit parameter. In organizations, the profit is computed dependent on the organizational form. Section 4.3 describes the profit distribution for all implemented organizational forms.

4.2 Performance Measures

In order to find out to what extent the application of sociological concepts as we propose them are advantageous to multiagent systems, we need a way to measure

the value these concepts add to the system. Value is measured by measuring the performance of the system. We used four performance measures to evaluate aspects of behavior: rate of failed orders, profit per provider, net income per organizational form, and number of messages.

4.2.1 Rate of Failed Orders

One aspect we are interested in is how many of the tasks assignments succeed. This measure is probably of little value to provider agents, but it might be in the interest of the customers or the system user to have as few tasks assignments fail as possible. Multiagent systems that assign tasks by applying auction or contract-net mechanisms do not always find solutions that could in principle be found by a brute-force approach that checks all possible assignment configurations. Examples of such failures are discussed in section 6.1 The reason they are often preferable despite their incompleteness is that the brute-force approach has prohibitive computational demands and requires the centralization of all information about the system. Auction mechanisms can find solutions when the centralized approach fails because not enough computational resources are available or a centralization of information is not possible due to practical reasons or issues of privacy. One motivation behind the idea of using social organizations in multiagent systems is the hope that an appropriate organizational structure will decrease the number of failed task assignments if the system is in principle able to complete the task.

4.2.2 Profit per Provider

The rate of failed orders is a performance measure that is of interest to customers and perhaps the system user, but we should also consider the viewpoint of the provider agents. It is the providers who decide when to form what kind of organization, and the most important criterion they should base their decision on is the difference in profit that they receive. The second performance measure, profit per provider, therefore looks at profit on the provider side. This measure will be of interest in scenarios where self-organization is allowed, as we expect that agents who dynamically change their grouping based on the market situation increase their profit, as opposed to agents who do not self-organize.

4.2.3 Net Income per Organizational Form

The performance measure "net income per organizational form" looks at the income of all organizations of a specific kind, so we can measure how much money goes to what organizational form. This performance measure will be of interest in scenarios where different organizational forms exist at the same time and compete

with each other. If the number of organizations of each form is equal, organizational forms with higher net income can be interpreted to be more successful in competitive scenarios.

4.2.4 Number of Messages

Customers want to get their tasks assigned, providers want to maximize their profit; both might prefer to reach their goals with as little communication as possible, if communication is costly in terms of computational resources. In our FIPA-OS based Java implementation, for example, the sending, receiving, and routing of messages turned out to be the major bottleneck for the rate at which the rounds in the market could be simulated. Even if computational performance is not an issue, the agents or the system designer might be interested in keeping the number of messages necessary for assigning a task as low as possible, for example if agents have to pay for sending messages. This could be the case in multiagent systems that are modeled after real organizations and are used to test theories about the working of those organizations. Our final performance measure is therefore the number of messages that need to be exchanged between the agents in order to assign tasks. In general, the more hierarchical oriented an organization is, the fewer intra-organizational messages it needs to send to process an order, so we expect scenarios where agents increase the level of organization by self-organizing in order to adapt to the market situation to show a decrease in the number of total messages sent in the system.

4.3 Selected Organizational Forms

The six forms of business networks presented in section 2.1.3 have been selected to test their suitability for market-based multiagent systems. These forms can be ordered on a scale from maximum to minimum autonomy of the agents. This section describes the models used to implement the network forms in order of decreasing autonomy. For the sake of simplicity, at most three layers are depicted in the figures: customers, providers, and, where appropriate, intermediate agents.

Each of the six organizational forms used in the simulation is modeled with a set of rules stating what the organization's member agents are allowed to do, what they are obliged to do, and what they must not do. Each rule is given in its linguistic and ADICO form. We also describe the profit distribution of the organizational form, but for simplicity only in linguistic form, not in the ADICO grammar.

4.3.1 General Rules

The following rules apply always and have no reference to a specific organizational form. Most organizational forms are product-specific, which means that member agents receiving a call for proposal (cfp) only process the order with the rules of that organization if the order's type corresponds to the organization's specific type determined at the time of its creation. If the order is of a different type, the membership in the organization does not have any effect on the processing of the order. Product-specificity has been introduced because there needs to be a way for agents who are members of several organizations to tell which of their organizations is responsible for an incoming order. Membership in organizations created for the same order are not allowed, so it is always clear how to process an order.

- All customers may make cfps to providers.

A	all customers
D	permitted
I	make cfps to providers

- Providers must not be members of several product-specific organizations if these organization were created for the same type.

A	all providers
D	forbidden
I	be member of more than one organization
C	organizations have been created for the same type

- Providers must process orders that match the type of a product-specific organization of which they are a member with the rules of that organization.

A	all providers
D	obliged
I	process order as organization
C	order is specific to that orgization

- Providers that process an order as head of an organization can start an external auction asking for parts of the order only for the resources the organization cannot do by itself and if the organization can do at least part of the original order.

A	heads of an organization
D	forbidden
I	delegate to outside agents the complete order or subtypes that could be done internally

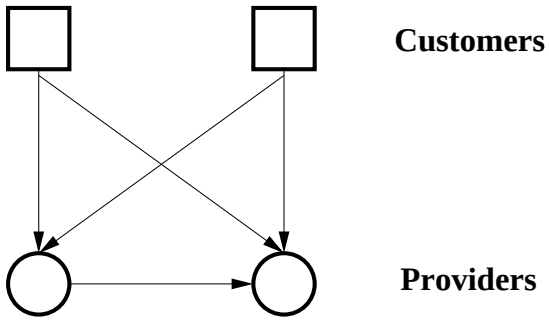

Figure 4.1: Task assignment in scenarios with single agents.

4.3.2 Single Agents

In this form of interaction, shown in figure 4.1, providers act as single agents. As was mentioned before, we forbid the delegation of the entire incoming order to prevent non-terminating rounds. Another limitation is that they can only start one auction for each incoming order, because otherwise they might start a new auction for each elementary type and diminish the efficiency of organizations that have advantages in processing composed, non-elementary orders.

- All providers acting as single agents may make cpfs to other providers, but they must not delegate all of a task.

A	all providers processing an order as single agents
D	permitted
I	delegate part of the order to other agents by starting a single auction
C	the delegated order must not contain all of the types of the original order

4.3.3 Virtual Enterprise

A virtual enterprise as shown in Figure 4.2 consists of provider agents with equal rights, there is no head agent designated in advance. A virtual enterprise is product-specific. Each member agent may accept cfps, but must start a new internal auction for each of its subtypes among its partners. This member agent becomes the head of the virtual enterprise for this specific order. This is the only organizational form that has order-specific heads. There is no specific profit distribution other than the normal economic exchange via the internal auctions.

- Providers that must process a cfp as virtual enterprise must start an internal auction for each of its subtypes among their virtual enterprise partners.

A	providers processing an order as virtual enterprise
D	obliged
I	start an internal auction for each subtype

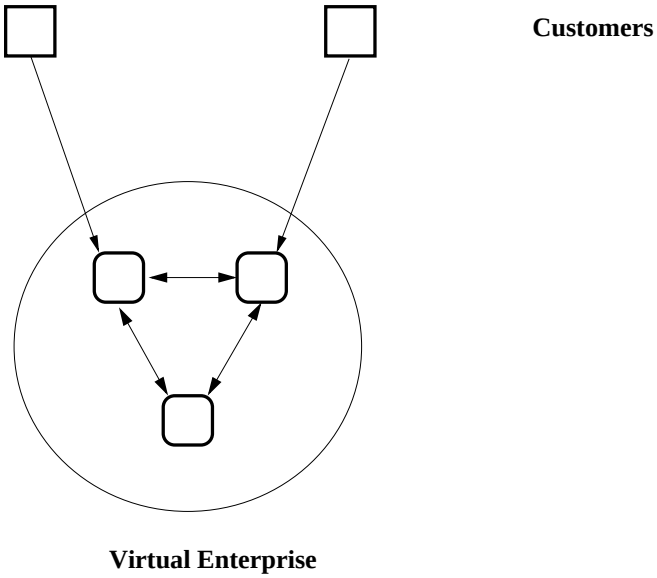

Figure 4.2: Task assignment in scenarios with a virtual enterprise.

4.3.4 Cooperation

Cooperations consist of a head agent and body agents. At the time of the creation of the cooperation, the member agents elect one of them to be the head of the organization. Cooperations are product-specific. If an incoming order matches the product of the cooperation, the rules applying to the receiving agent depend on whether it is the head or a body agent. Body agents may not directly accept cfps from outside the cooperation, but must *bounce* them. Bouncing means that they refuse the order, but send the name of their head inside the refusal message so the sender of the order can resend the cfp to the head instead. The semantics of this bouncing is that the head of the organization is the one responsible for the organization's interaction to the outside and all interorganizational communication must be channeled through him. If the cooperation head receives an order matching the cooperation type, it has to start an internal auction for each of its subtypes, just as the virtual enterprise head.

The head distributes the profit for orders completed by the cooperation according to a fixed ratio determined at the creation of the organization. Members of a cooperation may give gifts to each other. For the body agent, giving a gift means demanding a lower price for a good than the price that was initially agreed on; for the head, it means paying more than the agreed-on price. Agents remember their gift history with other agents in their models of these agents.

- Agents processing an external order as cooperation bodies must bounce the order.

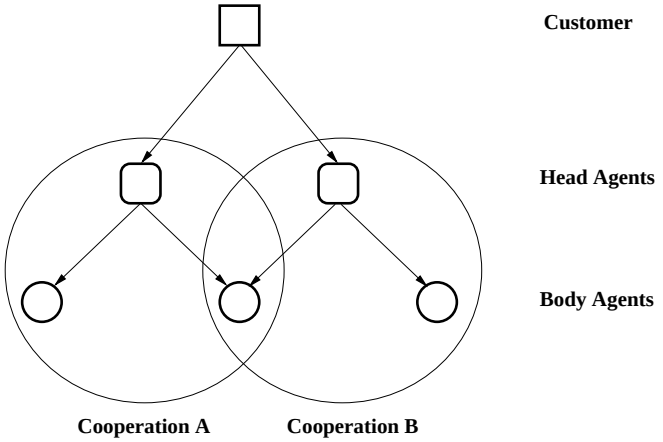

Figure 4.3: Task assigment in scenarios with cooperations.

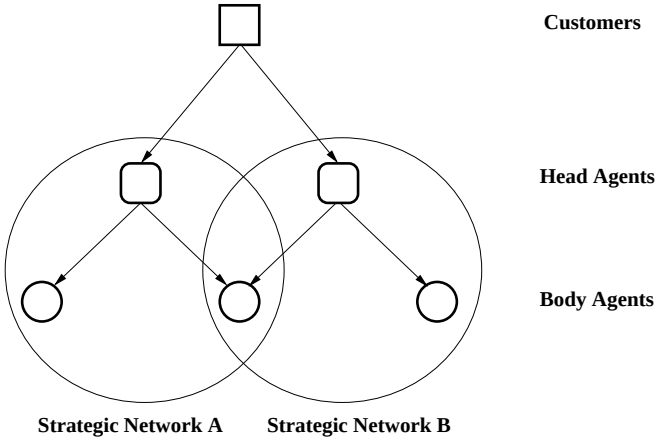

Figure 4.4: Task assignment in scenarios with strategic networks.

A	Agents processing an external order as cooperation bodies
D	obliged
I	refuse the order, sending the name of the head inside the refusal message

- Agents processing an order as cooperation heads must start an internal auction for each of its subtypes among their body agents.

A	agents processing an order as cooperation heads
D	obliged
I	start an internal auction for each subtype

4.3.5 Strategic Network

Just like cooperations, strategic networks (figure 4.4) consist of a head and body agents. Body agents again have to bounce incoming external orders. The difference between these forms is that heads in strategic networks know about their

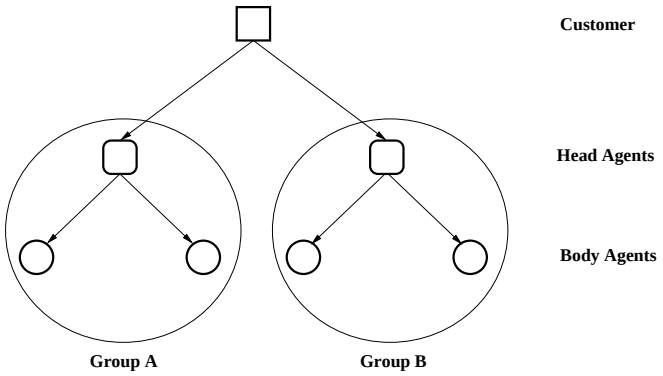

Figure 4.5: Task assignment in scenarios with groups.

body agents' schedules and resources, and can instruct them to do a task at a given time. Strategic networks are product-specific, so multiple memberships are allowed. Since multiple memberships can result in a head not being up-to-date about a body agent's resource allocation, the underlying strategic network authority protocol contains a confirmation step for the case that the body agent has processed an order outside the strategic network and his resource allocation has changed since he last informed the head agent about it. Body agents have to inform their heads about changes in resource allocation as soon as possible.

The profit distribution happens according to a fixed ratio, as in the cooperation. There is no gift giving among the members of a strategic network.

- Agents processing an external order as strategic network bodies must bounce the order.

A	Agents processing an external order as strategic network bodies
D	obliged
I	refuse the order, sending the name of the head inside the refusal message

- Body agents must accept orders from their heads, if possible.

A	all strategic network body agents
D	obliged
I	accept task ordered by a strategic network head agent
C	can allocate enough resources for task

- Body agents must keep their strategic network heads up to date about their available resources.

A	all strategic network body agents
D	obliged
I	keep all their strategic network heads up to date about their resources

4.3.6 Group

In contrast to strategic networks, groups (figure 4.5) are not product-specific. An agent who is a member of a group is not allowed to be a member of any other organization. Any incoming order has to be processed as group. Body agents have to bounce incoming orders. Head agents may order body agents to do a specific task. This inclusion of all economic activity in the group results in the head agent always being up-to-date about its body agents' resource allocations. The underlying group authority protocol does therefore not require a confirmation phase and is shorter than the strategic network authority protocol.

The head agent retains all the profit for orders completed by the group. Each round, it pays a fixed amount of money to each body agent.

- Members of a group may not be members of any other organization.

A	members of a group
D	forbidden
I	be member of any other organization

- Agents processing an order as group bodies must bounce the order.

A	Agents processing an order as group bodies
D	obliged
I	refuse the order, sending the name of the head inside the refusal message

- Body agents must take orders from their heads.

A	all group body agents
D	obliged
I	accept task ordered by a group head agent

4.3.7 Corporation

This last organizational form, the corporation (figure 4.6), is the result of the head of the group assimilating the resources of its body agents. Afterwards, the body agents no longer exist, the head agent acts like a normal single agent. There are no further rules for this organizational form apart from the rules for single agents.

4.4 Self-Organization

In most scenarios, it cannot be expected that the providers or their designers know in advance what the best system organization will be to fulfill the system requirements. One possible cause for this uncertainty could be changing customer demands that are determined from outside parties and other circumstances not known in advance. Another cause might be the complexity of the system, which precludes predicting system behavior and computing the optimal

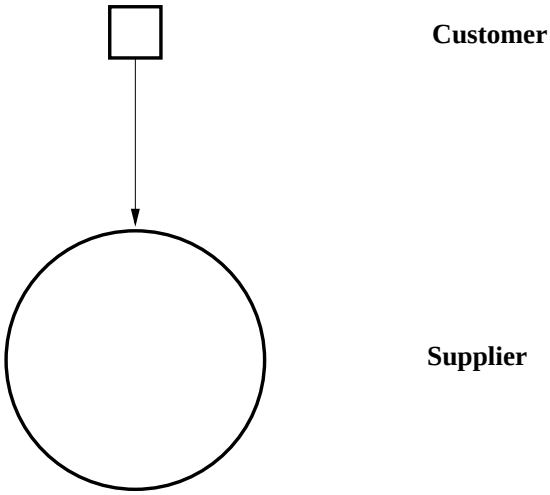

Figure 4.6: Task assignment in scenarios with a corporation.

structure. In such scenarios, one of the best options might be to let the providers organize themselves and decide during runtime what organizational structure is most profitable for them.

The adaptation of provider relationships to changing customer demand has two components: the forming of new organizations and the change of existing ones. The first section describes how agents decide when to form a new organization and with whom. The second section describes how agents evaluate their memberships in existing organizations and decide on whether to resolve the organization, keep it, or change it to a different form.

4.4.1 Creation of New Organizations

When an agent system has to do the same tasks for a longer period of time, it might be more efficient to adapt its structure to the structure of the tasks. If an agent finds that it repeatedly completes a task by working together with certain other agents, it should consider to facilitate this teamwork, as it can be expected to be asked to complete the same task again. The means of facilitating teamwork we investigate here is the formation of an organization, where the agents forming the organization commit to a certain long-term behavior in order to increase the efficiency of the group on certain tasks. This long-term commitment might be detrimental to the agent if the selection of partners for the organization was based on a rarely occurring constellation of orders in the system, so the agent should not commit carelessly.

How should it decide when to form an organization and with whom? One reasonable criterion for the selection of partners is the experience the agent had with them. If the agent has delegated many tasks to the other agent or received many task delegations from it, part of the preconditions of an efficient organi-

zation containing these agents are met. We formalize this experience by having each agent keep a record of the trade history with the other agents. The agent has an agent model for each other agent containing the order volume of tasks it delegated to it and of tasks it got delegated from the other. The decision process of whether to build a new organization and with whom piggybacks on the normal auction process: whenever a customer has found a group of agents that together can complete its task, this group checks whether they should form an organization. The motivation for this is that the completion of the task shows that this group of agents can successfully meet the current customer demand, so the members of the group are a reasonable choice of potential organization partners.

In order to avoid the complexity of recursive organizational structures in our system, we only let new organizations form if the agents participating in the completion of the task are doing so as single agents, not acting in the agenda of an existing organization. They can, however, be part of an existing organization that was built for a different product. The decision of whether to build a new organization is based on an algorithm working on a graph. The nodes of the graph are the agents that take part in the completion of the customer's order. Each agent in this graph checks its total trade volume with the other agents and selects those whose trade volume exceeds a given threshold. The agents selected this way are connected to the agent with an edge in the graph. Once all agents have computed their connections, the graph is checked for connectedness.

If the graph is connected, the agents agree to form a new organization. A node does not need to be connected to every other node, but there should be a path from it to every other node in the group. The reasoning behind this is that an organization should be "glued together" by the experience of profitable relationships, but it is not necessary that every member interacts directly with all others. Before building this new organization, each agent has to check whether this organization would conflict with its existing organizations or its plans of changing its membership in existing organizations. Since agents can complete more than one order in each round, the possibility that new potential organizations conflict with each other has to be checked as well.

