A Language-Action Perspective on the Design of Cooperative Information Agents

Verharen, E.M.

Publication date:
1997

A Language-Action Perspective on the Design of Cooperative Information Agents

Egon M. Verharen

A Language-Action Perspective on the Design of Cooperative Information Agents

A Language-Action Perspective on the Design of Cooperative Information Agents

Proefschrift ter verkrijging van de graad van
doctor aan de Katholieke Universiteit Brabant,
op gezag van de rector magnificus, prof. dr. L.F.W. de Klerk,
in het openbaar te verdedigen ten overstaan van een
door het college van dekanen aangewezen commissie
in de aula van de Universiteit op
vrijdag 14 maart 1997 om 14.15 uur

door

Egon Marc Verharen

geboren op 7 juni 1965 te Abcoude

Promotor: prof.dr.ir. M.P. Papazoglou
Copromotor: dr. H. Weigand

Cover design: Edgar Grimbergen

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Verharen, Egon Marc

A language-action perspective on the design of cooperative information agents /
Egon Marc Verharen. -
Proefschrift Katholieke Universiteit Brabant Tilburg. - Ill.
Met lit. opg. - Met samenvatting in het Nederlands.
ISBN 90-9010286-8
NUGI 855
Trefw.: communication modelling / language-action perspective / intelligent agents.

TABLE OF CONTENTS

PREFACE

Someone once said: "Doing research is a state of mind". For me this prevailed in 1988 while working on my master's thesis project on deductive databases at the research department of Hollandse Signaalapparaten BV. After I realized I wanted to do research, I was offered a research position at the now demised Institute for Language Technology and Artificial Intelligence (ITK) at Tilburg University. Here I had the chance to gain experience working on several projects and was allowed the freedom to work on my own ideas, with as a pinnacle the development of the LEDA (Legislative Design and Advisory) system, a hypertext-based authoring system that supports legislators in developing legislative policy and drafting legislative documents for the Dutch Department of Justice. Although many people urged me to write my thesis on this subject, my interest was (and still is) more with the application of language technology for the development of information systems. This really took off with the arrival of Hans Weigand at the Infolab. From that moment on we started working on the ideas of which the results lie before you.

At the end of 1995 the ITK ceased to exist. I was fortunate enough to be offered an assistant professor position at the Department of Informatics and Accountancy (BIKA) of the Faculty of Economics, under the condition that I finished the thesis as soon as possible (which, by the way, is my favorite deadline). This allowed me to keep on working on the thesis in a stimulating environment, discussing the ideas with fellow researchers, and providing me the infrastructure needed to write this dissertation.

The whole exercise took 'a little longer' than originally planned and hoped for, because of several reasons, that are all part of the life and work of a PhD student, however. Some fun but time-consuming, such as teaching classes, and setting up and maintaining computer facilities in the Infolab. I found out (sometimes the hard way) that Parkinson's Law: "work expands to fill the available volume" does not apply to PhD students; it often exceeds the available volume. Another reason was my style of writing. To show the progress and results of the research, papers have to be published. The final goal for every junior researcher, however, is to publish a dissertation in which all ideas (or at least most) can be presented without page limitations. I took this too literally, and to the shock of my supervisors (prof. Michael Papazoglou and dr. Hans Weigand) produced large piles of paper. I had to learn the hard way that "in der Beschränkung zeigt sich der Meister". Although still weighing in well over 200 pages, I feel the end result is a balanced book, that shows my ideas and state-of-the-art research. Probably the main reason however, was (and for some, is) my unbridled interest in new technological developments. Although such inquisitiveness might be a good quality for a researcher, the more important and needed quality is the ability to go deep into the subject matter and stick with it until new

insights can be gained. Finally, this prevailed with the current research on the Language-Action Perspective and its application in the design of cooperative information agents, and my hope is that when you read this thesis it provides you with such insights.

Of course, this research would not have been possible without the help of many people. And this preface would not be complete without thanking them. First of all, I would like to thank the committee: *prof. Mike Papazoglou*, my promotor, who at a late stage took on the job of supervisor and pointed out interesting details, worked conscientiously through my versions and showed me what writing is all about; *prof. Kees Takkenberg*, for supporting my work, the process guidance and discussions; *prof. Jan Dietz*, for his comments, and for being an indirect motivator with his work on business process design and communication modelling that was a great example for this work; finally special thanks to *dr. Hans Weigand*, my co-promotor, and *dr. Frank Dignum*, for their inspiring thoughts, day-to-day supervision, the discussions, and for being true mentors, showing me what doing research is all about.

I would also like to thank *dr. Peter Braspenning* for introducing me to the world of research, and *prof. Robert Meersman* for allowing me all the freedom to follow my own thoughts and interests. Furthermore, thanks to my colleagues from the Infolab: *Olga De Troyer, Jeroen Hoppenbrouwers, Leo Remijn, Winfried Minnaert, Hennie Daniëls and the other inhabitants of the Lab* for taking over some of my tasks while producing this thesis and the collaborative spirit in the process, and thanks to *Alice Kloosterhuis* for the secretarial support. I would also like to thank the Vakgroep BIKA and Faculty of Economics for giving me the opportunity to finish the job at work. Furthermore, thanks to *Sander Bos*, who took on the job of implementing the ideas and in the process pointed out some 'details' that had to be smoothed. Parts of this research have been carried out within the framework of the LICS (Linguistics in Information and Communication Systems) project. I would like to thank the participants from the Free University and Delft University of Technology for the discussions and being a sounding-board for some of the ideas. Also thanks to the illustrious MMUIS community. Our regular meetings over the years were an adequate and sometimes much needed exhaust-valve and escape from the PhD life.

Finally, I would like to thank my roommate *Peter Flach* for being a motivator, patient listener, and until now always said the right thing at the right moment, and for technical hints in finishing this thesis. Thanks to my family and friends for the patience and I'm all yours again. Thanks to my parents who gave me the opportunity to take up a study and supported me in my choices. Special thanks to *Edgar Grimbergen*, computer artist extraordinaire, for designing the cover which translates my ideas in pictures. And of course, my eternal gratitude to *Monique*, the woman by my side, for the stable home base where everything seemed to just happen by itself, her support in times it didn't seem to work, and just for being with me.

Egon Verharen
Vught, januari 1997

CHAPTER 1
INTRODUCTION

Every research project starts with an idea; something to be investigated or worked out. In this case it was a number of factors. First, the realization that Information Systems today are used in an entirely different way than in the last decades. Second, the important role that communication plays in our everyday life and the emergence of the Language-Action Perspective to describe communication. And third, the next silver bullet in software design: agent technology. The goal is to combine these three in a new approach to Information Systems development.

This chapter provides the reader with an overview of the motivation for this research, its goals, research questions and approach, the scope, application possibilities, and a description of the contributions this thesis makes, and an outline this thesis.

1.1. MOTIVATION AND JUSTIFICATION

This section explains my view on the world of Cooperative Information System development. Triggered by technological developments and insights in organizational theory the role of Information Technology (IT) in general and Information Systems (IS) in particular in supporting the business activities of organizations has shifted. Especially the important role of communication is getting increasingly accepted in both informatics and organizational theory.

Taking this perspective has consequences both for the perception of what an IS is and does, and for the design of such systems (1.1.1). Communication here is not seen as just the exchange of information but as the means to bring about coordination of activities of the communicating parties. This thesis proposes the use of the Language-Action Perspective as a theoretical foundation for looking at business communication and the design of ISs that support the coordination of activities, called Cooperative ISs (CIS) (1.1.2). Although it is possible to implement a CIS in a variety of ways in my view an agent architecture that supports design based on the Language-Action Perspective is important and can guide both the specification and implementation of CISs (1.1.3).

1.1.1. INFORMATION SYSTEMS EVOLUTION

In this section we take a look at the evolution of ISs and their development. Subsection 1 describes how the role of IS within organizations has changed. As a consequence also the modelling methods for developing such systems have changed (subsection 2).

1.1.1.1. THE CHANGING ROLE OF INFORMATION SYSTEMS

Over the years the role that ISs play in supporting the activities of an organization has changed. The schema below shows the changing role of ISs and their typical applications.

processing of data	(batch systems)
integration of data	(DBMS)
exchange of data	(interoperable DBMS)
business process support	(workflow)
collaboration support	(CSCW)
coordination of actions	(CIS, normative system)

table 1. The changing role of IS and typical applications

Traditionally an IS was a means of storing, processing and retrieving data for a specific function. Later, the IS was supposed to integrate different viewpoints on the Universe of Discourse of the organization as a whole. An IS was considered to be one central database and a set of users accessing it through application programs or directly via a query interface (cf. [Date, 1990]). In the mean time, a number of technological and organizational developments has challenged this viewpoint. The most far-reaching has to do with communication. In the technological infrastructure of organizations, the database is no longer a centralized commodity. Nowadays, they are often decentralized and the integrating function has been taken over by the network, the (information) systems are connected to each other and have to be accessed using electronic networks, while still maintaining their autonomy. The purpose of such interoperable databases is to support the exchange of information within one organization or company or between organizations.

Organizational developments had a more profound impact on the role of ISs. Organizational changes force an organization to focus on its business processes. The role of ISs now is to support these business processes, e.g. as is done in workflow applications. In parallel there is a development to support collaboration between people in performing their tasks. The common aspect in these developments is the *coordination of activities.* Recent years have shown a growing interest in cooperative systems (cf. [Power, 1993]). More and more applications, like EDI, Workflow management and collaborative computing or CSCW, are being developed to access different independent resources inside and outside an organization. Since complete integration of the various resources might not be possible for technical or organizational reasons, the applications rely on interaction between the systems. As is described below this can be achieved by communication.

To account for this evolution from central database to communicating IS that supports the coordination of activities in and between organizations the term *Cooperative Information System* (CIS) ([Papazoglou et al., 1992]) is used. [De Michelis et al., 1997] describe in their "Cooperative Information Systems: A Manifesto" that in the literature CIS are being described primarily as "next generation information systems". Taking a past call for papers for the CoopIS conference as an example, the emergence of CIS is described as: "...The paradigm for the next generation of ISs will involve large numbers of ISs distributed over large, complex computer/ communication networks. This paradigm ranges from the vast and visionary Electronic Superhighway, to the large and complex billing system of a telephone company, an even to the small patient IS in a one-doctor office. Such ISs will manage or have access to large amounts of information and computing services. They will support individual or collaborative human work. Computation will be conducted concurrently over the network by software systems that range from conventional to advanced application systems including expert systems, and multiagent planning systems. Information and services will be available in many forms through legacy and new information repositories that support a host of information services. Communication among component systems will be done in a centralized or distributed fashion, using communication protocols that range from conventional ones to those based on distributed AI. We call such next generation IS *cooperative information systems*...." [CoopIS, 1994]

The focus in this thesis is on Cooperative Information Systems. The main role of CIS is the coordination of activities. I view a CIS as a kind of normative system, in which the coordination is governed by making commitments, described by authorizations and obligations of the communicating partners. In line with this development also the modelling methods for the design of these kinds of systems have changed. This will be discussed in the next section. Below the main reasons for the development of ISs: organizational change and the role of communication are described in more detail, followed by a description of a CIS as normative system.

1.1.1.1.1. Organizational change

Rapidly changing products, markets, stronger competition, governmental regulations and deregulations, and technological developments force organizations to change their shape and the way in which they conduct business constantly ([Spewak and Hill, 1993]). To keep up with these changes an organization should be flexible and be able to adapt their activities to new requirements. Organizations, therefore, organize their activities around business processes.

Goldkuhl ([Goldkuhl, 1995]) defines a *business process* as a series of coherent activities that creates a result with some value for an external or internal customer, through solving a problem or task for him. The business process is a meaningful wholeness of value adding activities (value chain), and usually cuts across organizational boundaries within an organization. The process view emphasizes *what is done* in order to satisfy customers. It gives the organization better insight in their processes. The aim of the organization is to make their business processes as effective and efficient as possible. A second change in

doing business is that organizations focus on their core activities and rely on other organizations to provide additional but related functions, thereby forcing cooperation between the organizations. This is also true for large organizations that are split up in autonomous organizational units.

IT is generally seen as an enabler to achieve new forms of organization ([Hammer, 1990], [Scott Morton, 1991]). However, ISs are no longer used as passive data storage systems only, but they structure and support the way organizations do business. Their role is to support the coordination of activities in order to improve the business processes.

The coordination of activities can be done by establishing business rules and standard procedures. Although efficient, it is not very flexible. The introduction of workflows makes it more flexible, however they are usually aimed at describing the work within one organizational unit only, while today's business practice requires not only the coordination of activities within the organization (e.g. between several unit), but also between organizations. The most flexible and effective way of coordinating activities in a dynamic environment where the parties (organizational units or different organizations) have a strong autonomy is by communication.

1.1.1.1.2. The role of communication

[van Reijswoud, 1996] mentions the work of [Hooff, 1995] that describes that business processes demand effective and efficient business communication. In line with the research of Rice ([Rice, 1987], [Rice and Shook, 1990]) the article investigates the commonly accepted hypothesis that, nowadays, businesses need IT in general, and communication technology in particular, to develop the communication processes and the business processes to their full potential. Figure 1.1 depicts this relation.

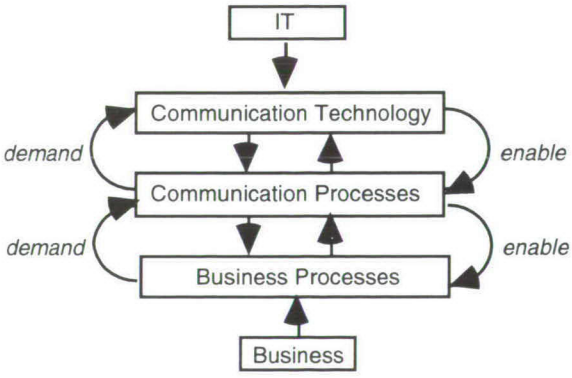


figure 1.1. The relationship between IT and Business (after [Hooff, 1995])

The idea that communication is responsible for the coordination of action in an organization is increasingly accepted in informatics and organizational theory. E.g.,

[Luhmann, 1985] and [Bennett, 1991] consider communication the essential, constituting, or defining feature of all forms of social organization. Also the organizational theories of [Galbraith, 1977], [Morgan, 1986] and [Mintzberg, 1989] emphasized the central role of communication.