4.4.2 Change of Existing Organizations

If agents decide to build an organization as described above, they form a virtual enterprise, the organizational form with the least commitment necessary. This organizational form can be upgraded to a form with more commitment if the collaboration in the organization has shown to be profitable for the agents, or it can be resolved if it turns out that the organization is inefficient and the agents are better off working as single agents again.

The progression of organizational forms is strictly along the spectrum of autonomy, that is, new organizations start out as virtual enterprises, can be up-

graded to cooperations, which can be in turn upgraded to strategic networks. Strategic networks can be upgraded to groups, and those to corporations. All organizational forms except the corporation can be resolved. Agents who are part of a group that decides to form a corporation merge into a single agent and stay so for the rest of the simulation.

Organizations that turn out to be inefficient do not downgrade one step down the spectrum, rather they resolve completely and all agents start out again as single agents, except if they were members of other organizations as well. The reasoning behind this is that the organization is likely to have become inefficient because the order situation has changed, so that a complete regrouping might be more reasonable than just lowering the commitments of existing organizations.

The decision of progressing along the spectrum rather than, for example, allowing a strategic network to build from scratch might not reflect the development of the respective organizational forms in human societies, but it seems a plausible implementation of careful behavior, given that in our scenario the agents have a high uncertainty with regard to the order situation.

The decision of whether to keep, upgrade, or resolve an existing organization is made by all agents together: each agent votes for one of the three options. Agents choose how to vote based on the average volume per round of orders they have processed via this organization. Orders they have processed as single agents or via other organizations are not considered for their vote. If this average volume exceeds an agent-specific threshold, the agent votes for upgrade. If it is below another agent-specific threshold, the agent votes for resolve.

If all agents vote for upgrade, the organization tries to upgrade. If at least one agent votes for resolve, it is resolved. In all other cases, the organization tries to keep the current form. By "tries" we mean that all organization members need to check whether the proposed change would conflict with their membership restrictions. For example, an agent who is simultaneously in a strategic network and a cooperation cannot upgrade the strategic network to a group without violating the restriction that members of a group may not be members of other organizations at the same time.

To reflect that organizations are long-term commitments and to avoid exceeding reorganization in the system, organizations that have formed or changed stay in their current form for a minimum number of rounds, after which they are reevaluated each turn.

We briefly considered allowing the change of organizations by letting existing organizations grow or shrink incrementally by assimilating or expelling single members, but the resulting complexity arising from membership restrictions was considered too great for the current implementation.

4.4.3 Membership Conflict Resolution Algorithm

Since agents can be members of several organizations at the same time, building new organizations or changing existing ones can violate the organizations' membership restrictions. For example, a strategic network cannot upgrade to a group if one of its agents is a member of another organization. We need to set priorities that decide how to resolve conflicts between different organization change proposals. These are the priorities we use in our system:

- If building a new organization would conflict with an existing organization, do not build the new organization.
- If upgrading a strategic network to a group would conflict with an existing organization, do not upgrade to group.
- If upgrading a strategic network to a group would conflict with building a new organization, do not upgrade to group.

In summary, keeping existing organizations has highest priority, followed by building new organizations, and upgrading to a group has lowest priority. Other upgrade conflicts cannot occur, because the product specific to an organization is the same before and after the upgrade, so that upgrades other than strategic network to group do not change membership restrictions. They can be treated as if the organization proposed to keep its form.

Algorithm 1 shows the pseudocode for the implementation of these priorities in our system.

4.5 Protocols

The parts of this section on the CNP, CNCP, and HCNCP are taken from our publication for the Multiagent Interoperability workshop at the German Conference on AI [Knabe et al., 2002].

The *Contract Net Protocol* (CNP) is a widely used protocol in DAI, as it proved to be a flexible and low communication interaction protocol for task assignment. The situation it is best suited for is that of a single task to be assigned among a number of individual agents. However, it has shortcomings if the setting for task assignment is more complicated. If there are several agents who concurrently start protocols to assign tasks, early commitment of bidder agents in the standard CNP leads to suboptimal outcomes, as assignments that are possible are not found. We propose the *Contract Net With Confirmation Protocol* (CNCP), an extension to the CNP that avoids the problems of early commitment. In scenarios where tasks are assigned in cascades, for example in holonic agents, this extension takes the form of the *Holonic Contract Net With Confirmation Protocol* (HCNCP). Examples of settings where the extensions improve on the

Algorithm 1 Membership conflict resolution algorithm.

```
1: procedure ResolveConflicts (orgs: existing organizations of this agent,
   orgsToBuild: new proposed organizations for this agent)
2:   agentInGroup? ← false
3:   for all OF in {VE, COOP, SN, GROUP} do
4:     for all org in orgs[OF] do
5:       if org.proposal != RESOLVE then
6:         lock org.product
7:         if OF = GROUP then
8:           agentInGroup? ← true
9:   for all newOrg in orgsToBuild do
10:    if agentInGroup? = true or newOrg.product is locked then
11:      newOrg.proposal ← RESOLVE
12:    else
13:      if newOrg.proposal = UPGRADE then
14:        lock newOrg.product
15:  for all org in orgs[SN] do
16:    if org.proposal = UPGRADE and more than one product is locked then
17:      org.proposal ← KEEP
```

standard protocol are given, as well as a discussion of the new protocols. We also introduce protocols for authority directions in strategic networks and groups, and for coordination of self-organization and voting.

4.5.1 Contract Net Protocol

Introduction

The assignment of tasks to agents and the (re-)allocation of tasks in a multiagent system (MAS) is one of the key features of automated negotiation systems [Weiß, 1999b]. The contract net protocol (CNP), originally proposed in [Smith, 1980], and other more general auction mechanisms can be widely applied to resource and task allocation problems. The contract net has been applied e.g. to online dispatching in the transportation domain [Bürkert et al., 2000, Fischer et al., 1995], meeting scheduling [Garrido and Sycara, 1996, Sen and Durfee, 1994] and flexible manufacturing [Shen et al., 1998, Camarinha-Matos and Afsarmanesh, 1998, Parunak, 1995]. Our discussion is based on the FIPA interpretation of the contract net [FIPA, 2001], which is a minor modification of the original protocol, in that it adds a rejection speech act and a mechanism to inform the initiator about the outcome of the task. Currently, this interpretation is the standard for a whole range of prominent agent platform implementations [FIPA-OS, 2002, JADE, 2002, ZEUS, 2002]. Figure 4.7 shows an UML interaction diagram for this

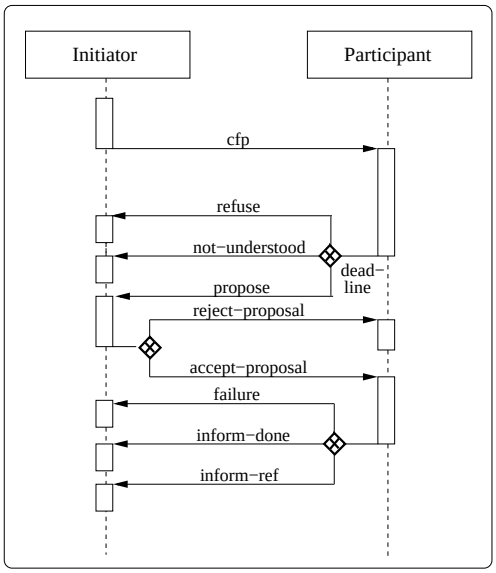

Figure 4.7: The FIPA Contract Net Protocol.

protocol. In order to comply with the FIPA standards, we call the agent with the task *initiator*, agents that compete for acquiring the task *participants*. In general, the procedure requires the initiator to send a "call for proposals" including a task description to all participants. They can specify their required costs for this task in a proposal (or refuse to do the task at all). The initiator then accepts one of these proposals, and rejects all others. The agent who got his bid accepted is then required to inform the initiator about the result of the task (or its failure).

This protocol was designed for distributing one task among a number of agents. However, if we assume a large number of initiators and bounded resources for each of the participants (as is common in today's multiagent systems), new problems arise. Although the execution of this protocol is very efficient, it is a hard problem for each agent to decide when to allocate the resources for which task. Imagine that among the agents in a large-size multiagent system there are n agents with tasks (initiators) and m providers of services (participants). While a participant is in negotiation with a large number of initiators, it may still receive more call for proposals without having received any reject messages as the initiators are still busy evaluating the proposals.

Up to now it remains an unanswered question which policy the agent should use for resource allocation, i.e. in what manner it should reserve resources for tasks it made a bid for. If the agent allocates too many resources too early, it may not get its bid accepted and therefore resources will not be available for other tasks. If it allocates too late, it may have committed to more tasks than it has resources. The term “Eager Bidder Problem” has been coined for this dilemma. Several approaches have been proposed: leveled commitments

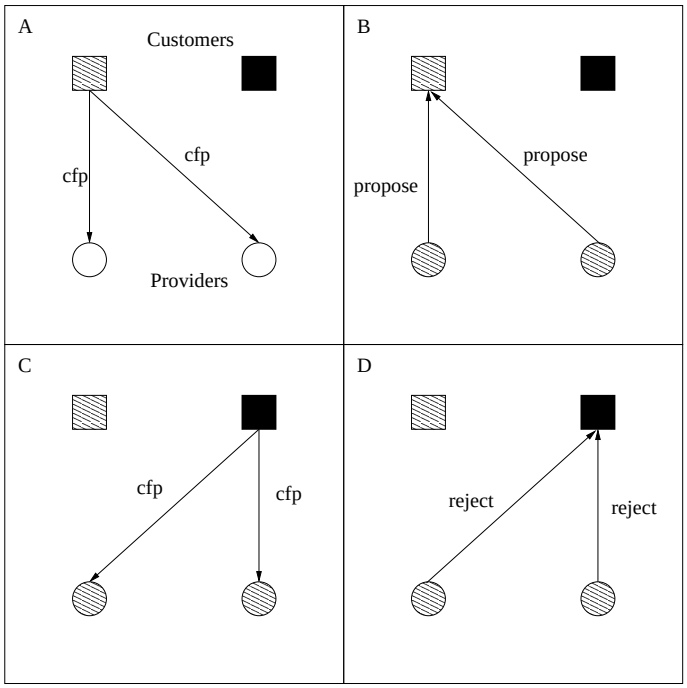

Figure 4.8: Example of CNP sub-optimality.

([Sandholm and Lesser, 1996]; for an extension see [Excelente-Toledo et al., 2001]), and statistical methods (as they are being used e.g. in flight booking systems) [Schillo et al., 2002]. The latter depend on data gathered over a long period of time and involve the risk of over-booking (as is the common experience with frequent flyers) while the former requires more complex communication, resulting in higher computational costs for both participant and initiator.

Shortcoming of the Contract Net Protocol

Let us consider the case where the agent allocates resources at the time of sending the bid. We call this solution the ad hoc solution, or the conservative approach. This solution makes sure that only correct assignments of tasks to agents are created, i.e. that every agent only commits to the tasks it can perform. However, if several participants send their proposal to the same initiators, which is not unlikely, the result is that only some of them get a task assigned, while others remain idle. Therefore, this procedure is not complete in that it will not compute assignments that could be found with better approaches.

Figure 4.8 gives an example of a simple situation with two customers and two providers where the CNP can lead to a suboptimal outcome. In phase A, the first customer sends a call for proposals (cfp) to each of the provider agents. The provider agents reply with their proposals in phase B, allocating the resources

required for the task. If, as shown in phase C, the second customer sends its cfps before the first could finish its protocols, both providers will still have their resources allocated for the first task, and therefore have to send reject messages to the second customer in phase D. Even though the two providers could in principle handle the two tasks, the system did not find this optimal solution.

The likelihood of failing to find a possible solution is even higher in scenarios with more agents. Consider using the conservative approach in a setting with 100 initiators, each having one task to assign and 100 participants, each capable of performing one task. Further consider that the deadlines are set in a way that the participants cannot reply to the calls sequentially (otherwise the multiagent approach would hardly apply). If in this case every participant uses a conservative approach to the problem and just sends one bid, the chance of getting a bid accepted assuming lottery on the side of the initiator is ca. 0.64 (the computation of this probability is out of scope here, but from the problem chosen, it is in any case clear that the probability is below 1). If other agents make more than one bid, the probability is even lower. So in more than one third of all cases, the available resources of the participant will be idle due to the conservative strategy. Correspondingly, the same number of initiators will be left with unassigned tasks, as they did not get any bids for their tasks, although the resources are in the system.

4.5.2 Contract Net with Confirmation Protocol

Solution to the Shortcoming of the CNP

Our approach is based on redesigning the protocol to postpone the time of commitment as far as possible. The major inefficiency in the CNP is that in every execution of the protocol all participants need to commit themselves to do the job, although only one of them will actually get the task awarded. We now present the contract net with confirmation protocol (CNCP), which precisely addresses this issue and improves the CNP procedure by drastically reducing the number of commitments made.

The CNCP (Figure 4.9) is very similar to the CNP. It starts with a call for proposals and gathers the responses from the participants, until the initiator received messages from all participants or the deadline has passed. As in the contract net protocol, this deadline safeguards that singular message dropouts do not prevent the whole protocol from terminating. In the original contract net, the participant makes its commitment in the bidding stage. In the CNCP this is not the case: the commitment is only made when the initiator requests that the participant should take over the task. For this purpose the initiator arranges all bids in a sorted list and sends requests to participants starting with the best bid to find out if they can actually do the job. The next participant is sent a request message if the previous participant has sent a refuse or a deadline has passed.

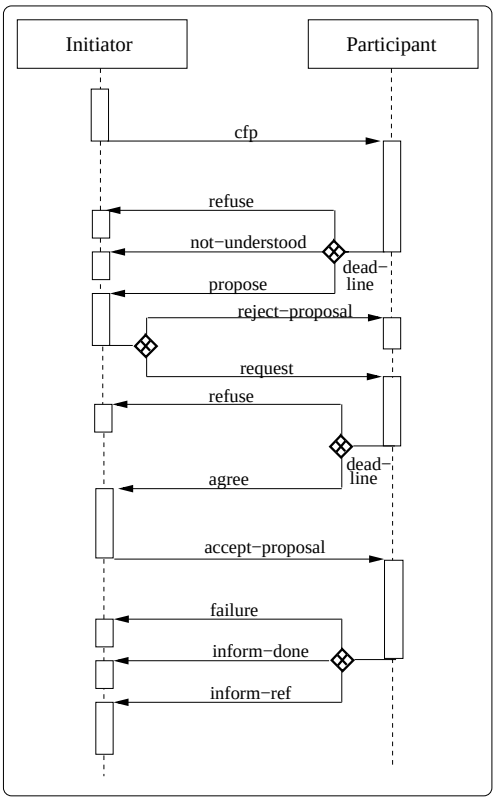

Figure 4.9: The Contract Net with Confirmation Protocol.

This iteration stops if one participant sends an agree message. All other agents are sent a reject-proposal message (except those who have already received the request and sent the refuse). The participant only needs to commit at the time of sending the agree message and can make bids to other cfps in the meantime. In order to trigger task execution and to correspond to the CNP it is required that the agent sends an accept-proposal while the participant will reply (as it does in the CNP) with failure, inform-done, or inform-ref.

Analysis

As well as the original contract-net protocol, the proposed procedure needs $O(n)$ messages, where n denotes the number of participants. In the best case, the CNCP requires only two more messages (the request for confirmation and the reply to it) while still solving the resource allocation problem of the initiator. In the worst case, the initiator needs to contact all participants to find out that no one can do the task. Although this results in a plus of $2n$ messages for the CNCP, its great advantage is that it only requires one agent to make a single commitment. This is achieved by using the confirmation stage in the protocol, to postpone the

commitment and allow the participants to reply to all incoming call for proposals without need to already allocate the resources at this early stage of interaction or to risk penalties for multiply allocating resources. A minor disadvantage of this approach is that the initiator possibly needs some overhead to repeatedly find the next best bid, while the CNP only requires it once to find the maximum. However, with careful implementation this additional computational effort is by several orders of magnitude lower than the effort spent for sending the messages, and is in the general use of MAS a negligible additional cost.

In order to guarantee termination even in the case of faulty participants the second deadline of the protocol is necessary. It makes sure that the next best participant can be sent a request message and has a chance to receive the task.

Shortcoming of the CNCP

Both the CNP and the CNCP work in conventional as well as in holonic multiagent systems (HMAS). HMAS require due to their recursive structure the recursion of negotiation protocols, and both protocols can be used in cascades, i.e. each participant which is a holon head can initiate another instance of the same protocol to subcontract the task to other agents (generally agents in the same holon). In the case of the CNP this leads to rapidly increasing allocations of resources as all participants must allocate their resources (see the discussion above). The CNCP avoids this inefficiency. However, in some cases a new inefficiency arises when applying a cascade of CNCPs, namely when some of the agents in the lower part of the cascade refuse to do the job after having made an initial bid.

Figure 4.10 shows a scenario of two customers and a holon consisting of two body agents and a head agent who is in charge of communicating to the outside. The cost of the first body agent for completing the task of either customer is 5, that of the second 6, so the second agent is less efficient than the first. Phase A shows what happens after both customers have sent their cfps to the holon head. The head reacts by starting another CNCP among the body agents its holon is composed of. It decides on the basis of its body agents' best bid how to reply to the initial cfp. The head chooses the cheapest agent in both cases and sends a proposal of 5 to both customers. After receiving the first request in phase B, the head forwards this request to its cheapest body agent, who in turn allocates the resources for the task, which are now no longer available. The request from the second customer in phase C has therefore to be rejected (phase D), as the holon can no longer complete the task for the proposed price of 5. The CNCP fails to find the solution of assigning the first task to the agent with cost 5 and the second to the agent with cost 6, which would be the optimal outcome in this scenario.

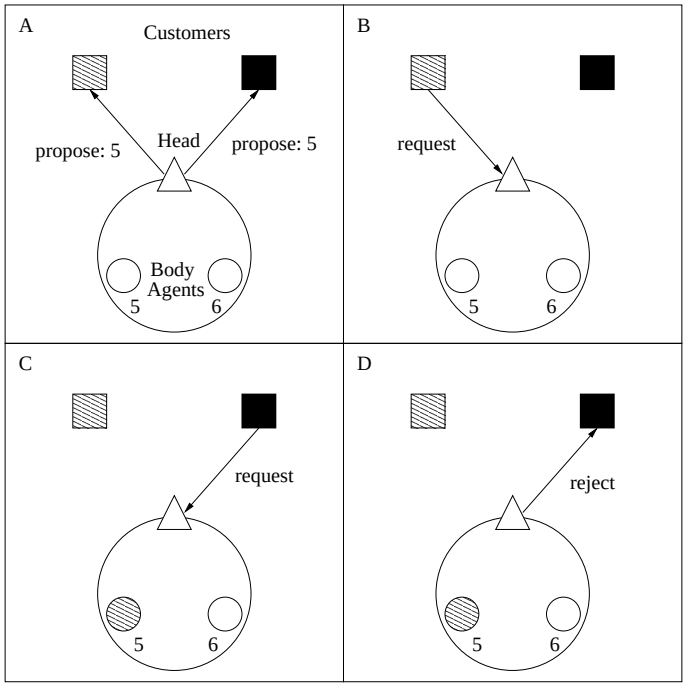

Figure 4.10: Example of CNCP sub-optimality in cascading applications.

4.5.3 Holonic Contract Net with Confirmation Protocol

Solution to the Shortcoming of the CNP

To avoid this problem in scenarios with holonic agents, we use a version of the CNCP that includes the possibility of a second proposal after the request has arrived. If the head finds that its cheapest body agent can no longer do the job, it can send a request to the second best bidder. If the second best bidder agrees, the holon head can send a second proposal to the initiator (which is equal or higher than the first), who can compare it to the bids it received from the other participants. Since the CNCP does not allow such a second proposal, the holon head would have to refuse the job even if its second best bidder could do it for a better price than anyone outside the holon. The modification to the CNCP for systems with holonic agents therefore consists of adding a second proposal as possible reply to a request. Figure 4.11 shows the resulting Holonic Contract Net with Confirmation Protocol (HCNCP).

The participant allocates the resources for the task when either agreeing to the request or making a second proposal. The second proposal requires a commitment to ensure the termination of the protocol. A noteworthy difference from the CNCP is that the initiator can send a reject even after the participant has committed. To see why this modification is necessary, recall the scenario mentioned above, where the holon makes a second proposal. It does so only after

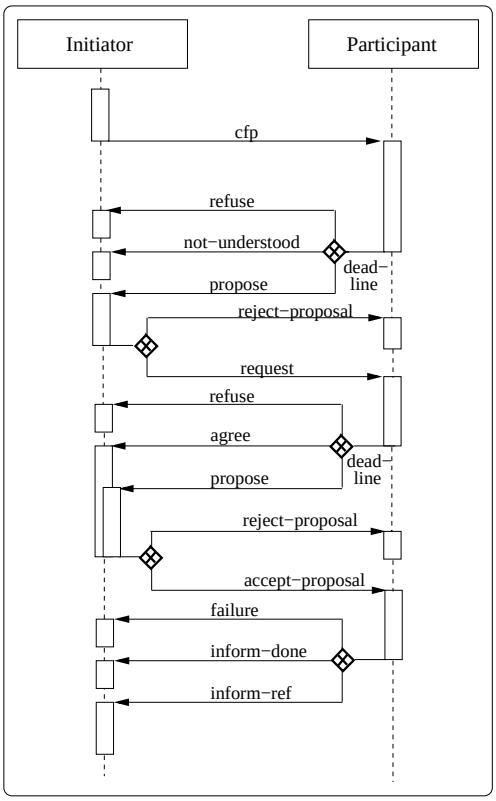

Figure 4.11: The Holonic Contract Net with Confirmation Protocol.

its second best bidder has committed. However, it is possible that this second proposal is rejected because it is no longer the best bid. In this case, the holon has to forward the reject to its committed subunit. In summary, the HCNCP is a recursively applicable protocol that reduces the number of unnecessary commitments by introducing a confirmation stage and that increases the flexibility of holons by allowing a second proposal to reach better solutions than the cascading CNCP.

Analysis

Assuming that all agents in a cascading HCNCP are nodes in a tree where the problem solving body agents are represented by the leaf nodes, using the second proposal the worst case occurs if in any holon with leaf node agents, the agent with smallest bid refuses to do the task and a leaf node agent in this holon commits. In this case the number of commitments increases to the number of parents of leaf nodes, but the protocol still reaches the same (optimal) solution as the CNCP for the non-holonic case would.

The HCNCP requires the same number of messages as the CNCP in systems

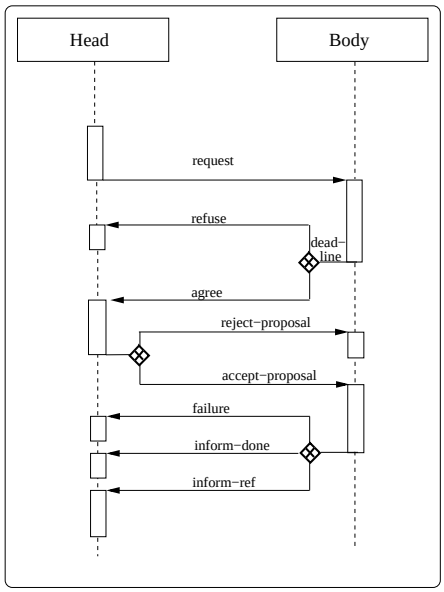

Figure 4.12: Authority Protocol with Confirmation.

without cascading protocol applications. In cases where the second proposal mechanism is used (i.e., in cascading systems where the CNCP would fail to find an assignment and send a refuse), each accepted second proposal requires two more messages and each rejected second proposal one more message.

4.5.4 Authority Protocol with Confirmation

The Authority Protocol with Confirmation (Figure 4.12) is used in strategic networks by the head to issue orders to its body agents. It differs from the HCNCP by the absence of a call for proposal stage and of an option for a second proposal. The head has access to its body agents' schedules and resource managers, so he knows which agent can do the job for the lowest price without having to ask them for proposals. He cannot, however, omit the confirmation stage: as agents are allowed to be members of several strategic networks, it is possible that a body agent has received a direction from another head since he last informed the first head about his schedule. He might no longer be able to allocate enough resources to fulfill both directions. In this case, the body agent has to reject the second direction.

4.5.5 Authority Protocol without Confirmation

Groups do not allow body agents to be members of several companies. As a body agent can have no more than one group head, he can not receive conflicting directions from different heads. The protocol used for directions in this organizational

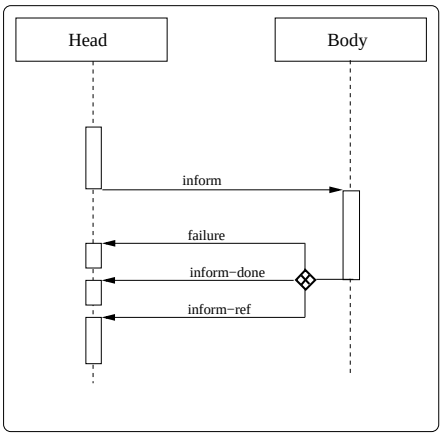

Figure 4.13: Authority Protocol without Confirmation.

form, the Authority Protocol without Confirmation (Figure 4.13), does therefore not require a confirmation stage. It consists of an inform performative from the head to the body agent, and a failure/inform reply.

4.5.6 Self-organization Protocol for Creating New Organizations

The communication necessary for the decision whether to build a new organization is not done via a separate protocol but rather piggybacks on the HCNCP during normal auctions. Whenever a customer has found a group of agents that can complete his order, this group decides whether to build a new organization. The information necessary for this decision is sent together with the normal content of the HCNCP messages.

When the customer sends a *request* message, it inserts an empty list into the content. A provider agent receiving a request message inserts itself into this list and forwards this new list in the content of the request message to its subcontractors, if it has any. In this way, the last subcontractor in this "order chain" receives a list of all providers in the chain. For each provider in this chain, it decides whether it would like the provider to be its partner in a new organization by comparing the average trade volume between the two agents to a threshold value. It only takes into account the trade volume of those elementary types that are part of the original customer order, because other types are irrelevant to the current demand.

The agent returns the list of providers that exceed this threshold in its agree/2nd proposal message. The receiver of this message forwards this list up the chain, together with its own list of "favorite" partners. Once the message propagation has reached the customer at the beginning of the chain, the customer checks whether the chain agents form a connected graph: the agents are

the nodes of the graph, and two nodes are connected if one of the agents is a member of the favorite partner list of the other. If the resulting graph is connected, the customer sends the list of chain agents in the accept proposal. Otherwise, this list is empty. Agents receiving an accept proposal with a non-empty chain members list know that they are supposed to form a new organization with these agents. They forward the list along the normal HCNCP accept proposal down the chain and note that they have to check at the beginning of the next round whether they really can form this new organization.

4.5.7 Self-organization Protocol for Existing Organizations

At the start of each round, before the customers emit their orders and the normal auctions begin, the provider agents compute changes to their organizational structure and communicate any changes to each other. For each existing organization it is a member of, a provider agent decides whether it wants to keep its current form, upgrade it to the next form, or resolve it, based on the average order volume per round it has processed via this organization. It needs to coordinate its decision with the other members and also take into account the membership restrictions for organizations.

The process of coordination is complex, because changes of an organization are restricted by changes in overlapping organizations. A proposed upgrade of a strategic network to a group fails if one of the agents is a member of another organization, but if that second organization has to resolve because one of its members finds it inprofitable, the upgrade of the strategic network is possible. We have decided to structure the coordination process in phases and for each organization let one of its members coordinate the communication.

The coordinator is the head in the organizational forms which have a head, and the first agent of the member list (which each member has an equal copy of) in the case of the virtual enterprise, which has no single head. New potential organizations that have not yet been built but are the result of the process described in the previous section participate in the coordination as well, their coordinator is chosen also by taking the first agent in the member list.

The coordination process has four phases. An agent keeps a separate phase for each organization and new-to-build organization it is a member of. A phase consists in either sending change proposals or receiving change proposals. After sending or receiving proposals, the agent marks this organization as ready to enter the next phase. Only if all of its existing and new-to-build organizations are marked as ready for the next phase does the agent initiate the next phase by starting the membership conflict resolution algorithm described in section 4.5.3. This algorithm checks for possible conflicts in the proposals and resolves them by changing some proposals. After that, the agent starts the new phase in each organization by sending the new proposals to the coordinator or its partners.

Figure 4.14 shows the protocols implementing the four phases. Proposals for

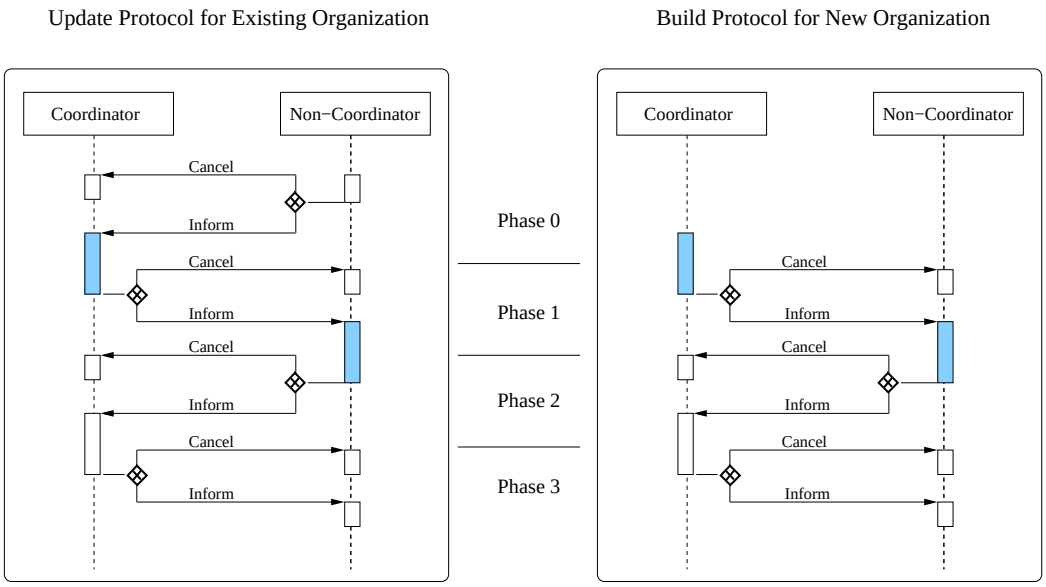

Figure 4.14: Self-organization protocols.

upgrade and *keep* are sent in messages with the inform performative, proposals for resolve with the cancel performative. The performatives needed to be different because resolve proposals always have the organization's resolution as consequence and therefore the protocol can terminate at this point, as the organization is no longer needed in the coordination process. The grey boxes indicate that the agent uses the membership conflict resolution algorithm. The left figure shows the protocol for updating existing organizations, the right figure for new, potential organizations.

In phase 0, the non-coordinators of an existing organization inform the coordinator of this organization about their proposal. After the coordinator received proposals from all of the organization's non-coordinators, it adds its own proposal to the set of collected proposals and computes the final proposal. If all agents proposed upgrade, the final proposal will be upgrade. If at least one agent proposed resolve, the final proposal will be resolve. After that computation, the coordinator marks this organization as ready for the next phase. When an agent has marked all of its organizations as ready for the next phase, it starts the membership conflict resolution algorithm. This may result in the "downgrade" of some proposals, for example, a strategic network that proposed to upgrade now only proposes to keep its form.

The coordinators send the resulting proposals to the non-coordinators in phase 1. Since the proposals have not yet undergone a conflict resolution on the side of the non-coordinators, all agents initiate another run of the resolution algorithm when starting phase 2, when all their organizations have received the proposals from the coordinators. This can result in further downgrade of some of the

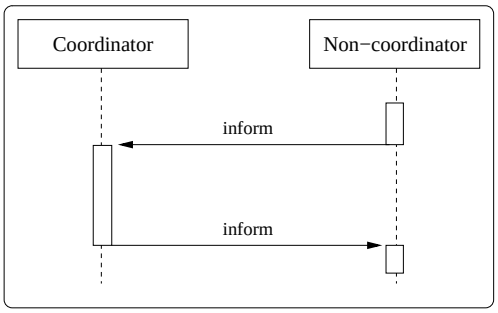

Figure 4.15: Voting protocol.

proposals. Finally, the resulting proposals are sent back to the coordinators, who compute the actual change for the organization in the way described above (resolve if at least one proposal is resolve, upgrade if all proposals are upgrade, otherwise keep form), and then inform the non-coordinators about the actual change in phase 3.

The protocol for new potential organizations (the right one in figure 4.14) is very similar to that for existing organizations. It differs only in that the non-coordinators do not send their first proposals in phase 0. The reason for this is that new organizations do not have a trade history to base a decision on, so the coordinator does not need to be informed by the non-coordinators about their proposal–it assumes the proposal to be *upgrade*. This proposal in the case of new-to-build organizations means to actually build the organization. The option *keep* is not used in the build protocol, as the potential organization does not exist yet.

4.5.8 Voting Protocol

Virtual enterprises do not have a designated head agent, but cooperations do. Whenever a virtual enterprise upgrades to a cooperation, the member agents need to decide which one of them will become the cooperation's head agent. They do this by voting: each agent votes for another agent, the agent with the majority of the votes becomes the head. The voting criterion is the trade history between agents. An agent votes for the virtual enterprise partner it has traded with most.

The voting process is coordinated by the first agent in the member list. After the self-organization phase has resulted in the upgrade of a virtual enterprise to a cooperation, the non-coordinators of that virtual enterprise send their vote to the coordinator. The coordinator then inserts its own vote into the set of received votes and computes the winner of the election. This result is then sent to all non-coordinators. Figure 4.15 shows the protocol used for this communication. The performative used for sending the vote and for sending the result is *inform*.

Chapter 5

Implementation

In order to test and evaluate the proposed application of the selected organizational forms to a multiagent system implementing a market for task assignments, we have developed two implementations for a simulation testbed. The first implementation was written in Java and uses the FIPA-OS platform for agent communication. The first section outlines the motivation for this implementation. In the final phase of the thesis, when the actual experiments were run, it turned out that this implementation does not meet the computational requirements for reasonably sized scenarios. Therefore we had to create a second implementation in C++. The second section goes into more detail about this implementation. The third section gives an overview of the scheduling algorithm we used: earliest-deadline-first. We need a scheduling algorithm since the tasks can have deadlines extending over several rounds. The fourth section deals with the communication that is not handled by the protocols specified in chapter Four.

5.1 FIPA-OS implementation

We want the agents in our electronic market to comply with the FIPA standard, in order to ensure interoperability with existing and future multiagent system projects and in order to draw on current work in agent communication. We had the option of writing the necessary implementations ourselves. This would have given us more control over the implementation details and a software package hand-tailored to our needs, but it would also have required more time. Fortunately, there are a number of different implementations of the FIPA standard available on the Internet. We chose to use FIPA-OS, a freely available software library that has the advantage of being used in other projects and diploma theses at the multiagent systems group of the German Research Center for Artificial Intelligence (DFKI) where this thesis was written. We could therefore profit from existing experience with this library and retain the option of future merging of code from the different projects. Example projects at the DFKI that use FIPA-

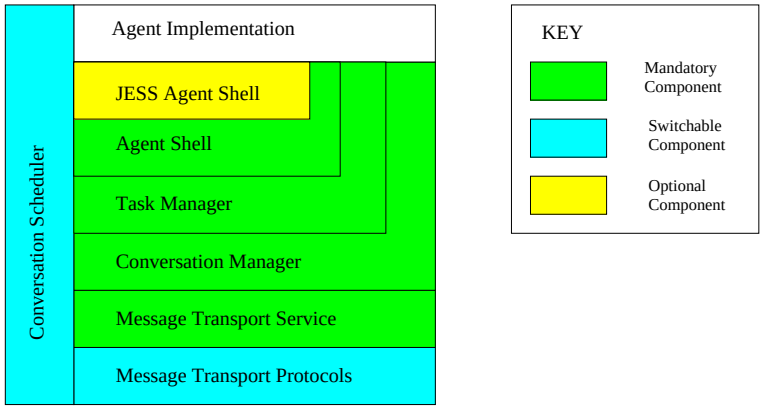

Figure 5.1: High-level architecture of FIPA-OS [Guide, 2001].

OS are CASA and SAID. The FIPA-OS implementation uses JESS as a reasoning engine. JESS' advantages were that it is free for academic purposes, has a good interface to Java, and is well integrated into the FIPA-OS framework.

5.1.1 FIPA-OS Overview

FIPA-OS is a toolkit for developing FIPA-compliant agents. It is an open source (suggested by the "OS" in the name) Java based multiagent system framework developed by Nortel Networks with the intent of providing a platform whose architecture emphasized ease of extension and modularity [Poslad et al., 2000]. Figure 5.1 shows the high-level architecture of FIPA-OS. The color of each component shows whether this component is mandatory and has to be executed by all FIPA-OS agents, switchable, meaning that agents developers can choose among different implementations for this component, or optional.

We will only discuss some of the components shown in this figure: the *Message Transport Protocol*, the *Conversation Manager*, the *Task Manager*, the *Agent Shell* and the *JESS Agent Shell*.

Message Transport Protocol The message transport protocol can be implemented as for example RMI, which restricts the interoperability to agents written in Java, or IIOP, which enables communication between agents that are written in a language that supports CORBA, but is less efficient than RMI. We do not use CORBA and have therefore chosen RMI as our message transport protocol.

Conversation Manager The conversation manager provides the ability to track conversation state at the performative level, as well as mechanisms for grouping messages of the same conversation together. All conversations

in our system follow protocols, and the conversation manager ensures that the conversation follows the protocol specified for each conversation.