The traditional informatical perspective on communication (being the common viewpoint in the IS community since the information theory from Langefors [Langefors and Samuelson, 1976]) was the exchange of information between intelligent actors (cf. [Senn, 1978], [Davis and Olson, 1984], [Reynolds, 1988], [Beek and Jager, 1993]). However, there is an increasing awareness that communication forms the backbone of organizations and is the means by which organizations are organized (cf. [Winograd and Flores, 1986], [Medina-Mora et al., 1992], [Dietz, 1992a], [Taylor, 1993]). [Taylor, 1993] explains the link between organization and communication as follows: "At the most elementary level, the purpose of an organization is to coordinate the efforts of people working on a collective task that has been broken down into a set of specialized activities. The sum of these activities represents the capacity of the organization. *Coordination is achieved through communication.* Communication is the glue of the organization. A productive organization, therefore, is a good communication system—one that achieves coordination with the least effort and minimal control cost".

[Gmytrasiewicz and Durfee, 1993] describe a 'meaningful' act of communication as the ability to judge what messages should be sent, to whom, and in what way, given a particular situation. *Communication*, in the definition used here, is not just information exchange, its essence is *to commit the partners in communication (called subjects or linguistic agents) to a course of action so that one can rely on the other.*

As described in section 2, a language-action perspective is taken, which says that by communicating (using language) one is not only exchanging information, but also performs actions that change the state of affairs. This communicative action framework emphasizes that information does not only has a content but also an action aspect, information says something about something, and what is stated, is communicated by someone to someone else. In this framework communication is the means by which mutual understanding is reached, which is a prerequisite of successful coordination. The goal is the coordination of actions through the commitments of the communicating partners to perform some action.

One has to realize that the way people communicate is changed when ISs are introduced in the organization. An IS in the communication perspective can be characterized as a vehicle for human communication ([Goldkuhl, 1993]). People use ISs for sending and receiving messages through the system, which in addition, can be used to store messages, process information, and create and present new messages ([Eriksson, 1996]). The ISs become information and communication systems. Such systems, here referred to as Cooperative ISs, can be seen as normative systems that manage the commitments made through communication in order to improve cooperation and support the coordination of business activities.

1.1.1.1.3. CIS as normative systems

When coordination of business activities is achieved by commitments of communicating actors, we can use an IS to store and manage the commitments resulting from the communicative acts of the actors. A commitment is something one must fulfil. The very nature of commitments implies that one must record them in some way, otherwise they cannot be 'true' commitments. In case one makes a promise and then forgets it, the commitment is broken. When a company makes business commitments there is need for keeping them in order, otherwise they might easily be broken. However, data records are often not viewed that way. E.g., take a credit limit field as a data record. The duration of the limit is part and result of a communicative action performed by a bank clerk. It is not just a description of an objective state of the world, but records a commitment made, and in this way governs future business actions by the bank and its employees.

[Jones and Sergot, 1993] argues that, at the appropriate level of abstraction, law, computer systems, and many other kinds of organizational structure may be viewed as instances of normative systems. They give the following definition of *normative system*: "A normative system refers to any set of interacting agents whose behaviour can usefully be regarded as governed by norms. Norms prescribe how the agents ought to behave, and specify how they are permitted to behave and what their rights are. Agents may be human individuals or collections of human individuals, or computer systems or collections of computer systems. Normative systems include systems of law, abstract models of computer systems, and hybrid systems consisting of human and computer agents in interaction". A normative system is guided by explicit norms and enforces them in order to achieve the rationalization of administration ([Weber, 1956]), improvement efficiency in coordination ([Galbraith, 1973], [Lee, 1985]) and social equity ([Ryu and Lee, 1992]) Norms have the common feature that they regulate behaviour, e.g., organizational regulations are the means of structuring and regulating administration.

In order to describe or specify systems in the behaviour of which norms play a role normative concepts such as the *authorizations* (permissions and prohibitions) and *obligations* of communicating partners are needed. An obligation is the result of a commitment to perform a certain act and authorizations restrain or allow the commitment to and operation of an act (including doing other communicative acts). In a communication situation, the authorization is mainly used to grant permission to the effective operation of a communicative act, i.e. an authorized speech act has direct effect on the obligations, knowledge and belief of the hearer (see section 6.3). Despite the growing interest in the behaviour of normative systems (e.g., see [Meyer and Wieringa, 1993]), little has been said about how norms that govern these systems are established. Many of the obligations and authorizations in normative systems exist as a result of *communication* with users and/or other systems, e.g., an accepted order for a certain product results in the obligation to deliver that product.

Cooperative Information Systems can be seen as normative systems that manage the commitments made through communication. Although not all activities can be fixed once and for all, due to uncertainties in the environment and the freedom of the actors, it is nevertheless a constant organizational effort to explicate commitments (both intra- and

inter-organizational) in order to improve the cooperation and support the coordination of the business activities. CISs are instrumental in this effort, not only by taking over certain routine tasks, but most importantly by explicating, up to the level of formalization, the rules of the communicative actions.

1.1.1.2. CHANGING MODELLING METHODS

In line with the change in role of the IS also the modelling methods used in designing these new systems changed.

In the field of IS-design the dominating perspective has been a "data storage paradigm" [Lyytinen, 1985]. Traditionally only the static data was modelled. The most fundamental activity of system design is seen as the mapping of a Universe of Discourse into abstract symbolic models describing a portion of the world, using a graphical diagram technique such as Entity-Relationship (ER). The application logic (transforming input into output) is modelled using Data Flow Diagrams (DFD). These models also are a means for software design, where the conceptual model is viewed as an early version of a database schema. The need for more flexibility and the shift towards radical modularization of system and system specification led to Object-Oriented modelling methods. Although these approaches have advantages, such as ensuring the reliability of a future system, understanding for non-computer-experts and, when given a formal and precise interpretation, suitability for software specification, it can be critized for presupposing that all information is used for one purpose only, i.e. to *describe* things. This is labelled "the descriptive fallacy" ([Austin, 1962]). The approach is likely to have problems with non-descriptive information, like customer contracts, promises and orders. It is difficult, if not impossible, to understand the meaning of a contract without taking into account how it is used. More generally, the descriptive approach provides little theoretical support in analyzing why certain types of information are needed. However, all these methods have problems modelling the business processes and communicative behaviour of the systems. A theory is needed that describes how and why people use information. Therefore communicative aspects of IS-usage and design need to be stressed more.