Task Manager FIPA-OS agent split their functionality into small, disjoint units known as *tasks* and implemented as separate objects. Task are event-based objects that can run concurrently and that can be associated with conversations, allowing the agent to conduct several different conversations at the same time. Tasks can spawn child-tasks, structuring the functionality of the agent in a hierarchy.

Agent Shell The agent shell is the level on which programmers implement their own agents. It provides the interface to the other components and the functionality of sending messages, retrieving the agent's properties, registration with platform agents, setting up tasks, and shutting down the agent.

JESS Agent Shell The JESS agent shell provides an interface to the JESS engine, which is discussed in one of the next sections.

5.1.2 Alternatives to FIPA-OS

There are a number of other multiagent system frameworks that comply with the FIPA standard and are available for free download: Zeus [ZEUS, 2002] and JADE [JADE, 2002]. A study by Fonseca et al. which compared the three frameworks showed that FIPA-OS has advantages in that it has a good conversation management by checking for protocol compliance, offering message retrieval and storage utilities, conversation tracking by identification number, and a messaging agent for manually driving and debugging conversations [Fonseca et al., 2001]. Its disadvantages include the lack of built-in support for customizing message routing and lack of built-in state machine support. We decided to use FIPA-OS because these disadvantages are only a minor inconvenience in our application, and there has been more experience with this framework in our working environment.

5.1.3 JESS

Our agents need a certain degree of reasoning ability to maintain a model of other agents and decide on its basis how to behave in the electronic market. We chose to implement the reasoning mechanism in form of an expert system, which is a rule-based approach to capturing an expert's knowledge in form of if-then rules. An expert system consists of a base of rules comprising the domain knowledge, facts that are parameters characterizing the current problem and an inference engine that makes rules fire if the facts matching their firing patterns are present in the knowledge base. Activated rules can assert new facts or modify old ones. Expert systems are often more efficient than other reasoning mechanisms because the inference engine uses the Rete Algorithm, which can exploit structural similarity

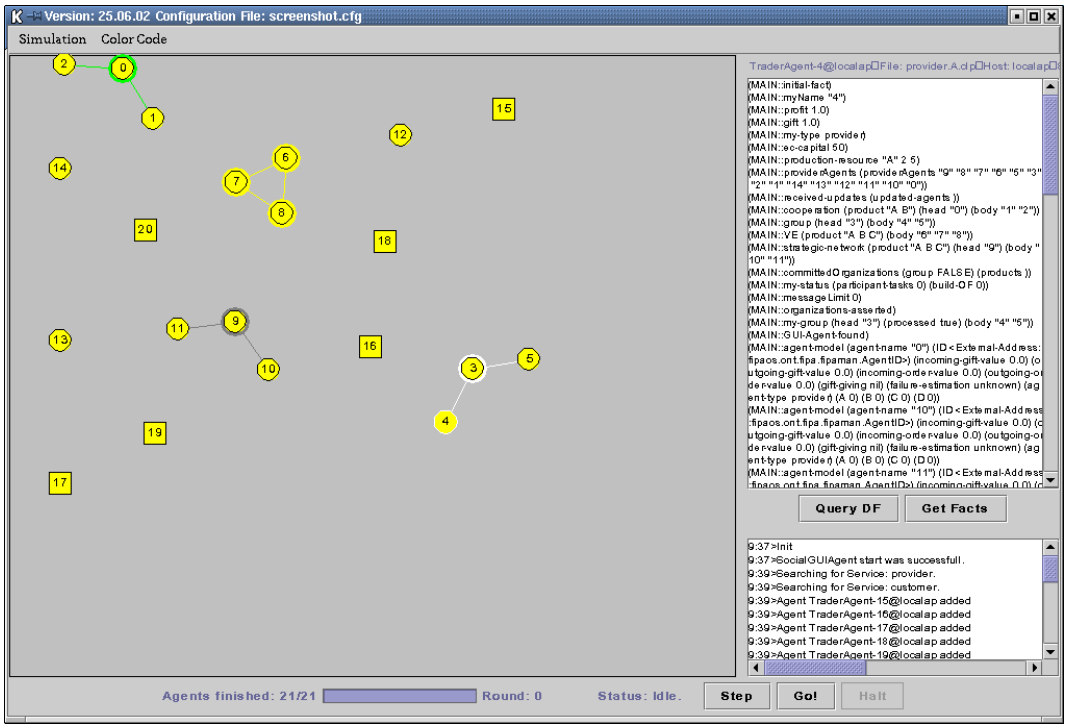

Figure 5.2: Testbed for organization in multiagent systems.

of rules to save computation. The software package JESS (Java Expert System Shell) provides such an expert system inference engine written in Java [JESS, 2002]. We used JESS because it is free for academic purposes, has a good interface to Java, and is well integrated into the FIPA-OS framework.

5.1.4 The User Interface

To evaluate the application of sociological concepts to task assignments in multiagent systems, we have developed the Testbed for Organization in multiagent Systems (TOM). The testbed is a round-based simulation of an electronic market where customer agents try to assign their tasks to provider agents, who can be organized in different groups and organizational forms, and who can change their organization. Figure 5.2 shows a screen-shot of the program.

The large graphic area displays the agents as squares and circles, with squares symbolizing customer agents and circles symbolizing provider agents. Agents that together form organizations are connected by lines, the type of organization determining the lines' color. Each organization has one or more heads that are marked by a ring around their symbols. The ring's color is the same as the link color for the organization. Once the simulation is running, agents that successfully assign tasks are connected by black lines to the agents they assigned the tasks to. The position of the agents in the graphics area is determined at ran-

dom at the beginning. After that, the positions change depending on the links between the agents. A link can be seen as a spring with a given rest length. If the actual length is larger than the rest length, the spring is extended and the spring will exert a force on the agents it connects, trying to draw them towards each other. If the actual length is smaller than the rest length, the force will try to push them apart. The rest length for organization links is a small constant value. The rest length for black links depends on the number of successful task assignments between the two agents: the more task assignments, the shorter the rest length. Black links between agents that did not trade with each other for the last few rounds are deleted.

Clicking on an agent lets the user inspect its JESS facts by displaying them in the right window. The user can control the simulation by letting it run stepwise with the program halting and waiting for further instructions after each round or in continuous mode, with the next round starting automatically at the end of each round. The setting is specified by configurations, which the user can create, load, save, or edit at the start of the program. The configuration determines the number of agents and parameters for each agent.

The current version supports only agents on the same platform, but future versions might be extended to enable the user to simulate a market where the interacting agents are located on different platforms and communicate over the Internet. We did not implement this advanced version for our experiments, because it turned out that the performance bottleneck in our application is the speed of message routing as opposed to local processing, so we would not have gained much if the agents were distributed on different systems.

The agents print all incoming and outgoing messages to the console, as well as some special information like their current capital and whether they failed to assign an order. The console log is then processed by Python scripts and the resulting data fed into Microsoft Excel for further analysis.

5.2 C++ Implementation

5.2.1 Reasons for a Second Implementation

Unfortunately, after the FIPA-OS testbed was implemented and used to run large scenarios, it turned out that the implementation had severe limitations. Only a very small fraction of the simulation runs completed without crashing, which we attribute to the instability of FIPA-OS in scenarios with thousands of auctions running at the same time on the same machine. Our specification of the market scenario requires a number of auctions exponential in the number of elementary types in the customer orders and in the number of provider agents. The discussions on the FIPA-OS mailing list suggest that this may have been the first time that a system using FIPA-OS has had to meet requirements of such

scale, which could be the reason we have not heard of similar difficulties with FIPA-OS before.

We estimated that these difficulties with FIPA-OS will not be solved in an acceptable time, so we decided to implement a multiagent system in C++ that uses FIPA conforming messages and that allows us to run experiments with scenarios containing a reasonable number of agents and completing in an acceptable time. The C++ implementation has been used for the experimental evaluation for this thesis. The FIPA-OS implementation might become of interest again in the future when the library reaches a more stable state or if the scenarios need fewer agents and less complex orders.

5.2.2 Differences Between the Implementations

The two implementations have in common the GUI and the architecture based on conversation tasks. A major difference is the concurrency: the FIPA-OS implementation uses concurrent threads for its tasks, while the C++ implementation is based on global *ticks*. Each conversation task in the C++ implementation has two sets that can contain messages: an inbox containing all messages sent to the task in the last tick, and an outbox containing the messages the task wants to send this tick. A tick consists of two phases: a *send phase* and a *run phase*. In the *send phase*, each task sends the messages in its outbox by removing the message from the outbox and putting it in the inbox of the message's receiver. At the end of the send phase, all outboxes are empty. After the send phase, agents start the *run phase*, in which each task processes the messages in its inbox. For example, a HCNCP participant task that finds a call for proposals in its inbox will react to the cfp by evaluating the order and either putting a refuse message or a proposal message in its outbox. The tick ends when the agents cannot process anymore without messages being sent. Then the next tick starts. Such a tick-based communication has been used before in experiments with multiagent systems by Turner and Jennings [Turner and Jennings, 2000].

5.2.3 Hardware Used

The experiments were run on a cluster of 39 dual Pentium III 800 MHz computers with 256 MB RAM each.

5.3 Scheduling Algorithm

5.3.1 Scheduling in our Implementation

The provider agents have limited resources: each turn, they can produce no more than a limited amount of a certain type. The customers, in turn, have limited

time: they want their orders executed in no more than a certain number of turns. The providers' resource limit is called their capacity, the customers' time limit is called their deadline. When deciding whether he can fulfill an incoming order, a provider has to check if he has enough capacity to execute the new order together with the orders he has already accepted such that each order is finished before its deadline expires. The problem of finding a plan which results in the timely completion of all orders is called a *scheduling problem*.

5.3.2 Scheduling Overview

Scheduling problems are widespread in the areas of business and computing. Optimal algorithms and heuristics have been suggested for different classes of such problems. In general, scheduling can be described as the problem of determining whether a given list of tasks can be completed in such a manner as to satisfy a deadline for each task. The task can be periodic or aperiodic, and they can be known in advance or may change at runtime. Systems that know their tasks in advance and build a schedule only once are called *static schedulers*. When new tasks can arise at runtime or old tasks change, the schedule will have to be updated every time a critical change arises. Such systems are called *dynamic schedulers*. Scheduling problems are also differentiated in those that allow *overload* (missing a deadline is not fatal) and those that do not, as well as systems that can process only a single task at a time, and those that can run several tasks in parallel.

5.3.3 Scheduling Requirements in our Application

In our system, each task has to be scheduled as a single instance, hence the tasks are *aperiodic*. The provider agents do not know the tasks in advance; they have to decide on accepting new incoming tasks and executing accepted old ones in each round, hence the scheduling must be *dynamic*. In our system, we consider deadlines to be strong requirements: we do not allow *overload* of the processing units. Finally, each agent can process only one task of each elementary type at a time, so the providers act as *uniprocessors*. We want to emphasize that the scheduling problem in our application to which we want to apply a scheduling algorithm is restricted to finding a *local* schedule for a single agent who has a list of orders to process. The *global* scheduling problem, the assignment of tasks to a number of agents, is known to be NP-complete and is handled by the auction mechanism, not with a specific scheduling algorithm.

5.3.4 Choice of Scheduling Algorithm

There is an algorithm that is optimal for such systems which are aperiodic, dynamic uniprocessors with no overload: the earliest-deadline-first algorithm

(EDF). This algorithm computes a schedule by always executing the task with the earliest uncompleted deadline. EDF has been shown to be optimal in that if a schedule exists that will meet all deadlines, EDF will find it [Liu and Layland, 1973]. A further advantage is that it will produce a schedule with the shortest completion time. It is a very popular algorithm and is the basis of a large part of the dynamic priority scheduling research. Its only drawback is in systems that allow overload. When not all deadlines for the accepted tasks can be met, EDF degrades ungracefully. But since our systems does not allow overloads, this drawback is not an issue here.

5.3.5 Speed of Selected Algorithm

To compute the algorithm's performance, we need to describe the implementation in more detail. Each provider maintains a list of accepted and unfinished tasks in order of ascending deadline. To check whether a new task can be scheduled, the agent inserts the new task into the list and, for each task in the new list, checks if its *laxity* is greater or equal to the time required for completing all tasks preceding it in the list. A task's laxity is defined as the time from now to the task's deadline minus the time required to complete it. The laxity is a measure of free time available for other tasks. When checking or accepting a task, the scheduling algorithm's running time is $O(n)$, where n is the number of orders already scheduled. Finding the task to be processed next is $O(1)$, as this task is the first element of the sorted list.

5.4 Communication not Handled by the Protocols

Apart from messages sent according to one of the protocols described in the fourth chapter, agents communicate for other purposes, like telling each other when to start a new round or that they are no longer participating in new auctions. This section lists the communication that happens outside of the protocols, except for trivial messages that are sent for updating the GUI. The only communication described here that contributes to the performance measure *number of messages* is the special communication for strategic networks and groups (Section 5.4.2), all other messages are not counted.

5.4.1 Simulation Control Communication

The simulation is controlled by a special agent who does not take part in the auctions. This *control agent* is responsible for starting and shutting down other agents, initiating new rounds, and keeping a count of auctions that have not yet finished in the current round. Each round is divided into two phases: the initiation phase and the auction phase. When starting a new round, the control

agent sends a message to all other agents telling them to initiate for the new round. Customer agents do nothing in the initiation phase, but provider agents initiate by starting self-organization protocols, depending on the organizations they are a part of and whether they agreed in the previous round to build a new organization. Once the self-organization protocols are completed and the system has changed its state into a new configuration, the agents tell the control agent that they have finished initializing for the new round. The control agent then sends a message to each agent telling him to start the auction phase. On receiving this message, the customer agents emit their orders for the new round by sending call-for-proposals to a number of provider agents. Once all auctions have completed, the customers tell the control agent that the auction phase is over, so he can start another round.

5.4.2 Special Communication for Strategic Networks and Groups

Strategic networks and groups require a lower number of messages for assigning tasks because the task assignment is done by a central instance, the organization's head, who has information about the capacities and available resources of its body agents and the authority to decide what to do with their free capacities.

The information about their resources and capacities is sent to the head by the body agents when forming the organization. This is sufficient for the group, as body agents cannot be members of other organizations, so the head does not need to be informed about changes in resource allocation, as he is the only one in control of this allocation and can keep his own account of it. In the strategic network, however, body agents can be members of several organizations and can accept orders without informing their head about them, if the order asks for a product of a type different from the strategic network's specific product type. Since the head should be kept as up to date about their available resources as possible, body agents send their new capacities to their heads as soon as they accepted an order outside the strategic network.

These additional messages somewhat lessen the advantage of the two organizational forms, but since they are only sent when forming the organizations and when an order has been successfully assigned, it is reasonable to expect the number of messages to be still significantly lower than in other organizational forms.

5.4.3 Special Communication for Corporations

Forming a corporation is an irreversible process; body agents who merge into a new corporation can no longer participate as provider agents for the rest of the simulation. Other agents therefore should not send them any more call-for-

proposals, as this would be useless and reduce the system performance. Corporations are formed from an existing group, which has a dedicated head that will represent the corporation to the outside. All other members of the former group send *leave* messages to all agents in the system, telling them to delete them from the list of provider agents and not to consider them in future auctions. Since it is possible that they receive a call-for-proposals before their leave messages have reached their destinations, they simply refuse any orders sent to them after they have formed a corporation; they do not bounce them by asking the agent sending the order to send it to the corporation head instead.

Chapter 6

Experimental Evaluation

This chapter deals with the empirical evaluation of the proposed concepts with the testbed introduced in the last chapter. Empirical validation is becoming more popular in the field of AI for two reasons: first, increasing computing power makes it easier to perform large experiments [Walsh, 2001]. Collecting enough data in reasonable time to make statistically valid statements was not always possible. The second reason is that there is an increasing awareness that much of current AI research lacks in validation of its results and statements [Cohen, 2002]. Validation methodology, like a sound experimental plan and rigid statistical analyses, that are common in other fields which rely on empirical results and experiments, is starting to gain more acceptance among AI researchers.

To comply with the scientific method, experimental evaluation must start with the formulation of hypotheses, designing the experiments, then proceed with collecting the data, analyzing the results, and finally accepting or rejecting the hypotheses. The initial hypotheses must be motivated by the goal of the research, and can originate from theoretical considerations or exploratory test runs with the system. In the first section of this chapter, we state the hypotheses that we are going to test in our experiments. The second section describes the experimental plan designed to produce the data that can accept or reject the hypotheses. The third and final section presents the results and analyzes the data, drawing conclusions about our stated hypotheses.

6.1 Hypotheses

As stated in the chapter *Problem Description*, the primary goal of this research is to provide an understanding of how certain concepts from the field of sociology that describe forms of cooperation in human societies can be applied to agent societies with beneficial effects on given performance measures. What is of interest to us is therefore how different sociological configurations in agent systems effect the performance of the overall system and that of individual agents.

We have three hypotheses for scenarios containing one form of organization, three for scenarios with different forms of organization coexsting, and three for scenarios examining self-organization. Our working hypotheses are:

Hypothesis 1 In scenarios where all organizations are of the same form (*homogenous scenarios*), the rate of failed orders in the system will be lower for more hierarchically oriented organizational forms.

Failure to assign a task even though the system has sufficient resources is not uncommon in settings where the task cannot be completed by a single agent and other agents need to be sub-contracted, as our experiments have shown. This remains true even if there is no message limit and agents are allowed to send cfps to all providers in an auction. Figure 6.1 shows an example where the timing of the messages is such that two orders AB that consist of elementary types A and B both fail to get assigned, even though the providers could in principle do one of them.

The scenario consists of two customers who each have a task AB, a provider that has resource A and a provider that has resource B. In phase a, the customers send their call for proposals to both providers. The providers check the orders, find that they cannot complete them alone, and each sends two calls for proposals (one for each call for proposals it received) in phase b to the other provider asking for the resource type it cannot produce itself. They answer these calls in phase c with two proposals each. This is followed by sending proposals for the complete order AB to the customers in phase d. Each customer chooses a provider from the two and sends a request for confirmation (phase e). In our case, each customer chose a different provider. In phase f, the providers receive the requests; each allocates its own resource for the order and sends a new request to the other provider. However, in phase g both have to refuse the request, as they already have allocated their resource and no longer have the capacity for the requested type. The providers thus send refuses to the customers in phase h, releasing their resources. Each customer now tries the provider it did not try before; two new requests are sent. The situation now, in phase i, is analogous to the one depicted in phase e, so the same deadlock can happen.