1.1.1.2.1. Business process modelling

Business process is a central concept in the field of Business Process Modelling and Reengineering (or Redesign) (BPR, [Davenport, 1993, 1995], [Hammer and Champy, 1993]). In the development of the BPR model concepts were borrowed from organisational science and economics, such as industrial organization economics for strategic management ([Porter, 1980]) like Porter's Value Chain ([Porter, 1985]). But as [Dietz, 1994a], [Davenport and Stoddard, 1994], and [Goldkuhl, 1995] state, current approaches to BPR lack the ability to support communication in the more specific aspects of systems design, creating "more myth than practical methodology" ([Dietz and Mulder, 1996]). As van Reijswoud shows in his thesis ([van Reijswoud, 1996]) "the use of methods for BPR and TQM in which business communication is considered fixed instead of variable results in sub-optimal solutions".

On the other hand methods from computer science were introduced to model the business processes. These design methods (of which many are still used today) are based on the idea of similarity between business and computer software processes. Business processes were seen mechanistically as if they were computer programs ([Taylor, 1993]). These methods therefore are an extension of software engineering principles. This implementation driven approach focuses on the documental and informational level for increasing the efficiency of the current organization only and not on the 'essential' level, describing the core business processes where new facts are created ([Dietz and Mulder, 1996]). As Dietz describes, "One of the rules of thumb [for developing IS] became: automation has to fit precisely in existing organizational procedures and documental flows. This rule is contrary to the approach of business redesign".

In order to improve business a new paradigm for BPR is needed. Since business processes structure the work of people in the organization the social and communication aspect is important. As we will see in section 2 the language-action perspective describes how language and communication can be used for this.

1.1.1.2.2. Communication semantics

The communication aspect has been largely ignored in IS development methods (e.g., it is virtually absent in the [ISO, 1982] report on conceptual modelling methods). The main focus was on information content only. Now, the field of IS engineering and research has switched its focus from data to communication. This has consequences for what is called data semantics, which traditionally was the way the UoD was captured in the form of conceptual models and dealt with the static and dynamic integrity rules of database systems ([ISO, 1984]). However, now a database must be able to communicate as well. Thus data semantics should include semantics of the communicative behaviour. The focal object of data semantics is no longer the fact, or proposition, but the message. The contents of the database become less interesting than the interfaces between systems. For the organization, it is of utmost concern that the various systems inside and outside can cooperate, and that the semantics of this interaction is well-defined. The systems can be of various types and the management is often decentralized, causing their local semantics to be less important. Since interfaces often have to reconcile conflicting viewpoints, and have to be established by different, autonomous, parties there is an increasing need for standards. This is evident on lower levels of communication, for which standards have been developed already, but also applies to the semantic aspect (OSI's 7th layer).

In conventional modelling techniques the idea of information sharing is conceptualized as data flows (e.g., see [Olle et al., 1988]). Communication is considered as a black box that has a certain input and output that is defined as the sharing of information between different actors. However, it is recognized that it is not enough to view data flow and its control when redesigning organizations and developing ISs ([Verharen et al., 1996b]). Or as Goldkuhl states ([Goldkuhl, 1996]): "An 'information factory' metaphor of an organization has its severe restrictions, since —when focusing on data objects and data flows— it tends to neglect human actors and their action and co-action". What is needed is a way to describe this coordination of actions. A good way of doing this is by looking at the communication in organizations.

1.1.2. TAKING A LANGUAGE-ACTION PERSPECTIVE

Recent years have shown a growing awareness that besides organizational, social, and mathematical theories also linguistic theories can be relevant for IS design in general, and CIS design and the modelling of communication patterns in particular. The introduction of the *Language-Action Perspective* (LAP) in the field of ISs by Flores and Ludlow ([Flores and Ludlow, 1980]) has proven to be a new basic paradigm for a new generation of IS design methods and BPR Models ([Dietz, 1994a]). This section describes first the LAP approach in general, and after this is described how taking a language-action perspective influences my approach to business communication modelling and the design of CIS.

1.1.2.1. BASIC ASSUMPTIONS

Much of the work in today's organizations is performed through language. We attend meetings, talk to colleagues, make promises, negotiate with customers, speak in mobile phones, send faxes and email. The core of much work has come to be communication and social interaction. The basic concern of the language-action perspective is to understand why people need to communicate at work for different purposes. In contrast to traditional views of data flow, the language-action perspective emphasizes what people **do** while communicating; how they create a common reality by means of language and how communication brings about a coordination of their activities. Flores and Ludlow propose the following claim as the basis for the language-action perspective ([Flores and Ludlow, 1980]): "Human beings are fundamentally linguistic beings: action happens in language in a world constituted through language".

As [Holm,. 1996] states: "The explosive development of networking and infrastructure enables innovative usage of information technology (IT) to support communication and cooperation in organizations and work settings. This calls for theory and methodology to inform design". The theoretical foundations for the language-action perspective, first proposed by Flores and Ludlow ([Flores and Ludlow, 1980]) and developed in greater detail by Winograd and Flores ([Winograd and Flores, 1986]), have remained relatively stable over the years. A set of methods, techniques and software artefacts have evolved, that may be seen as a kind of "communication paradigm" in the way [Winograd and Flores, 1986] argued for a 'new foundation for design', and are collectively referred to as the *Language-Action Perspective* (LAP).

This new orientation in design is directed towards the development of computer software for organizational communication and action. Organizations are viewed as networks of commitments and undertakings ([Flores et al., 1988]). [Hirscheim, 1985] describes the LAP as one of the views on organizations, that looks at organizations in terms of social action mediated through communication. The focus is on language actions as a means to understand organization. The physical attributes of a given human activity and the rules of the human interaction are framed as linguistic phenomena ([van Reijswoud, 1996]). The analysis of organizations is performed on the basis of the communication that is used by the workers in an organization.

The basic assumptions underlying the LAP are ([Lyytinen, 1985], [Holm, 1996]):

- the primary dimension of human cooperative activity is language. Action is performed through language in a world constituted through language. Natural language sentences correspond to the performance of social acts.
- the meaning of sentences for the actors in a social setting is revealed by describing the kinds of acts that have been performed through their utterance.
- cooperative work is coordinated by the performance of language actions, called speech acts.
- the speech act is the basic unit of communication.
- speech acts obey socially determined rules. Since these rules govern the performance of speech acts, they permit the systematic study of meaning, and more generally, linguistic behaviour.
- the design of IT has a focus on getting things done, whenever work involves communication and coordination among people. The act of doing something, the recurrent patterns of interaction, and the articulation of these are what concerns the designer of information systems.
- whenever a task is being performed for a customer, a generic sequence of speech acts occur. The sequence typically starts with a request from a customer, then the performer makes a promise, etc.
- worker accountability and customer satisfaction are made explicit.

The core of the theoretical foundation of the LAP is formed by Speech Act Theory. On the basis of this philosophy, organizational communication is seen as the exchange of speech acts for the purpose of coordinating organizational activities. Although different theories of speech acts have been proposed, the most dominant is Searle's Speech Act Theory ([Searle, 1969, 1979], described in more detail in chapter 2). Speech Act Theory provides a means to analyze the communication in detail at three levels: content (propositions), intention (illocution) and effect (perlocution). Form (syntaxis) of the communication is of less importance here than 'why' and 'what' is communicated. Speech Act Theory is based on the initial work of Austin ([Austin, 1962]) on performative use of language. He described that although one usually thinks of conversations as exchanges of information, in certain kinds of conversation the communications may not only be informative but also performative in that they change the state of affairs and commitment among the parties.