The timing in this example caused a failure because the providers, who should have worked together to complete the order, have two separate "interfaces" to the outside: each can be addressed by a different customer. If the providers had formed an organization that has a single head, like a cooperation, this particular scenario would not have caused a deadlock, as all communication would have gone through the head. This is one reason why we expect that increasing the number of organizations in a multiagent system might lower the rate of failed task assignments. Of course, deadlocks can still happen in organizations: if the timing of the messages happens to be such that reject-proposals, which cause the

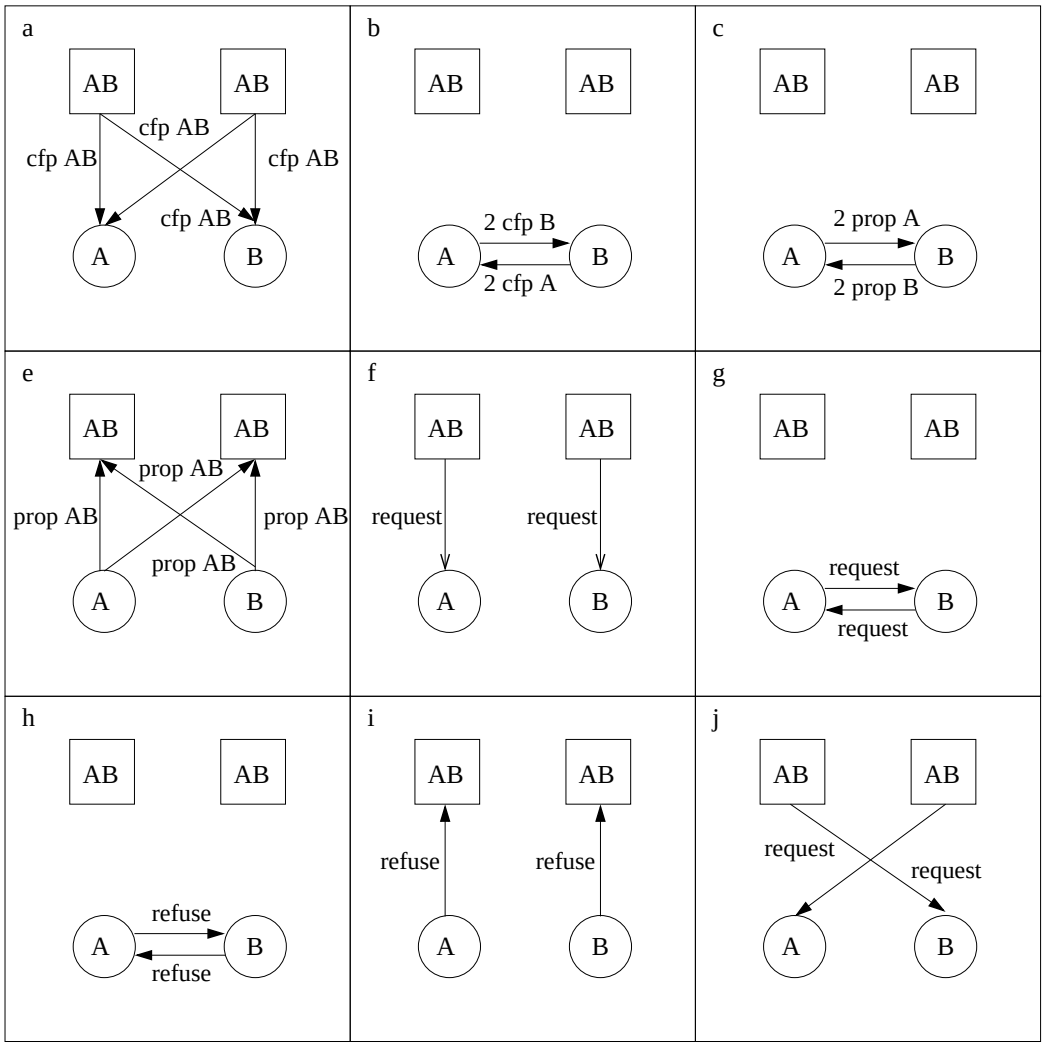

Figure 6.1: Example of a deadlock scenario.

recipient to release the resources it allocated for the order, do not arrive soon enough, then the allocated resources are not available for incoming requests and these requests have to be refused. The effect of organizations on the rate of failed orders is not easy to predict, so the rate of failed orders is a good candidate for a dependent variable. It will be interesting to see how it varies with the different experimental configurations.

In sum, the reason why we expect hierarchically oriented organizational forms to have a lower rate of failed orders is that they funnel the communication through a central instance. In addition, those organizational forms need fewer messages to assign the same tasks, and therefore there should be fewer possibilities for deadlock situations to occur. We expect the rate of failed orders to be highest for single agents and lowest for scenarios with corporations.

Note that agents do only act as organizations if the orders require products specific to the organizations. Otherwise, they act as normal single agents. We assume that the order situation matches the specific products of the organizations to the extent that the organizations do actually have an effect on the system.

Hypothesis 2 In homogenous scenarios, the number of messages in the system will be lower for more hierarchically oriented organizational forms.

Since the protocols for more hierarchically oriented organizational forms tend to use fewer messages, it is reasonable to expect that the total number of messages in such homogenous scenarios will decrease with increasing orientation towards a hierarchy. However, this hypothesis still needs to be tested because it might be possible that other factors influence the number of messages and interact with the effect of slimmer protocols.

Hypothesis 3 In homogenous scenarios, a stricter message limit (number of *calls for proposals* allowed to be sent per auction initiator) will increase the rate of failed orders in scenarios with single agents more than in scenarios with other organizational forms.

If the number of agents that can be sent a call for proposals is limited by the system designer, it becomes more important that the agents asked for a proposal have a high probability of being able to accomplish the task. Agents that form an organization whose specific product matches the product requested by the customer can rely more on their partners to have the capacity required for the task or parts of it, and should therefore be less negatively affected by the higher message limit than agents who have no such relationship to other providers. Along this line of reasoning one might assume that more authoritative organizational forms, like the group, where the head can rely on having the full capacity of its partners at its disposal, will be even less affected than more market oriented organizational forms.

Hypothesis 4 In scenarios where different organizational forms can coexist (*heterogenous scenarios*), the more hierarchically oriented organizational forms will have a higher net income.

The decision of which organizational form to choose depends in large part on the expected income of the organization. Self-interested agents who give up more of their autonomy do so only if the payoff is likely to be higher. The order situation has a large influence on the profitability of organizations. Otherwise, we would only see one kind of organizational form in the real world, the most profitable one. Since we expect that more hierarchically oriented organizations have a lower rate of failed orders, we also expect that they will be more successful in getting task assignments.

Hypothesis 5 In heterogenous scenarios, the presence of single agents will lower the net income of other organizational forms.

Single agents and organizations delegate those parts of the order which they cannot fulfill themselves to external providers. Single agents usually have a lower overall capacity than an organization, and are therefore likely to start more external auctions. They will send cfps to other providers asking for the part of the order complementary to their own resources, including providers in organizations. The organizations have formed to satisfy the demand of the customers, and their specific products therefore probably do not match those the single agents ask for. In such cases, the agents in the organizations process the orders as if they were single agents, and lose any advantage the processing of orders as organizations might have given them. Since more orders are processed by agents acting as single agents, the net income of non-single organizational forms is likely to decrease.

Hypothesis 6 In heterogenous scenarios, a stricter message limit will reduce the net income of single agents more than that of other organizational forms.

This is directly related to hypothesis 3, since agents and organizations that are less successful in getting tasks assigned to them will have a lower income. Note that hypothesis 3 makes a statement about the rate of failed orders, which is a measure of the overall system performance, while hypothesis 6 says something about the performance of individual organizational forms. This hypothesis reflects the assumption that it should be better for agents to form organizations if the order situation is stable enough.

Hypothesis 7 In scenarios where agents start out as single agents and can self-organize to form new organizations, the rate of failed orders of the system will be lower than if self-organization is not allowed.

The way we have implemented self-organization, agents will form increasingly hierarchically oriented organizations as long as the order situation does not change. If our first hypothesis turns out to be correct, we can expect more tasks assignments to succeed because the newly formed organizations can process the orders with fewer chances of a deadlock happening.

Hypothesis 8 In self-organization scenarios where agents start out as single agents, the number of messages in the system will be lower than if self-organization is not allowed.

The hierarchically oriented organizational forms require significantly fewer messages to process orders that request the product they have been built for than more market oriented organizational forms and single agents. As long as the order situation does not change, the provider agents will build increasingly hierarchically oriented organizations, which should lower the number of messages sent in the system.

Hypothesis 9 In self-organization scenarios where agents start out as single agents, the profit per agent will be higher than if self-organization is not allowed.

This is related to hypothesis 7, which states that self-organization will lower the rate of failed orders. A lower rate of failed orders translates into more successful task assignments, which means more money for provider agents. This hypothesis tries to give a motivation for providers to self-organize if they want to maximize their profit.

6.2 Experimental Design

In accordance with Cohen, we use the experimental design methodology known from the social sciences [Cohen, 1995]. This methodology requires the researcher to isolate independent and dependent variables and restate the hypotheses in terms of causal relationships between these two groups. According to him, "understanding something is equivalent to explaining its variance, what statistical methods do" [Cohen, 2002]. We therefore want to check what influence the factors we can control have on the factors we can not control. Due to the large number of resulting variables and possibilities of interactions, we cannot examine all possible influences. We cannot vary all causal factors through their number of possible values; we have to keep some of them fixed and decide what factors to concentrate on. In our experiments, we will not manipulate the variables described in Section 6.2.3, *Fixed Factors*, but we will vary the other independent variables and measure the dependent variables in order to collect data that can either support or reject our hypotheses. This section will describe the independent variables, dependent variables, fixed factors, and the experimental settings by specifying the configuration of the individual experiments.

6.2.1 Independent Variables

Independent variables are those controlled by the experimenter. The experimenter varies these variables across a range of values to create different experimental conditions, and measures the effect of this manipulation on the variation of other variables.

Self-Organization

This binary variable specifies whether agents are allowed to form new organizations or dissolve the ones they are a member of. If set to *false*, agents will remain in the organizational form that is given in their initial configuration files throughout the experiment. If set to *true*, agents can change organizations they

are in if they no longer seem to be efficient, and they can form new organizations if the order situation seems favorable to it.

Homogeneity

This binary variable specifies whether the organizations in a setting are all of the same form or whether different organizational forms can coexist at the same time. Settings with only one organizational form allowed, where this variable has the value *true*, are called *homogenous scenarios*. A value of *false* for this variable specifies a *heterogenous scenario*. This variable does only have an effect in settings where self-organization is not allowed. In settings with self-organization, the dynamics of the re-organizing agents will almost inevitably lead to the coexistence of different organizational forms.

Organizational Forms

This variable specifies the organizational forms in homogenous scenarios. It can have one of the six values *single agents*, *virtual enterprise*, *cooperation*, *strategic network*, *group*, and *corporation*. This variable has no effect in heterogenous scenarios.

Existence of Single Agents

This binary variable specifies whether the scenario contains agents which are not members of an organization or not. It is only considered in heterogenous scenarios, and has no effect in homogenous ones.

Order Situation Change

This scalar variable determines the percentage of orders that change in a round when the old order profile (the configuration of order that the customers emit) is replaced by a new one. A value of 50 in this variable means that 50% of the customers change the order they emit each round. Changes in the order profile happen at certain rounds in the simulation, as described later in the section *Fixed Factors*. In our experiments, we will only use two values for this variable to limit the number of experimental configurations to a manageable size. One change in order situation will be 0%, the other 50%.

Message Limit

This scalar variable specifies the number of cfps agents can make to providers outside their own organization when starting a new auction for a resource that they or their organization cannot produce on their own. We briefly considered having *unlimited* as a possible value for this variable, but it turned out that the

time required for a single round to complete with a reasonable number of agents and a reasonable complexity of products (i.e., the number of elementary types they are composed of) is too great to conduct a number of experiments sufficient for statistical analysis. We only use message limits of 8 and 12 in our experiments.

6.2.2 Dependent Variables

Dependent variables are those that are not directly controlled and manipulated by the experimenter, but are supposed to be affected by the independent variables. Dependent variables are what is being measured and variation in their values has to be traced back to different experimental configurations.

Rate of Failed Orders

The rate of failed orders is a scalar variable that measures the percentage of customer orders in each round that were not assigned. It does not include orders emitted by provider agents to subcontract other providers for a part of an order composed of subtypes.

Net Income per Organizational Form

The net income per organizational form is a scalar variable that measures, for each round, the income of an organizational form by summing up the net incomes of all organizations of this form. The net income per organization is computed by summing the income of all orders the organization has been assigned to this round and subtracting the money it has paid to external agents.

Profit per Provider

The profit per provider is a scalar variable that measures, for each round, the average profit over all providers. The profit for a single provider is computed by the sum of profits for all orders it got assigned this round. The profit for a single order, in turn, is computed by taking the difference between the money it received for this order and its costs for the resources spent to complete it.

Number of Messages

We measure two kinds of messages in the system: auction-related messages and self-organization-related messages. The number of auction-related messages is a scalar variable that measures, for each round, the total number of auction-related messages sent in the system. Auction-related messages are those messages that are sent in accordance with one of the protocols *Holonic Contract Net with Confirmation Protocol*, *Authority with Confirmation Protocol*, or *Authority without Confirmation Protocol*. Self-organization related messages are those messages

that are sent in accordance with one of the protocols *Self-organization Protocol for Creating New Organizations*, *Self-organization Protocol for Existing Organizations*, or *Voting Protocol*. Other messages, like communication between the agents and the agent controlling the simulation, are not counted.

6.2.3 Fixed Factors

The independent variables listed above are not the only ones that are under the experimenter's control and affect the dependent variables. A number of other factors have a significant causal influence and could be manipulated to study the relationships of factors in our market-based model more thoroughly. However, we already have quite a number of different configurations each specifying an experimental scenario, and for each configuration we need data from several experimental runs to draw statistically valid conclusions. The experimental runs differ even if the configuration is the same, because there are random influences like the order arriving messages. Considering that each run takes a lot of computational resources, we had to limit our empirical examinations to manipulating only a subset of controllable variables. We had to choose plausible values for the rest and keep them fixed throughout the experiments. If more time was available, a deeper investigation might inquire into how these choices influenced our results. We now present a list of factors that were kept fixed in our experiments, but could be manipulated in future evaluations of the model.

Period of Order Situation Change

The time at which we change the order situation determines how much time the agents have to adopt to the new situation before it changes again. We chose a time frame of 50 rounds between the start of the simulation and the order situation change, as it turned out that 50 rounds are enough to stabilize the system's behavior to a reasonable extent.

Parameters of Self-Organization

The choice when to build a new organization with other providers is currently coded as thresholds for trade volume between agents. It is likely that varying these thresholds might influence the results somewhat, but this influence is mostly limited to the speed of adaptation, and given the relatively large period of order situation change, agents have enough time to adapt to a given order situation. The chosen values should therefore be representative enough for our model of self-organization. The threshold to *upgrade* is 0.6, for *downgrade* 0.2. The minimum duration for an organization is five rounds. The threshold for building new organizations is 0.3.

Trading Parameters of Customer and Provider Agents

The customer and provider agents in our experiments have always the same configuration of resources, cost, demand, profit parameters, and gift parameters. Provider agents can each produce one of four elementary resource types, in a quantity of one unit per round. There are three cost levels: 4, 5, and 6 Euros per unit. The profit parameter for all agents, customers and providers, is set to 1.1, and the gift parameter to 1.0. At this value, the gift parameter is neutral, i.e. the agents act in the same way as if there was no gift mechanism in the system. They neither give gifts nor consider the gift history when dealing with other agents. We therefore did not examine gift giving in our experiments.

Preference for Known Agents

The preference for provider agents has an impact when either selecting agents to send a *cfp* to or determining the willingness to form an organization with other agents. This preference is computed by taking into account the trade history between the agents. Each agent assigns a scalar value to provider agents it knows by summing up the past trade volume of the elementary types present in the current order. There might be different ways of computing a preference value based on order history, but the resulting ordering of provider agents should not be that much different.

6.2.4 Experimental Scenarios

This section describes the individual experimental scenarios, specifying what variables we varied and measured in each scenario and the values for the independent variables involved. Appendix A gives an overview of the configurations in tabular form.

6.2.5 Scenario for Hypotheses 1-3

This scenario contains 60 provider agents for each of the types A, B, and C. Each type occurres in the price levels 4, 5, and 6. There are 60 customer agents emitting one order of type ABC each round, with the deadline of the current round. There is no self-organization allowed. There are six different configurations, one for each organizational form. In each configuration, the organizations are of the same form. Each organization has three members agents, one for each of the three types. One member agent has cost 4, one cost 5, and one cost 6. The message limit for hypotheses 1 and 2 is 12, for hypothesis 3 the message limit is varied and takes the value 8 in one set of configurations and 12 in another.

6.2.6 Scenario for Hypothesis 4

The agents are the same as for the first three hypotheses. The message limit is set to 12, there is no self-organization allowed. There are no single agents; all agents are organized in non-overlapping organizations. There are twelve organizations of each of the five organizational forms. The organization size, type and price configuration is the same as for hypotheses 1-3.

6.2.7 Scenario for Hypothesis 5 and 6

This scenario contains 36 provider agents for each of the types A, B, and C, each type occurring in the price levels 4, 5, and 6. There are 36 customer agents emitting one order of type ABC each round. There is no self-organization allowed. The scenario contains all five organizational forms and single agents. There are six organizations of each form and 18 single agents. The organization size, type and price configuration is the same as for hypotheses 1-3. The message limit is 12 for hypothesis 5, for hypothesis 6 the message limit is varied and takes the value 8 in one set configurations and 12 in another.

6.2.8 Scenario for Hypotheses 7-9

This scenario contains 30 providers of each of the types A and D, and 60 providers of each of the types B and C. Each type occurs in the price levels 4, 5, and 6. There are 30 customers who emit order ABC in each round and 30 customers who emit order ABC in rounds 1-50 and order BCD in rounds 51-100. The message limit is 12. There are no organizations at the start. In one configuration self-organization is allowed, but not in the other.

6.3 Results and Discussion

This section describes the results for the tests of each hypothesis and discusses them. Each graph is a summation over 100 simulation runs. The results are summarized at the end of the section.