The introduction of the LAP in the area of ISs has grown to a mature line of research, and has become a well-known framework and foundation for modelling and design of computer support for communicative action. The first results of taking a LAP to the development of ISs were laid down in the communication supporting tool the Coordinator ([Winograd and Flores, 1986], [Winograd, 1988], [Flores et al., 1988]). Based on speech act theory and the performative use of language Winograd and Flores developed what is called the 'conversations-for-action' theory. Examples of such conversations-for-action are negotiations, where each of the parties ends by committing to a certain plan. Kensing and

Winograd articulate its relevance for the analysis of cooperative work ([Kensing and Winograd, 1991]): "Cooperative work is coordinated by the performance of language actions, in which parties become mutually committed to the performance of future actions and in which they make declarations creating social structures in which those acts are generated and interpreted".

CSCW and computer mediated communication systems still are main areas of application for the LAP (e.g., [Gasparotti and Simone, 1990], [Dietz and Widdershoven, 1991], [Agostini et al., 1994], [De Michelis and Grasso, 1994]). The use of speech act theory for the development of ISs has further been explored and discussed extensively. Examples are the CHAOS system ([De Cindio et al., 1986]), the SAMPO method ([Lyytinen, 1985], [Auramäki, 1988], [Auramäki et al., 1988, 1992a, 1992b]) and the Ordit approach ([Blyth et al., 1992]). From the early applications of the LAP in IS development its use has also spread to application in analyst-user communication (e.g., [Janson and Woo, 1992], [Tan, 1993]), workflow management (e.g., [Schäl and Zeller, 1993a,b], [Schäl, 1995]), electronic commerce (e.g., [Covington, 1996]), business process modelling (e.g., [Dietz, 1994a, 1994b], [van Reijswoud and van der Rijst, 1995]) and organizational modelling for the purpose of IS design (e.g., [Holm, 1994, 1996], [Johannesson, 1995], [Goldkuhl, 1995], [Dignum and Weigand, 1995b], [Weigand et al., 1995], [Verharen et al., 1996], [Verharen and Dignum, 1997]).

The more widespread the LAP has become, the more it has also been debated and critized, e.g., see [Bowers and Churcher, 1988], [Dietz and Widdershoven, 1992], [Bowers, 1993], [Schmidt, 1993], [Suchman, 1994], [CSCW, 1995]. Part of this criticism came from disciplines such as the philosophy of language, linguistics, sociology and ethnomethodology, and is purely conceptual, i.e. it is not grounded in specific empirical studies of the LAP in use. I will not go into the critique here in detail, but refer to section 2.1.2 for this.

Much of the critique does not really denounce Speech Act Theory, but Searle's version of it. Especially its adequacy for describing human communication is criticized. The criticism is *not* on the use of speech act theory for the design of IS as we are interested in here. In this thesis Speech Act Theory is not used as a descriptive framework, but instead its elements are used to model communication for the design of automated systems. Section 2.1.2 describes how Speech Act Theory is adapted to be used for this purpose.

However, since the interest is with supporting the coordination of activities based on the mutual commitment of the communicating partners that arises from performing communicative actions the critique of Habermas must be taken into account ([Habermas, 1981]). Habermas makes a distinction between *communicative actions* that are directed towards mutual understanding of the communicating partners, and *strategic action* where people strive after their own goals. He also criticized the one-sidedness of Speech Act Theory in only paying attention to the role of the speaker and not so much the role of the addressee. Furthermore, he makes a distinction into three reference worlds: *objective world* (referential), *subjective world* (individual), and *social world* (social). These worlds describe the different intentions of people when communicating. In a social situation these can be questioned and may be negotiated. In section 2.1.1.4 an overview of the work of Habermas is presented, and in the next subsection the consequences for the approach taken are described.

1.1.2.2. THE APPLICATION TO BUSINESS PROCESS MODELLING

Here some of the important modelling decisions that are made when following a LAP are described and the consequences this has for the design of business communication and supporting IS.

Inspired by the LAP, methods like DEMO ([Dietz, 1992a, 1994a,b]), Action Workflow ([Medina-Mora et al., 1992]) and BAT ([Goldkuhl, 1995]) focus on communication processes to understand the business, instead of on the current organisational structures or documental flows. The LAP argues that the core of an organization is the network of discrete recurrent communicational actions (or conversational transactions). Taking this approach I also aim at capturing the essence of organizations by analysing the business processes as discrete recurrent (communicative) transactions.

The generic model for business transactions (based on Goldkuhl's BAT (chapter 3) and the communicative action theory of Habermas (section 2.1.1.4) that are adopted in this thesis provide an ontology which serves as a foundation for design, specifying the mechanisms which have to be supported. Instead of only focusing on the exchange of speech acts the focus is on rational coordination of action leading to mutual agreement by negotiation about the conditions for it. This approach is described by [Hirschheim, 1985] as the transactional view on organizations. It perceives organizations as arenas of information exchanges which operate on the basis of contracts. Organizations are viewed as stable networks of transactions which are regulated through processes of coordination and control, by a set of contracts.

IS development is considered to be development of action and communication in the organization. It is also the development of business relations. Taking a LAP has consequences for the way to delimit business processes in the development of them and their supporting ISs. The supplier has its business process, and this process is interlinked with the business processes of the customers. When describing a business process one should focus on the initiation of them through offers and orders and the termination of them through fulfilment of commitments. One should also take into account the achievements of customer *and* supplier satisfaction.

In this thesis the focus is on a theory for the design of (automated) ISs as instrument for organizational change. As in BPR the processes are not viewed to be rigid and the organization to be a strict bureaucracy where the actors' behaviour is predetermined and predictable. Taking a LAP does not allow for the design of *all* organizational communication, but is applicable in situations where routinization and control of a set of activities is desirable, such as an ordering procedure in organizations.

The application of Speech Act Theory here differs from an attempt to describe all kinds of organizational communication in terms of speech acts. Here, some of its basic ideas are applied to organizational communication related to the use of ISs. Furthermore, the focus is on institutionalized behaviour, not on everyday use of natural language. It is in this light that the business logic framework should be seen as a practical building block in IS design and not as ultimate theory about human coordination. By imposing customer

and supplier roles in situations where these labels are not naturally used, I believe (as [Winograd, 1994]) that in BPR organizational effectiveness is increased because of the possibility to describe communication structures by the use of explicitly identified, clear and unambiguous speech acts which lead to a more effective coordination.

Although not all activities in a business situation can be fixed once and for all it should be a constant organizational effort to explicate commitments (both intra- and inter-organizational) in order to *improve* the cooperation and support the coordination of the business activities.

1.1.2.3. LAP AND CONCEPTUAL MODELLING

This subsection describes the different views on the discourse to be modelled, and the formal framework for the modelling techniques.