6.3.1 Hypothesis 1

> *In scenarios where all organizations are of the same form (*homogenous scenarios*), the rate of failed orders in the system will be lower for more hierarchically oriented organizational forms.*

Figure 6.2 shows the rate of failed orders of all six organizational forms in one setting. The rate decreases for all organizational forms, because the agents learn which providers can provide which resources by building up their preferences in

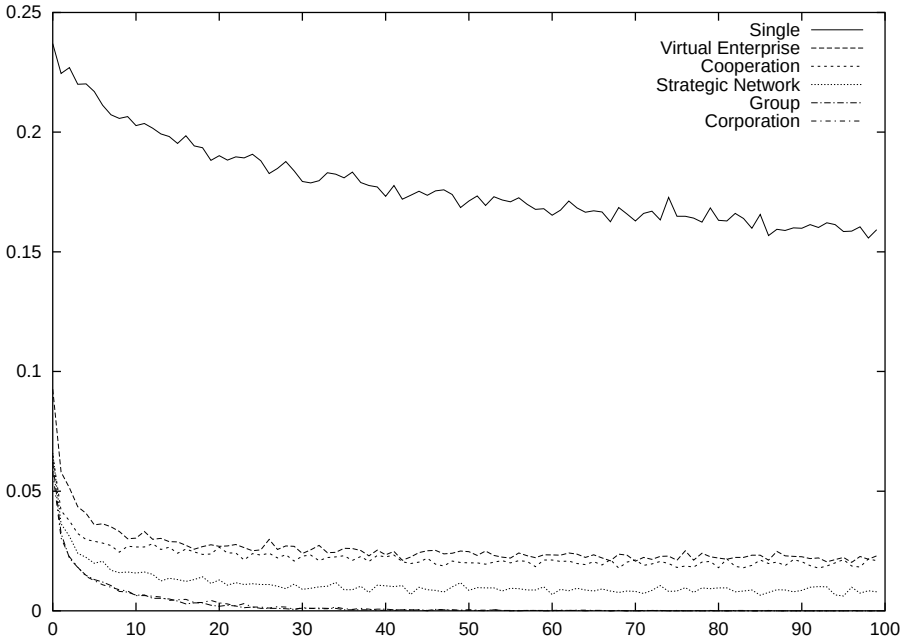

Figure 6.2: Rate of failed orders of all six organizational forms. Single agents are least successful in assigning orders.

the agent models. The rate of failed orders is significantly higher for single agents than for the other organizational forms.

Figure 6.3 leaves out the single agents and zooms in on the other five organizational forms. The organizational forms build three clusters: the clusters differ in rate of failed orders from each other, but organizational forms within each cluster do not. The first cluster is formed by the virtual enterprise and the cooperation. These two organizational forms are the least effective of the five in terms of rate of failed orders, but still significantly better than single agents. The fact that the communication is channeled through one agent in the case of the cooperation and through several in the case of the virtual enterprise does not seem to influence the rate. The strategic network forms a cluster on its own, its efficiency being somewhere between that of the virtual enterprise/cooperation cluster and the group/corporation cluster. The increase of efficiency over the first cluster might be explained by the reduced protocol; unlike the virtual enterprise and the cooperation, the strategic network does not need to collect proposals from its body agents. It therefore seems that the proposal phase is a potential cause of deadlocks. The group and corporation have an even lower rate of failed orders, probably because they use a protocol that has neither a proposal phase nor a confirmation phase. The communication of informing the body agent about an assigned order that is necessary in the group but not the corporation does not seem to have an effect on the rate of failed orders. This is plausible, because unlike the proposal and confirmation communication, it cannot cause any conflicts

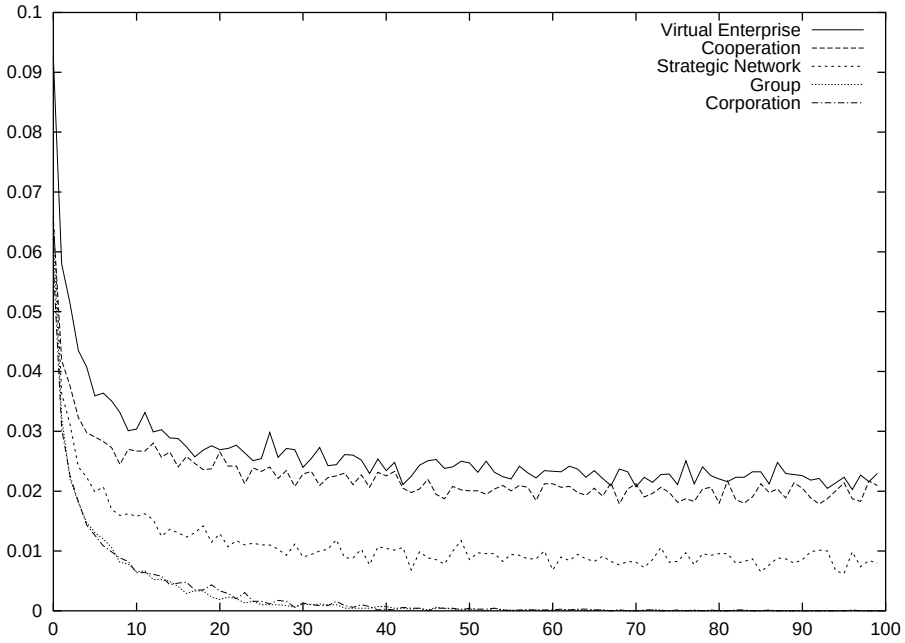

Figure 6.3: Rate of failed orders of the non-single organizational forms. Virtual Enterprise and Cooperation are least, Group and Corporation most successful. The Strategic Network falls between the two groups.

in resource allocation with unfortunate message timing, which happened in the deadlock example described in Section 6.1.

6.3.2 Hypothesis 2

> *In homogenous scenarios, the number of messages in the system will be lower for more hierarchically oriented organizational forms.*

Figure 6.4 shows the number of messages of all six organizational forms together. The number of messages is significantly higher for single agents than for the other organizational forms. It stays relatively constant for the five non-single organizational forms, but increases slightly for single agents. Simulations of variations of this scenario showed that the increase disappears if either a) all providers have the same price b) there is no message limit or c) the agents do not learn preferences. The explanation for the increase is therefore that the agents, by learning preferences, increasingly only ask agents who have the requested resources, hence get more *proposals* instead of *refusals*, and due to the varying costs receive more *second proposals* than *agrees*, which often causes further requests to agents with cheaper proposals. The non-single organizational forms do not show the increase because they all can produce the product for the same price, as each organization has exactly one agent of one of the three price classes.

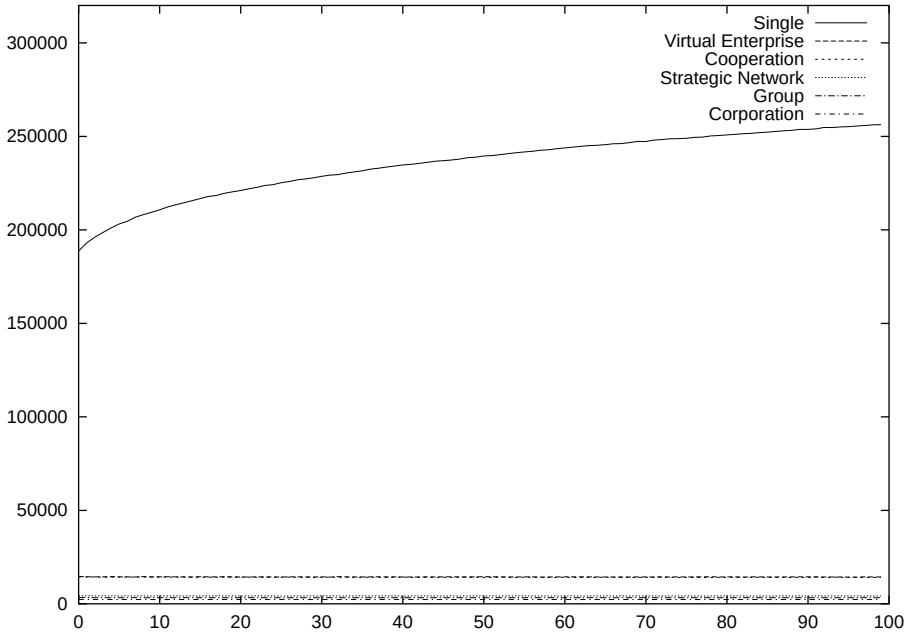

Figure 6.4: Number of messages of all six organizational forms. The number is highest for single agents, and increases with time.

Figure 6.5 leaves out the single agents and zooms in on the other five organizational forms. It shows that the number of messages is about the same for the virtual enterprise and the cooperation, with both being significantly less efficient than the strategic network, the group, and the corporation. The explanation for this is that the virtual enterprise and the cooperation are both based on a full internal HCNCP, whereas the strategic network leaves out the proposal phase, the group the proposal and the confirmation phase, and the corporation all internal communication.

6.3.3 Hypothesis 3

> *In homogenous scenarios, a stricter message limit (number of* calls for proposals *allowed to be sent per auction initiator) will increase the rate of failed orders in scenarios with single agents more than in scenarios with other organizational forms.*

Figure 6.6 shows, for all organizational forms, the difference between the rate of failed orders in scenarios with message limit 12 and 8. The absolute difference is significantly larger for single agents than for the other organizational forms. Plotting the relative difference is not clarifying because the values of failed orders for the non-single organizational forms fluctuate around zero, causing the graph for the relative difference to behave chaotically. The reason why the difference

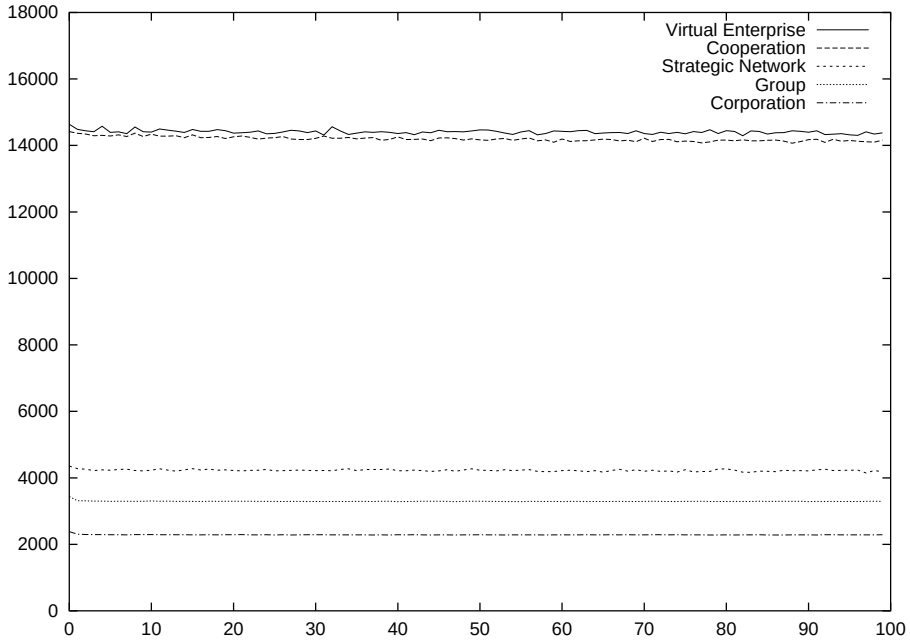

Figure 6.5: Number of messages of the non-single organizational forms. Virtual Enterprise and Cooperation are less efficient than the other three organizational forms. More hierarchical forms use fewer messages.

is negative for some organizational forms is probably that a message limit that allows more messages to be sent allows more deadlocks to occur.

6.3.4 Hypothesis 4

> *In scenarios where different organizational forms can coexist (*heterogenous scenarios*), the more hierarchically oriented organizational forms will have a higher net income.*

Figure 6.7 shows the income of the five organizational forms, when competing in the same scenario. The income does not differ significantly, with the exception of the corporation. The reason why the corporation performs so poorly, especially at the beginning, is that it consists of only one agent, while the other organizational forms consist of three. Since at the beginning all provider agents are sent a cfp with the same probability, the corporation is less likely to be included in an auction when the number of messages is limited. As the agents build up their preferences however, the corporation's income rises towards the level of the other organizational forms.

It seems that the advantage of a lower rate of failed orders of more hierarchically oriented organizations in homogenous scenarios does not translate into an increased income in scenarios where they have to compete with other organiza-

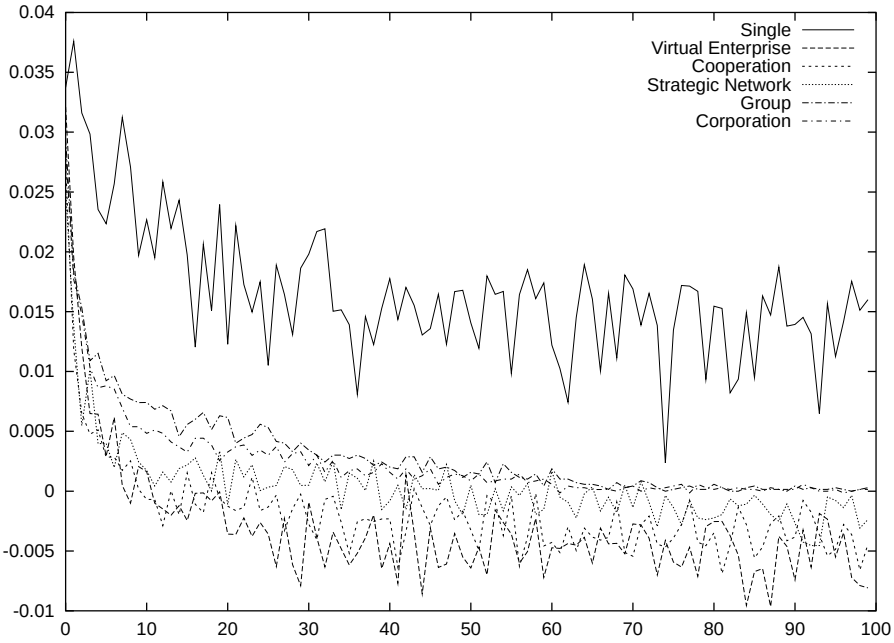

Figure 6.6: Difference between rate of failed orders for message limits 8 and 12. Single Agents experience the highest drop in successful task assignments when fewer messages are allowed.

tional forms. It is however possible that this can be attributed to a ceiling effect, i.e., the rate of failed orders in this heterogenous scenario is minimal, and that the hypothesis would turn out to be true for another part of the configuration space that we have not examined.

6.3.5 Hypothesis 5

> *In heterogenous scenarios, the presence of single agents will lower the net income of other organizational forms.*

Figure 6.8 shows the net income of all six organizational forms. The income of the three product-based organizational forms, virtual enterprise, cooperation, and strategic network, quickly falls to very low levels. The reason for this is that the single agents introduce a number of sub-orders into the system that is far greater than the number of original orders, thus the product of most incoming orders that an agent as part of an organization receives does not match the product of its organization. Most products are therefore not processed by product-specific organizations. The non-product specific organizational forms group and corporation are not that vulnerable to the disruptive presence of single agents, hence their income does not drop similarly. Since most agents act as single agents, the income of the single agent form is higher than of the group and corporation.

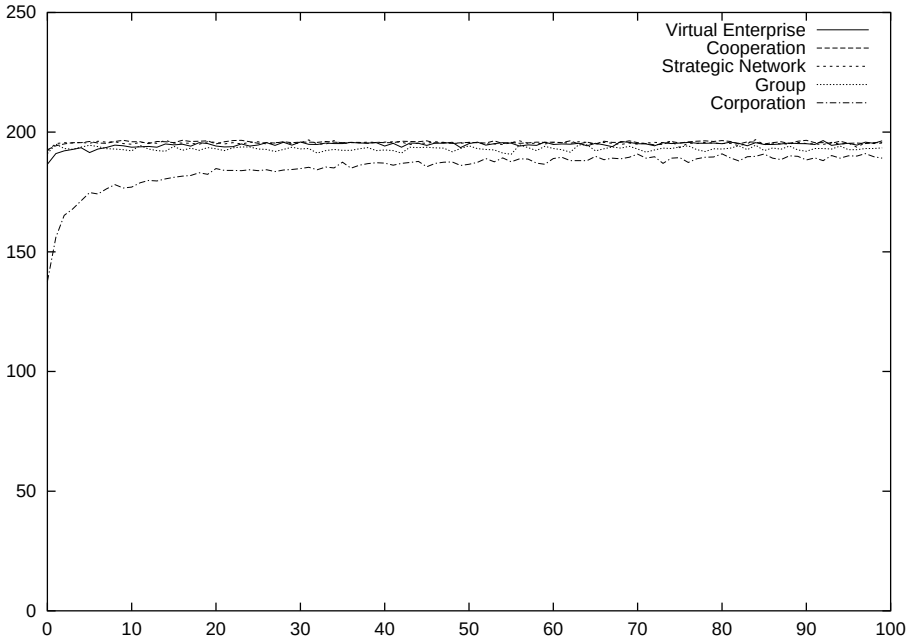

Figure 6.7: Income of the organizational forms in a heterogenous scenario without single agents. All are roughly at the same level, except for the Corporation, which is slightly worse at the beginning.

Figure 6.9 leaves out the single agents and zooms in on the other five organizational forms. It shows that the income for corporation is slightly better than the income for groups, probably because its smaller protocol has a lower chance of deadlocks.

6.3.6 Hypothesis 6

> *In heterogenous scenarios, a stricter message limit will reduce the net income of single agents more than that of other organizational forms.*

Figure 6.10 shows the difference between the income in scenarios with message limit 12 and 8, for all organizational forms. The advantage of organizations over single agents in income difference seems to be limited to the first third of the simulation. When the agents have learned better preferences, the single agents get more *call for proposals* matching their resources and become more efficient at disrupting the product-specific organizational forms. This can be seen very clearly in the fact that at the beginning, the curve for the product-specific organizational forms is strongly negative and mirrors the curve for the single agents. The non-product-specific organizational forms group and corporation are less affected.

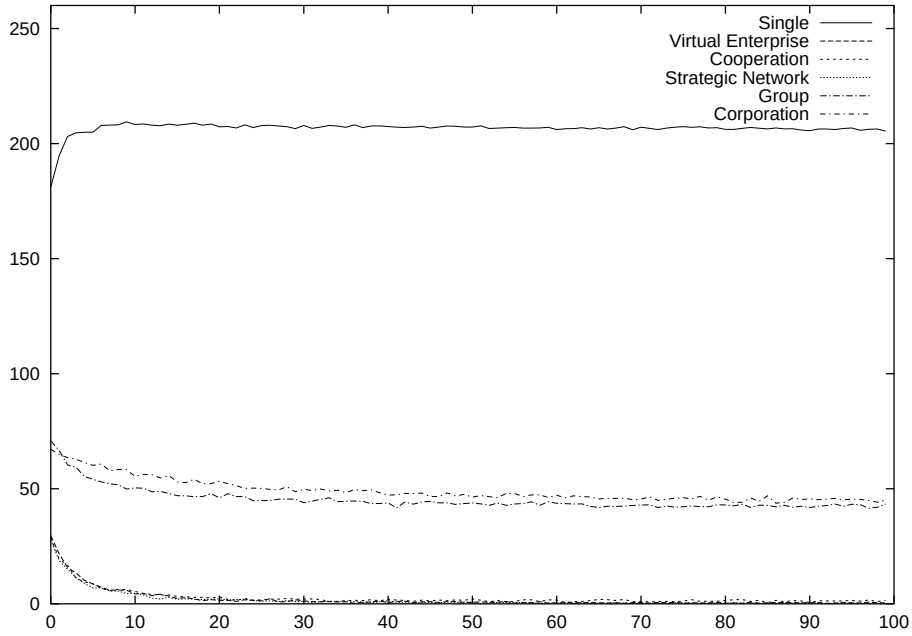

Figure 6.8: Income of all six organizational forms in a heterogenous scenario with single agents. Single agents have the highest income.