IS development begins with an understanding of the organization. It is important to understand both the information context (the business processes) and the information content (the information the processes work on). Taking a LAP we not only focus on the information structures but also how they are used in the business processes and the communication within and between organizations to improve the coordination of the processes. To be most useful, ISs must be derived from this base of knowledge about the organization.

The central task in IS design is building a Conceptual Model (CM) of the domain for purpose of problem understanding and system design. It is an explicit description of the domain. Since the CM should describe all relevant aspects, rules, etc., of the specific domain (environment of the organization, or a part of it), taking a LAP two aspects or 'projections' of the CM should be distinguished: on the one hand, the IS will be situated in some organizational environment where it supports the communication and coordination (information context as it was called before). On the other hand the system holds information (data, rules) *about* the domain (the information content). The former is represented in the *Environment of Discourse* (EoD) and the latter in the *Universe of Discourse* (UoD).

The EoD describes the discourse as a process without going into the contents. What is said, and more in particular the meaning of these terms, is described in the UoD. Typical objects in the EoD are the linguistic agents (human or computerized), the message types, and the rules that prescribe and describe the communication (authorizations and obligations). In the UoD typical objects are domain related objects like 'order form', 'product description' and their relationships. Whereas the IS itself is usually not found in the UoD —the IS takes an objective stance— it is the central object in the EoD. Of course, the two domains are closely intertwined, because the discourse going on in the EoD is about the UoD. The EoD is not worked out in traditional IS design methods and is also largely missing from today's popular OO design methods. In chapter 7 the separation between EoD and UoD and the modelling techniques based on it are described in more detail. Figure 1.2 gives a graphical representation of the relationship between EoD and UoD and IS.

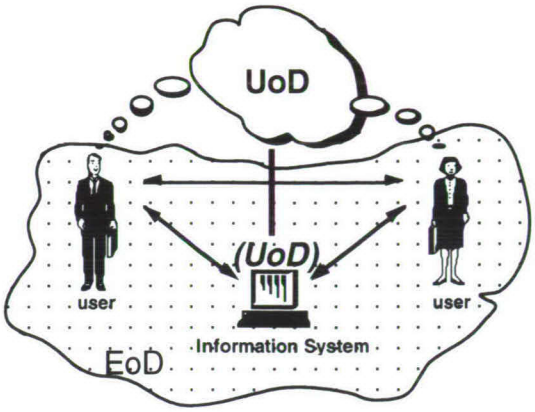


figure 1.2. Environment of Discourse, Universe of Discourse and the IS

The analysis of the domain is important to obtain a better insight into the problem. If the goal is to design an automated IS the analysis models are used as input for a design phase. To be able to verify that the IS will work as anticipated a formal design specification is desired. In the previous section it was argued that taking a LAP to design of CIS can be valuable. The major thrust of this approach is the description of communication models. Although a number of methods have appeared that follow this line of thinking (see section 1.1.2.2 above and section 2.1.3.2) there is not much convergence yet on the formal and logical underpinnings of the models.

Speech Act Theory has been given a formal framework with *illocutionary logic* ([Searle and Vanderveken, 1985]), described in section 2.1.1.3. However, the basic idea behind the theory (and logic) is the description of human communicative behaviour. In the approach taken here the basic concepts of the theory are used to describe the communication between formal systems. Therefore some adaptations are made to the original Speech Act Theory (described in more detail in section 2.1.2.1). This also means an adaptation of the original illocutionary logic. How the changes influence the formal framework is described in detail in chapter 6.

A second observation is that in the LAP the important concepts of authorization and obligation, needed for the successful description of coordination of business activities, have not been worked out. While authorizations are not new either in Speech Act Theory (e.g. the conditions for illocutionary force of Searle, in section 2.1.1.3) or computer science, it is surprising to see that almost none of the methods to IS design for communication support (e.g. DEMO ([Dietz, 1994a,b]), Action Workflow ([Medina-Mora et al., 1992]), SAMPO ([Auramäki et al., 1992a]), and BAT ([Goldkuhl, 1995])) explicitly model the authorizations. Not only the approach presented here explicitly does, also the dynamics of authorizations (and obligations) have been worked out in more detail.

In order to be able to reason about, and manage, and even enforce, authorizations and obligations *deontic logic* is used. Deontic logic is a variant of modal logic which refers to

the logical study of normative use of language in which the statements of "it is obliged...", "it is permitted..." etc. occur. It can be used to describe what it means and what should happen if illegal but possible behaviour occurs. To this end, special modal operators are used that indicate the status of behaviour, i.e., whether it is legal (normative) or not. The fundamental reason for the use of deontic concepts is that coordination of behaviour always requires some form of agreement and mutual commitment. Under the action-oriented framework of communication, it is vital that obligations and authorizations of service providers and customers are modelled and enforced for cooperation. In this way it is possible to reason over any uncooperative action rather than classify them as true failure (leading to unnecessary rollback or recovery actions). Essential here is that deontic logic is valuable whenever it is necessary to make explicit, and then reason about, the distinction between what *ideally* is the case on the one hand, and what *actually* is the case on the other.

This thesis claims that the combination of *illocutionary logic* (the logical formalization of speech acts theory, describing the types and effects of the messages) and *deontic logic* (the logic to reason about obligations and authorizations) can contribute in modelling the communication processes and resulting norms, and reason about the communication structures. The use of deontic logic also makes it possible to describe the course of action in the cases that the communication protocol fails, which is of particular interest for the specification of flexible transactions in the context of CISs. In chapter 6 the formal framework for the specification technique for business communication modelling and their supporting ISs is discussed.

1.1.3. COOPERATIVE ISS AS INTELLIGENT AGENTS

In this thesis CISs are seen as intelligent agents. This section clarifies my choice for using the agent concept as design metaphor. First is described how a CIS can be seen as an agent, being an active participant in the communication processes. Many LAP researchers object against attributing human properties like communication and commitment powers to automated systems. In their view ISs are only supporting communication between human agents. Subsection 2 contains some of my thoughts on seeing computer systems as communicating agents and their role in directing human behaviour. The third subsection describes how such agents can be specified.

1.1.3.1. COOPERATIVE INFORMATION AGENTS

For ISs to be able to cooperate they must have an intelligent interface that can cope with all types of requests for information from users or other systems. In this light ISs actively maintains its information; it communicates with other systems and reasons about the information it contains. It might decide to search for information it needs by enquiring for it from other ISs, preferably in ways it negotiates with (and lays down in contracts with) those other systems. It can respond intelligently to messages explaining why a request

does not have an answer, propose alternatives, or it can negotiate about which requests it responds to and which ones have no effect. For this purpose the IS should contain a task module that plans the tasks it has to fulfil. Active interoperability of IS in the sense of task-oriented cooperation between intelligent front-ends has led to the development of CISs. I refer to the autonomous, intelligent CIS with tasks and communicating according to contracts as a *Cooperative Information Agent* (CIA).

The interest in CIAs from the (cooperative) IS-, database- and distributed artificial intelligence research communities has led to the organization of the first international workshop on Cooperative Information Agents in Februari 1997, where also the approach described in this thesis is presented ([Verharen and Dignum, 1997]).

As described above I propose to take a Language-Action approach to designing CIAs. Looking ahead to the design methodology (chapter 7) in my eyes the best way to generalize over the use of communicating and cooperating systems, is by modelling the communication structures in the domain in terms of (human) agents and the communication lines between them. Each communication line can be viewed as an authorization, e.g. the authorization to place an order or ask for payment. The automation of a communication model assigns a CIA to each agent (assuming here that the communication is automated totally, which need not be the case of course). Each of these CIAs falls under the responsibility of the respective agent. The original communication structure can then be replaced by communication between the CIAs.

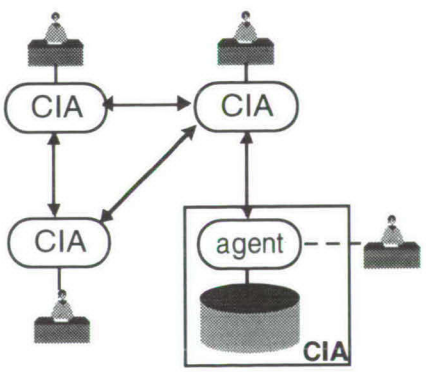


figure 1.3. Communicating agents

It is considered important that the IS contains the one and only correct version of the data, hence reducing redundancy and possible conflicts resulting from that redundancy. As a consequence, the *responsibility* of the system is a company concern, typically delegated to a central person or department. In the case a traditional database exists, and we do not want to reengineer the business processes, it is possible to 'agentify' the database by encapsulating it in a CIA (akin [Shoham, 1993]) thereby integrating it in the agent architecture as described in section 4.2.

In this thesis the term 'agent' is often used both for human users (e.g. supplier and customer in a contracting situation), following the normal usage in linguistically oriented

methods, and for a CIA. From an analysis point of view the agent is a real-world person or company. From a design point of view the agent can also be a piece of software to which the person or organization has delegated some of his tasks. The CIA is to be distinguished from the objects stored in the database. The CIA communicates with other agents and executes actions on its objects on the basis of the incoming messages or on its own initiatives according to the tasks delegated to it.

Recent years have shown a growing interest in intelligent agents. Although there is no consensus in the research community on what an agent is precisely, in this thesis an agent is defined as follows: *An agent is an autonomous computational entity and its behaviour is not predefined but based on commitments to other agents.*

The choice of agents as the solution technology was motivated by the following observations (also described in [Jennings et al., 1996a]):

- the typical application domains, like banking, health care, agriculture, and the example used throughout the thesis of booking a trip involve an inherent distribution of data, problem solving capabilities, and responsibilities;
- the integrity of the existing organisational structure and the autonomy of its subparts needs to be maintained;
- interactions can be fairly sophisticated, including negotiation, information sharing, and coordination;
- the problem solution cannot be entirely prescribed from start to finish, the problem solver needs to be responsive to changes in the environment and to unpredictability in the business process and pro-actively take opportunities when they arise.

Other candidate solutions like distributed object systems have the encapsulation but not the reasoning capabilities required for social interaction and pro-activeness, and distributed processing systems deal with the distributed aspect of the domains but not with the autonomous nature of the components. The set of requirements leaves agent technology as the strongest solution candidate.

An agent is seen as an autonomous entity with certain tasks that communicates with other agents by means of messages. Each agent has certain capabilities, actions that it can perform. These actions can be conventional, such as opening a window, or communicative, such as providing some piece of information. Each agent also has an agenda (or a set of 'obligations', 'things to be done') containing the actions to be performed by the agent, instantly or at some designated point in time. The agenda is not fixed but can be manipulated. New obligations can be added to the agenda, e.g., at the request of another agent. Other obligations stem from the agent's personal tasks. Obligations can also be removed by performing the obligated actions or by violating the obligation. In the latter case the agent usually is obliged to perform some new action that compensates for the violation. The life cycle of the agent consists of processing incoming messages, checking the agenda, and performing the actions due at that time. The architecture of a CIA is described in section 4.2.

1.1.3.2. COMPUTERS AS AGENTS, DIRECTING HUMAN BEHAVIOUR

In the LAP research community, there is a sound aversion against attributing human qualities to machines. It rightly maintains that interpretations and commitments are not made by systems, but by the users and designers. However, there are two reasons why I think the agent conceptualization is applicable: as a metonym and as a metaphor.

By a metonymic use of 'agent' is meant that the computer typically plays a role as "active structured communication medium" ([Winograd and Flores, 1986]), and its 'tasks' and 'commitments' are just shorthand for the tasks and commitments of its users. I agree with [Auramäki and Lyytinen, 1996] that "agreement cannot be coerced by computer support but computers can give spaces for representation where people can compare their views and negotiate, help them to recall commitments and contracts, track the states of commitments, and also remind people of their roles in (routine) workflows". Also for us humans as linguistic agents, 'agent' is a useful design metaphor rooted in our own experience and richer than other design metaphors such as 'object' or 'desktop'.

The same more or less holds for my view of CISs as normative systems and the use of deontic logic. Most of the applications of deontic aspects in computer science involve the prescription of human behaviour in deontic logic. As [Wieringa and Meyer, 1993] describe: "The specification of norms for user behaviour, of organization policies, or organizational behaviour are all applications in which norms applicable to the behaviour of people are specified in deontic logic. If the specification is implemented in a computer, then we have the novel situation that computers may actively deduce permissions, obligations or prohibitions applicable to people".

Although it is not new to let computers direct human affairs, e.g. payroll programs apply law in their computation of tax deductions or the use of traffic lights that regulate traffic, the breadth and complexity increases if we implement deontic logic specifications in computers. Wieringa and Meyer [ibid.] give conditions under which the application of computers to interpret norms becomes *possible* and *practically* feasible. An important principle here is *delegation*. As Wieringa and Meyer say, delegation of authority to a machine is possible, and even necessary, if the machine is to issue instructions to people, as long as the final responsibility for the acts of the delegate (the one delegated the task to) remains with the principal of delegation (the one delegating). As described above the user can delegate some tasks to the CIA, but the responsibility always rests with that user.

1.1.3.3. AGENT SPECIFICATION

Intelligent agents have gained much interest in recent years. Research in agent technology mainly focused on devising formal theories with which (mental) component of agents as intentional systems can be described. Architectures for agent systems that are proposed range from purely theoretical, based on the formal theories, to hardware architectures for robots. The next chapter discusses several approaches.

Most theory and architectures are based on a form of belief, desires and intentions framework. The task of providing a complete model of an agent in terms of mental attitudes is highly complex. It is beyond the scope of this thesis and left for future work.