6.3.7 Hypothesis 7

In scenarios where agents start out as single agents and can self-organize to form new organizations, the rate of failed orders of the system will be lower than if self-organization is not allowed.

Figure 6.11 shows the rate of failed orders in the system with and without self-organization. Self-organization does not seem to provide the expected reduction rate of failed orders. Both show a falling tendency, which can be explained by the learning of preferences. The preferences are also responsible for the fact that the rate does not increase after the order situation has changed. The agents have learned which provider can do which resource, so that an order situation change in which most of the elementary types are the same as before does not result in more failed orders. The potential advantage in reduction of the rate of failed orders of organizations that has been shown in hypothesis 1 seems to be neutralized in scenarios with self-organization, probably because at most stages in the simulation there are still single agents. These single agents have a disruptive effect, as indicated by the results of the experiments for hypothesis 5.

6.3.8 Hypothesis 8

In self-organization scenarios where agents start out as single agents, the number of messages per system will be lower than if self-

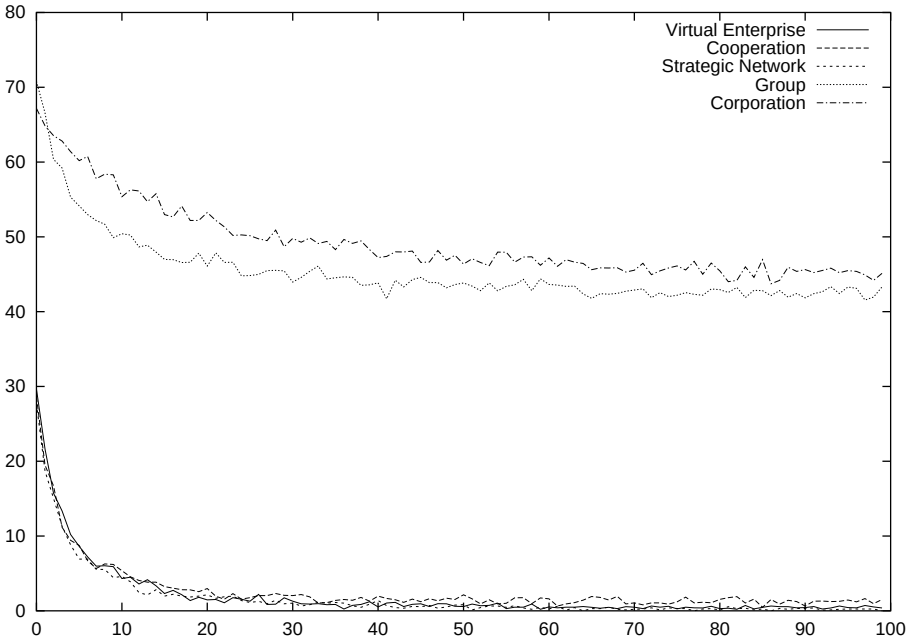

Figure 6.9: Income of the non-single organizational forms in a heterogenous scenario with single agents. Except for the Group and the Corporation, the income of all organizational forms quickly falls to very low levels.

organization is not allowed.

Figure 6.12 shows the number of messages in the system with and without self-organization. The number of messages is significantly lower when self-organization is allowed. In both cases, the number increases due the reasons discussed in hypotheses . The advantage of self-organization is lower after the order change, most likely because the corporations, which have formed in the old order situation, cannot resolve and adapt to the new situation, unlike the other organizational forms.

The messages counted in figure 6.12 are only the auction-related messages. Figure 6.13 shows the number of messages that are used in communication for self-organization. This number is significantly below the difference between the graphs shown in figure 6.12, so the overall number of messages in self-organizing systems is still significantly lower than in non-self-organizing systems.

6.3.9 Hypothesis 9

In self-organization scenarios where agents start out as single agents, the profit per agent will be higher than if self-organization is not allowed.

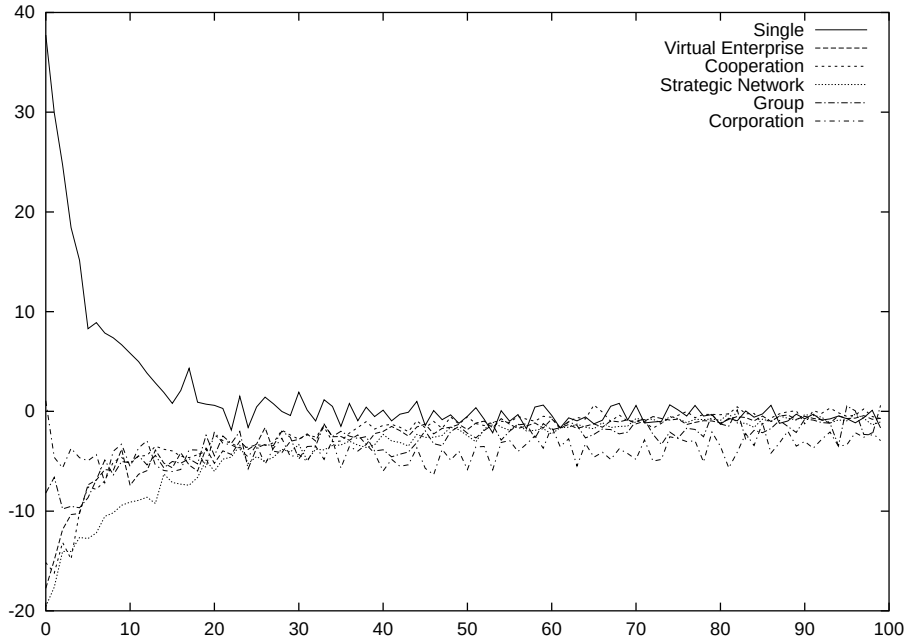

Figure 6.10: Income difference between scenarios with message limit 8 and 12 for all organizational forms. The more negative effect of a stricter message limit on single agents only shows at the beginning.

Figure 6.14 shows the profit per provider with and without self-organization. The two graphs do not differ in the way predicted by the hypothesis. The profit per provider is about the same with and without self-organization. This is related to the results of hypothesis 7: since self-organization does not provide an advantage in the rate of failed orders, it also does not increase the profit per provider.

6.3.10 Summary

Many of the tested hypotheses turned out the way we predicted, while some showed surprising results. The predicted results are:

- Organizations have advantages in terms of number of messages and rate of failed orders in homogenous scenarios.
- In general, the more hierarchically oriented an organization, the lower its rate of failed orders and the number of messages in homogenous scenarios.
- Organizations are less affected by a tighter message limit than single agents both in heterogenous and homogenous scenarios.
- Single agents disrupt the efficiency of product-specific organizations.

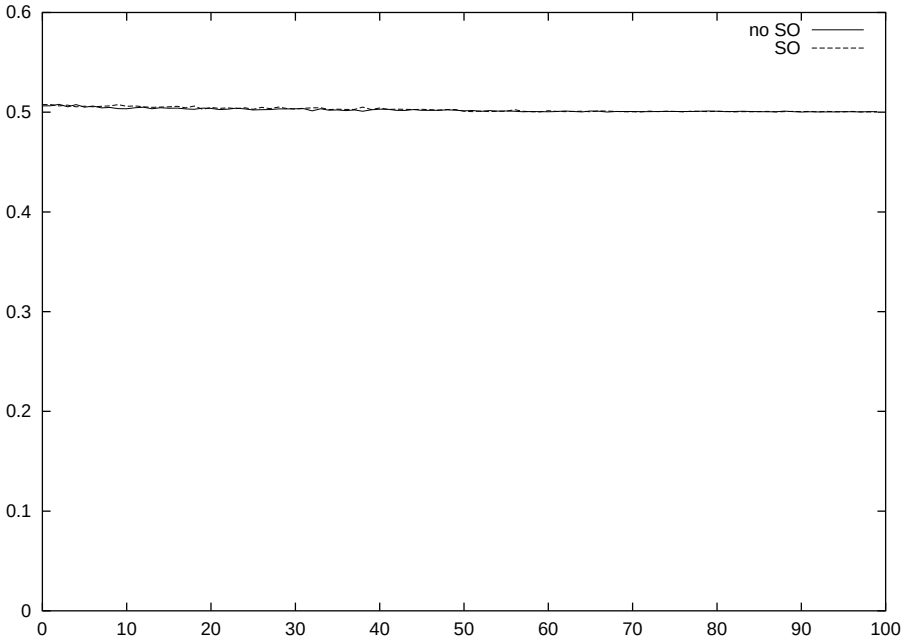

Figure 6.11: Rate of failed orders in the system with and without self-organization. The rate does not seem to be affected by self-organization.

- Self-organization reduces the number of messages.

The unexpected results are:

- The virtual enterprise and the cooperation do not differ much in terms of rate of failed orders and number of messages.
- The net income of organizational forms in heterogenous scenarios does not differ significantly.
- Self-organization does not influence the rate of failed orders or the profit per provider.

On first sight it seems that the group is the best organizational form to use in all cases. However, such a conclusion is not warranted by our experiments. First, we did not use overlapping organizations in the scenarios without self-organization, which means that a significant advantage of "lower" organizational forms over the group was neutralized. Second, we did not investigate whether self-organization with Group as organizational form allowed is better for the system or individual agents than self-organization with Group not allowed, it is therefore not clear what importance the Group form has for self-organizing systems. Experiments in this direction are an interesting possibility for future work.

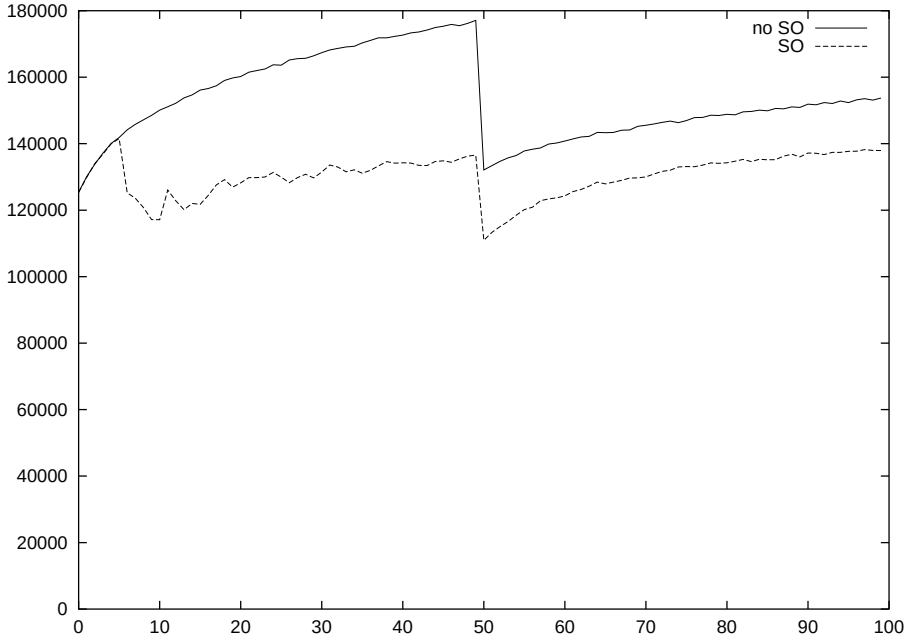

Figure 6.12: Number of auction-related messages in the system with and without self-organization. Self-organization reduces the number of messages sent in the system by several ten thousands.

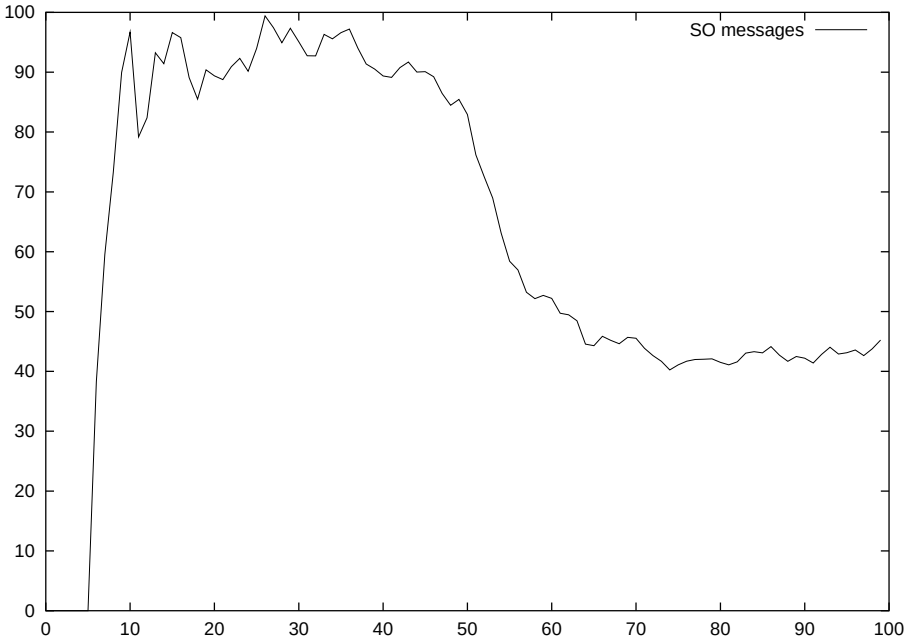

Figure 6.13: Number of self-organization messages. This number is around 100, which is much lower than the tens of thousands auction-related messages saved by self-organization.

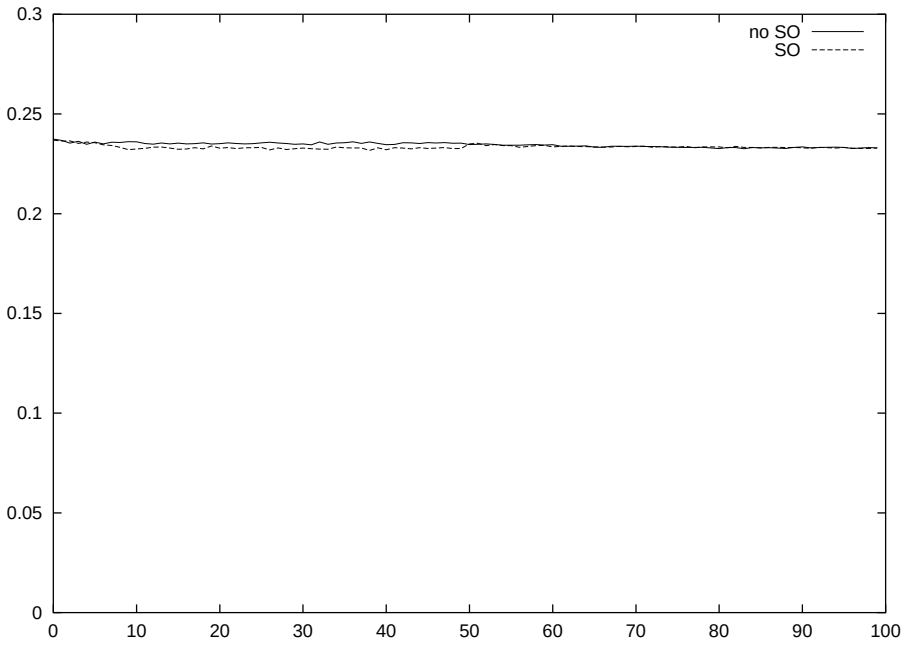

Figure 6.14: Profit per provider with and without self-organization. Self-organization does not seem to influence the profit.

Chapter 7

Conclusions

This chapter summarizes the contributions of this thesis and lists further research questions that might be worth investigating based on the current work.

7.1 Summary

We describe how the three major goals of the thesis have been met: the specification of organizational forms and a mechanism for self-organization, the development of protocols, and the experimental evaluation.

7.1.1 Specification of Organizational Forms and a Mechanism for Self-organization

We chose six different organizational forms from sociological theories and gave a detailed specification for their implementation in a multiagent system. The six organizational forms are: single agents, the virtual enterprise, the cooperation, the strategic network, the group, and the corporation. These organizational forms differ from each other in their orientation towards a market or a hierarchy. The stronger the orientation towards a hierarchy, the more do the agents commit themselves to give away part of their autonomy. We have also specified a mechanism for forming new organizations and adapting existing ones to new situations. The mechanism for self-organization is supposed to allow the agents to increase some local or global performance by acting on locally available information. The premise of the thesis was that using the organizational forms specified increases can increase the efficiency of multiagent systems and that the agents can effectively decide on their own how to organize to reach this goal.

7.1.2 Development of Protocols

In order to ensure interoperability of different systems using the concepts proposed in the thesis and to facilitate compatibility with other multiagent projects, we used the FIPA standard to realize all agent communication. We extended existing protocols and developed new ones to enable the different organizational forms to efficiently solve their communication tasks. The Holonic Contract Net with Confirmation Protocol is an extension of the traditional Contract Net Protocol. It is used in a reduced form in the Authority Protocol with Confirmation and the Authority Protocol without Confirmation. Among the new protocols are protocols for coordinating the updating process of existing organizations, the forming of new organizations, and the election of a new head agent.

7.1.3 Experimental Evaluation

To test whether the concepts proposed can be usefully applied to multiagent systems, we have defined a number of performance measures and investigated the effect of different organizational forms and of the process of self-organization on these measures. The measures are the number of messages sent in the system, the rate of failed task assignments, the net income of organizational forms, and the average profit of provider agents.

Our experimental evaluation showed that organizations and self-organization can increase the efficiency of multiagent systems. Whether to pre-design the organizations or use self-organization depends on the specific conditions of the scenario. Our results suggest that if the order situation does not change and it is important to have a low rate of failed task assignments and few messages used for the auctions, it is advisable to either put all agents into organizations such that no single agents are left or, if single agents have to be present, make all organizations either groups or corporations. If the order situation does change, then starting all agents as single agents and allowing them to self-organize will still provide the system with the advantage of using fewer messages, without increasing the number of failed task assignments.

7.2 Future Work

Organization in multiagent system is a wide topic and the present thesis could only touch a fraction of the unanswered research questions in this field. Almost every aspect of the system we used opens a plethora of further research possibilities. We list some of them here, although the list is not complete.

Parameter Variation The most obvious further research would involve varying the parameters that were kept fixed in the current implementation and investigating the effect of the variations on the performance measures. Exam-

ples for these fixed parameters are the thresholds for self-organizations, the resource capacities of provider agents, the deadlines for the tasks, the complexity of orders, the number of agents, and the customer demand/available provider resources relationship.

Intra-organizational Differences The organizations we specified divide the member agents into two classes: head and body agents. The difference in roles has two main effects: head agents interact stronger with the outside, reaping potential benefits like a better reputation or higher customer preference as providers to assign tasks to. Secondly, the different profit distribution models for each organizational forms result in different provider profits within the organization. Future research might investigate the consequences of these differences in more detail.

More Complex Order Changes In the scenarios we investigated, the order situation either did not change at all or consisted in a switch from one stable profile to another. A possible alternative is to have a certain percentage of orders be determined randomly. For example, 90% of the orders might be "ABC", while the remaining 10% would be chosen randomly from the set {BCD, ABD, ACD}.

Failures In our scenarios providers never fail to complete an accepted order, their failure probability is 0%. If one increases this failure probability, new concepts like reputation and trust become interesting. Agents will have to learn which other agents are reliable and which have a high failure probability.

Dropouts Organizations might prove to be resilient against agents that at some point in the simulation drop out seizing all activity and no longer responding to messages. Organization heads that find one of their body agents has dropped out could use this information to keep the remaining orders that this organization processes from failing by reassigning parts of them.

Gift Giving Although we have implemented the mechanisms for gift giving, we did not have the time to investigate its effects on the performance measures. Gift giving is a concept that might be especially interesting if one also uses the concepts trust and reputation.

Punishments We have specified the organizations in the ADICO grammar, which is flexible enough to include punishments for violation of rules. For example, a body agent of a strategic network that fails to forward a call for proposal to its head and decides to process the order itself might be required to pay a certain amount of money to the head if its behavior is found out. Punishments do only make sense however if the agents have the

capacity to reason about the consequences of their behavior, which would introduce a complexity that is beyond the scope of the current work.

Recursive Structures The self-organization mechanism we implemented results in flat organizational structures: organizations can overlap, but an organization can not be the body agent of another organization. If this restriction is dropped, agents can organize in fractal structures that increase the complexity of the system.

AI Techniques The agents could be made more complex by implementing techniques known from research in artificial intelligence: using different kinds of logic, genetic algorithms, neural nets, etc. More complex agents might be better able to adapt their organizational structure to the current and expected future situations.

Better Internal Auctions The virtual enterprise and the cooperation might be made more efficient if the head of the organization would only ask those agents to complete an order who actually have the resources to do it. This would require that the agents tell each other about the type of resources they can do e.g. during the formation of the organization.

Customer Profit In measuring profit, we concentrated on the side of the providers. An extension of this research would be the investigation of how profitable the concept of organizations and self-organization is for the customers.

Customer Organizations Similar to the last point, one could investigate what kind of organizations on customer side would increase the performance of the system.

Choice on Provider Side In our scenarios, customers choose from a number of providers. But it is principally also possible that the providers collect call for proposals from different customers and choose which customer to send a proposal to. It might be interesting to investigate the effects of organizations on market systems where providers have more complex choice options.

Broadcasting the Winner An alternative to the learning of preferences used here is the method of sending the name of the winner of an auction in the reject-proposal messages. In this way, the other agents get to know who is a successful provider and therefore might be a valuable partner in a new organization.

Appendix A

Configuration of the Experiments

The left table shows provider configuration, the right table customer configuration. Customers can emit one type of order in rounds 1-50 and another type in rounds 51-100.

Agent configurations for hypotheses 1-4:

quantity	*type*	*costs*
20	A	4
20	A	5
20	A	6
20	B	4
20	B	5
20	B	6
20	C	4
20	C	5
20	C	6

quantity	*type* $1-50$	*type* $51-100$
60	ABC	ABC

Agent configurations for hypothesis 5 and 6:

quantity	*type*	*costs*
12	A	4
12	A	5
12	A	6
12	B	4
12	B	5
12	B	6
12	C	4
12	C	5
12	C	6

quantity	*type* $1-50$	*type* $51-100$
36	ABC	ABC

Agent configurations for hypotheses 7-9:

quantity	*type*	*costs*
10	A	4
10	A	5
10	A	6
20	B	4
20	B	5
20	B	6
20	C	4
20	C	5
20	C	6
10	D	4
10	D	5
10	D	6

quantity	*type* $1-50$	*type* $51-100$
30	ABC	ABC
30	ABC	BCD

Configurations specific to the hypotheses:

In the table, set notation means that the corresponding variable takes on one of the values of the set for a configuration. For example, hypothesis 3 has 12 different configurations, because *message limit* has two possible values and the variable *organizations at start* six.

hypothesis	*SO?*	*message limit*	*organizations at start*
1	no	12	{none, 60 ve, 60 coo, 60 sn, 60 grp, 60 cor}
2	no	12	{none, 60 ve, 60 coo, 60 sn, 60 grp, 60 cor}
3	no	{8,12}	{none, 60 ve, 60 coo, 60 sn, 60 grp, 60 cor}
4	no	12	12 ve 12 coop 12 sn 12 grp 12 cor
5	no	12	6 ve 6 coop 6 sn 6 grp 6 cor
6	no	{8,12}	6 ve 6 coop 6 sn 6 grp 6 cor
7	{yes, no}	12	none
8	{yes, no}	12	none
9	{yes, no}	12	none

Abbreviations:
SO = self-organization
sin = single
ve = virtual enterprise
coo = cooperation
sn = strategic network
group = group
cor = corporation

When organizations are specified at the start, they always have three member agents, one for each type. One member agent has costs 4, one costs 5, and one costs 6.

Bibliography

[André et al., 1990] André, J., A. Mouginot, and M. Venet (1990). A framework for dynamic reorganization. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 31–37.

[Ashby, 1947] Ashby, R. (1947). Principles of the self-organizing dynamic system. *Journal of General Psychology*, 37:125–128.

[Barnard, 1968] Barnard, C. I. (1968). *The Functions of the Executive*. Harvard, Cambridge, 30th anniversary edition.

[Bürckert et al., 1998] Bürckert, H.-J., Fischer, K., and Vierke, G. (1998). Transportation scheduling with holonic mas–the teletruck approach. In *Proceedings of the Third International Conference on Practical Applications of Intelligent Agents and Multiagents (PAAM'98)*.

[Bratman et al., 1988] Bratman, M. E., Israel, D., and Pollac, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355.

[Brooks and Durfee, 2002] Brooks, C. and Durfee, E. (2002). Congregating and market formation. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 96–103.

[Brooks, 1986] Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23.

[Bürkert et al., 2000] Bürkert, H., Fischer, K., and Vierke, G. (2000). Holonic transport scheduling with TELETRUCK. *Applied Artificial Intelligence*, 14(7):697–726.

[Bussmann, 1998] Bussmann, S. (1998). An agent-oriented architecture for holonic manufacturing control. In *Proceedings of 1st Int. Workshop on Intelligent Manufacturing Systems*, pages 1–12.

[C. Gerber, 1999] C. Gerber, J. Siekmann, G. V. (1999). Holonic multi-agent systems. Technical report, German Research Center for Artificial Intelligence (DFKI).

[Camarinha-Matos and Afsarmanesh, 1998] Camarinha-Matos, R. R. L. and Afsarmanesh, H. (1998). Multi-agent perspectives to agile scheduling. In Rabelo, R., Camarinha-Matos, L., and Afsarmanesh, H., editors, *Intelligent Systems for Manufacturing*, pages 51–66. Kluwer Academic Publishers.

[Castelfranchi and Falcone, 1998] Castelfranchi, C. and Falcone, R. (1998). Towards a theory of delegation for agent-based systems. *Robotics and Autonomous Systems*, 24:141–157.

[Cohen, 1995] Cohen, P. (1995). *Empirical Methods for Artificial Intelligence.* The MIT Press, Cambridge, MA.

[Cohen, 2002] Cohen, P. (2002). Empirical methods for analysis of agent systems. Tutorial held at the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002).

[Cohen and Levesque, 1987] Cohen, P. and Levesque, H. (1987). Intenion = choice + commitment. In *Proceedings AAAI-87*, pages 410–415.

[Crawford and Ostrom, 1995] Crawford, S. E. S. and Ostrom, E. (1995). A grammar of institutions. *American Political Science Review*, 89(3):582–599.

[Durfee et al., 1989] Durfee, E., Lesser, V., and Corkill, D. (1989). Cooperative distributed problem solving. In Barr, A., Cohen, P., and Feigenbaum, E., editors, *The Handbook of Artificial Intelligence*, volume 4. Addison Wesley.

[Excelente-Toledo et al., 2001] Excelente-Toledo, C. B., Bourne, R. A., and Jennings, N. R. (2001). Reasoning about commitments and penalties for coordination between autonomous agents. In Müller, J. P., Andre, E., Sen, S., and Frasson, C., editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 131–138, Montreal, Canada. ACM Press.

[Finin et al., 1994] Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1994). KQML as an Agent Communication Language. In Adam, N., Bhargava, B., and Yesha, Y., editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA. ACM Press.

[FIPA, 2001] FIPA (2001). The FIPA Contract Net interaction specification. `http://www.fipa.org/specs/fipa00029/XC00029F.html`.

[FIPA, 2002] FIPA (2002). The foundation for intelligent physical agents.

[FIPA-OS, 2002] FIPA-OS (2002). `http://fipa-os.sourceforge.net/`.

[Fischer, 1999a] Fischer, K. (1999a). Agent-based design of holonic manufacturing systems. *Journal of Robotics and Autonomous Systems*, 27:3–13.

[Fischer, 1999b] Fischer, K. (1999b). Holonic multiagent systems — theory and applications. In *Proceedings of the 9th Portuguese Conference on Progress in Artificial Intelligence (EPIA-99)*, pages 34–48.

[Fischer et al., 1995] Fischer, K., Müller, J. P., Pischel, M., and Schier, D. (1995). A model for cooperative transportation scheduling. In *Proceedings of the First International Conference on Multiagent Systems*, pages 109–116, Menlo park, California. AAAI Press / MIT Press.

[Fonseca et al., 2001] Fonseca, S. P., Griss, M. L., and Letsinger, R. (2001). Agent Behavior Architectures - A MAS Framework Comparison. Technical Report HPL-2001-332, Hewlett Packard Labs.

[Freichel, 1992] Freichel, S. L. K. (1992). *Organisation von Logistikservice-Netzwerken*. Erich Schmidt Verlag.

[Garrido and Sycara, 1996] Garrido, L. and Sycara, K. (1996). Multi-agent meeting scheduling: preliminary results. In *1996 International Conference on Multi-Agent Systems (ICMAS '96)*, pages 95 – 102.

[Gasser, 1991] Gasser, L. (1991). Social conceptions of knowledge and action: DAI foundations and open systems semantics. *Artificial Intelligence*, 47:107–138.

[Genesereth and Ketchpel, 1997] Genesereth, M. R. and Ketchpel, S. P. (1997). Software agents. *Communications of the ACM*, 37(7).

[Gerber et al., 2001] Gerber, A., Klusch, M., Ruß, C., and Zinnikus, I. (2001). Holonic Agents for the Simulation of Supply Webs. In *Proceedings of the 5th International Conference on Autonomous Agents*.

[Gouldner, 1961] Gouldner, A. W. (1961). The norm of reciprocity. *American Sociological Review*, 25:161–179.

[Guide, 2001] Guide (2001). FIPA-OS Developers Guide. `fipa-os.sourceforge.net/docs/Developers_Guide.pdf`.

[Guttman and Maes, 1998] Guttman, R. H. and Maes, P. (1998). Cooperative vs. competitive multi-agent negotiations in retail electronic commerce. In *Proceedings of the Second International Workshop on Cooperative Information Agents (CIA'98)*, Paris, France.

[Hales and Barker, 2000] Hales, K. and Barker, J. (2000). Searching for the virtual enterprise. Working Paper, http://www.it.bond.edu.au/publications/.

[JADE, 2002] JADE (2002). `http://sharon.cselt.it/projects/jade/`.

[Jarillo, 1988] Jarillo, J. C. (1988). On strategic networks. *Strategic Management Journal*, 9:31–41.

[Jennings, 1999] Jennings, N. (1999). Agent-based computing: Promise and perils. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1429–1436.

[Jennings et al., 1998] Jennings, N., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems Journal*, 1(1):7–38.

[JESS, 2002] JESS (2002). `http://http://herzberg.ca.sandia.gov/jess/`.

[Kemmner and Gillessen, 2000] Kemmner, G. and Gillessen, A. (2000). *Virtuelle Unternehmen*. Physica-Verlag.

[Klir, 1991] Klir, G. (1991). *Facets of Systems Science*. Plenum Press.

[Knabe et al., 2002] Knabe, T., Schillo, M., and Fischer, K. (2002). Improvements to the FIPA contract net protocol for performance increase and cascading applications. In *International Workshop for Multi-Agent Interoperability at the German Conference on AI (KI-2002)*, Aachen, Germany.

[Koestler, 1967] Koestler, A. (1967). *The Ghost in the Machine*. Hutchinson & Co, London.

[Liu and Layland, 1973] Liu, C. and Layland, J. (1973). Scheduling algorithms for multi-programming in a hard real-time environment. *Journal of the Association of Computer Machinery*, 20(3):46–61.

[Malsch, 2001] Malsch, T. (2001). Naming the unnamable: Socionics or the sociological turn of/to distributed artificial intelligence. *Autonomous Agents and Multi-Agent Systems (AAMAS)*, 4:155–186.

[March and Simon, 1958] March, J. G. and Simon, H. A. (1958). *Organizations*. Wiley, New York.

[Mayo, 1945] Mayo, E. (1945). *The Social Problems of an Industrial Civilization*. Harvard Graduate School of Business, Boston.

[Metzger et al., 2002] Metzger, J., Schillo, M., and Fischer, K. (2002). A multiagent based peer-to-peer network in java for distributed, efficient spam filtering. In *International Workshop on Scientific Engineering of Distributed Java Applications (FIDJ 2002)*, page submitted, Luxembourg.

[Müller and Pischel, 1993] Müller, J. and Pischel, M. (1993). The agent architecture interrap: Concept and application.

[Odell, 2000a] Odell, J. (2000a). Agents (part 1): Technology and usage. Technical report, Cutter Consortium.

[Odell, 2000b] Odell, J. (2000b). Agents (part 2): Complex systems. Technical report, Cutter Consortium.

[Ouchi, 1980] Ouchi, W. G. (1980). Markets, bureaucracies, and clans. *Administrative Science Quarterly*, 25.

[Panzarasa and Jennings, 2001] Panzarasa, P. and Jennings, N. (2001). The organisation of sociality: A manifesto for a new science of multiagent systems. In *Proceedings of the Tenth European Workshop on Multi-Agent Systems (MAAMAW01)*, Annecy, France.

[Parunak, 1997] Parunak, H. V. D. (1997). Go to the ant: Engineering principles from natural agent systems. *Annals of Operations Research*, 75:69–101.

[Parunak and Brueckner, 2002] Parunak, H. V. D. and Brueckner, S. (2002). Cox: Defining what agents do together. AAMAS 2002 Workshop on Teamwork and Coalition Formation.

[Parunak, 1995] Parunak, V. (1995). Manufacturing experience with the contract net. In Huhns, M., editor, *Distributed Artificial Intelligence*, pages 285–310. Pitman, London.

[Poslad et al., 2000] Poslad, S., Buckle, P., and Hadingham, R. (2000). The fipaos agent platform: Open source for open standards. In *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, pages 355–368.

[RoboCup, 2002] RoboCup (2002). `http://www.robocup.org`.

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

[Sahal, 1979] Sahal, D. (1979). A unified theory of self-organization. *Journal of Cybernetics*, 9:127–142.

[Sandholm, 1996] Sandholm, T. (1996). *Negotiation among self-interested computationally limited agents*. PhD thesis, University of Massachusetts, Amherst.

[Sandholm and Lesser, 1996] Sandholm, T. W. and Lesser, V. R. (1996). Advantages of a leveled commitment contracting protocol. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR.

[Schillo et al., 2002] Schillo, M., Kray, C., and Fischer, K. (2002). The Eager Bidder Problem: A Fundamental Problem of DAI and Selected Solutions. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems.*

[Schillo et al., 2001a] Schillo, M., Zinnikus, I., and Fischer, K. (2001a). Towards a theory of flexible holons: Modelling institutions for making multi-agent systems robust. *2nd Workshop on Norms and Institutions in MAS at Agents 2001.*

[Schillo et al., 2001b] Schillo, M., Zinnikus, I., and Fischer, K. (2001b). Towards a theory of flexible holons: Modelling institutions for making multi-agent systems robust. *2nd Workshop on Norms and Institutions in MAS.*

[Searle, 1969] Searle, J. (1969). *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press.

[Sen and Durfee, 1994] Sen, S. and Durfee, E. H. (1994). On the design of an adaptive meeting scheduler. In *Proc. of the Tenth IEEE Conference on AI Applications*, pages 40–46.

[Shen et al., 1998] Shen, W., Xue, D., and Norrie, D. H. (1998). An agent-based manufacturing enterprise infrastructure for distributed integrated intelligent manufacturing systems. In Nwana, H. S. and Ndumu, D. T., editors, *Proceedings of the 3rd International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-98)*, pages 533–548, London, UK.

[Shoham, 1993] Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60:51–92.

[Singh, 1997] Singh, M. P. (1997). Commitments among autonomous agents in information-rich environments. In *Modelling Autonomous Agents in a Multi-Agent World*, pages 141–155.

[Smith, 1980] Smith, R. (1980). The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113.

[Suda, 1989] Suda (1989). Future factory system formulated in Japan. *TECHNO JAPAN*, 22.

[Suda, 1990] Suda (1990). Future factory system formulated in Japan (2). *TECHNO JAPAN*, 23.

[Turner and Jennings, 2000] Turner, P. J. and Jennings, N. R. (2000). Improving the scalability of multi-agent systems. In *Proceedings of the first International Workshop on Infrastructure for Scalable Multi-Agent Systems.*

[Ulieru et al., 2001] Ulieru, M., Walker, S., and Brennan, B. (2001). Holonic enterprise as a collaborative information ecosystem. In *Proceedings of the Workshop on Holons: Autonomous and Cooperating Agents for Industry, Autonomous Agents 2001.*

[van de Ven, 2000] van de Ven, J. (2000). The economics of the gift. `http://ideas.uqam.ca/ideas/data/Papers/dgrkubcen200068.html`.

[Walsh, 2001] Walsh, T. (2001). Empirical Methods in CS and AI. Tutorial held at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 01).

[Weiß, 1999a] Weiß, G., editor (1999a). *Multiagent Systems*, chapter Prologue, pages 1–23. MIT Press.

[Weiß, 1999b] Weiß, G., editor (1999b). *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence.* The MIT Press, Cambridge, MA.

[Williamson, 1975] Williamson, O. E. (1975). *Markets and Hierarchies: Analysis and Antitrust Implications.* Free Press, New York.

[Wooldridge and Jennings, 1994] Wooldridge, M. and Jennings, N. R. (1994). Intelligent agents: Theory and practice. `http://www.doc.mmu.ac.uk/STAFF/mike/ker95/ker95-html.h`.

[ZEUS, 2002] ZEUS (2002). `http://more.btexact.com/projects/agents/zeus/`.