Instead I focus on reasoning about interactions. For this a model of reasoning about actions is needed. The aim is to describe a formal model and language that captures the required properties of actions and can give formal definitions to useful concepts, such as deadlines. The action logic is combined with deontic logic for specifying and reasoning about obligations and authorizations, and should lead to a clear way of computing with these concepts. The framework proposed here applies a designer-oriented perspective of agent-oriented systems. This approach is similar to for instance [Wagner, 1996] whose vivid agent model is both logical and operational, describing knowledge bases and rule sets for action, commitment and interaction. I attempt to define and formalize agent to agent commitment. Based on the theory, a set of rules is developed which specify under what conditions commitments are formed and maintained, and under what conditions an agent needs to communicate certain facts, to which agent. The basis for the agent-to-agent communication is the theory of communicative action in which agents strive after mutual understanding and agreement. This is in contrast with many agent theories that are based on economical decision science and game theory. Their main concern is rational choice of resources and goals. Both start with the assumption of strategically acting agents that strive after the satisfaction of their own goals.

In designing CIAs a traditional approach to IS specification can be taken, giving DFDs as functional specification, ER Diagrams for database specification and transforming them into a C++ implementation. However, I opt for an architecture of a CIA that is closer to the analysis models used, following the language-action approach, i.e., support for tasks and contracts, and the obligations and authorizations for the agents following from them. (This does not mean however I claim it cannot be done using traditional design techniques and implementation architectures). An architecture for a CIA is described in chapter 4.

1.2. GOALS OF THE THESIS

The starting point is clear: as the role of ISs is rapidly changing from central data repository to cooperation and communication system, new techniques for the design of such systems are needed. I propose to take a LAP to the design of CISs.

This dissertation has four goals. First it aims at contributing to the discussion on the fundamental concepts and logics for communication modelling, thereby gaining further insight in the use of the LAP in IS design in general and CIS design in particular. The thesis tries to build a bridge between two rather discrete worlds: the research on complex interoperable, flexible transactions, where the focus is on the global consistency of distributed systems, and the research on business process models, where the focus is on identifying recurrent patterns of business communication. The goal is to provide useful structuring mechanisms for transactions describing the interaction between CIAs.

In order to be able to describe the meaning of communication models, a logic is needed with which it is possible to reason about the obligations and authorizations that are created during communicative processes. The second goal is to show that formalizing the communication between the agents using a combination of illocutionary and deontic logic provides for a natural and sound framework and integrated semantics for CIAs and their design, as well as for Business Modelling specification.

Third goal of the thesis is to describe conceptual modelling and specification techniques founded in logic for the design of Cooperative Information Agents.

The fourth goal is to provide for a complete and implementable agent architecture to be used in developing Cooperative Information Agents.

1.2.1. RESEARCH QUESTIONS

Having established the goals, the main research question is formulated as follows:

> How can a Language-Action Perspective contribute to the design of Cooperative Information Systems ?

In this thesis the hypothesis is tested that the LAP and the underlying speech act theory can be successfully applied to the design of Cooperative Information Agents by bringing over the research on how people communicate to the field of IS engineering

As derived research objectives I focus on:

- the role of the combination of illocutionary and deontic logic for use of formal specification techniques:

> Can we develop a formal specification technique (and semantics) for the specification of CISs based on illocutionary and deontic logic ?

- an architecture that supports the design of CISs:

> What does an agent-oriented architecture for Cooperative Information Systems look like ?

- and a methodology for designing CISs:

> Can we give a design methodology for Cooperative Information Systems based on the agent architecture and formal specification techniques ?

1.2.2. RESEARCH APPROACH

As little research has been carried out in this field before (but more and more is done right now), I can hardly lean on prior reports, or utilize and test ready at hand theories and hypotheses on the subject. The approach taken in this thesis is design-oriented rather than an empirical research of IS development.

The starting point is the observation of the changing role of ISs, and also the awareness that linguistic theories are relevant for IS design on the one hand. On the other hand there is the interest in the possibilities of agent technology for the development of ISs. It is reasoned that formally defined techniques for the specification of cooperative information agents can be a solution to the problem. Furthermore, the combination of illocutionary and deontic logic can give the formal underpinnings of the techniques.

A method is constructed in which the different techniques are integrated. The applicability of the concepts is demonstrated by working out selected cases and providing a prototypical implementation of a CIA, with which our ideas are tested.

1.3. SCOPE

This thesis describes a multidisciplinary research, combining perspectives from Linguistics, Artificial Intelligence, Logic and Information Systems Engineering. Here a brief overview of the different fields is given.

1.3.1. LANGUAGE ACTION PERSPECTIVE

Recent years have shown a growing awareness that linguistic theories can be relevant for IS design, especially where communication plays an important role. This approach, that has become known as the Language-Action Perspective, has proven to be a new basic paradigm for IS design in general and CISs in particular. Most of the work within the LAP is based on the Speech Acts theory as developed by Austin [Austin, 1962] and Searle [Searle, 1969], which starts from the assumption that the minimal unit of communication is the performance of kinds of language acts, such as requests and promises. Illocutionary logic [Searle and Vanderveken, 1985] is a logical formalisation of the speech acts theory and is used to formally describe the communication structure, i.e., the types and effects of the messages. Since then there has been commented upon Searle's approach both from the linguistic and philosophical sciences, of which the best known is Habermas.

This thesis hopes to contribute to the discussion of the formal underpinnings of the LAP by combining illocutionary logic and deontic logic and also show the applicability of linguistic theories like speech act theory for the design of CIAs.

1.3.2. ARTIFICIAL INTELLIGENCE

The need for more intelligence in computer systems, e.g., to assist the user in his work, or to make the system be able to adapt to its environment, has led to a strong impulse for research in Artificial intelligence (AI). AI can be described as "the field that aims to automate human cognitive abilities, in order to improve the usefulness of computers"[1] [Flach, 1995]. Or, from an engineering point of view, the design and implementation of algorithms and software that exhibits intelligent behaviour. Much AI research today focuses on techniques and formalisms for describing knowledge and behaviour of intelligent agents. Until the mid 1980s mainstream AI researchers gave relatively little consideration to the issues surrounding agent synthesis. However, since then there has been a flowering of interest in the subject.

I follow [Wooldridge and Jennings, 1994] that define AI as: "the subfield of computer science which aims to construct agents that exhibit aspects of intelligent behaviour". This does not mean however, that all fundamental concepts of agents are fixed and understood. On the contrary, there still is not a single universally accepted definition of what an agent precisely is. The research on agents is threefold. Agent theorists are concerned with the question of what an agent is, especially what cognitive properties an agent has, and how to reason about them. Agent language designers are concerned with designing software

[1] I regard AI as a subfield of computer science rather than cognitive science.

systems for programming and experimenting with agents. Finally, those working on agent architectures are concerned with the problem of devising software (using agent languages) and hardware systems that satisfy the properties specified by agent theorists. The work of the last group has not received as much attention as the other two. Most agent theorists are not concerned with the implementation of their agent properties, and most agent language designers are not concerned with fundamental agent principles and properties. However, in my view, the work on architectures is of key importance to the acceptance of this discipline and its applicability in mainstream computer science in general, and IS engineering in particular. This thesis tries to contribute to this by giving an agent architecture and a formal specification technique for its design.

1.3.3. LOGIC

Logic, as the science concerned with questions regarding the patterns underlying human reasoning, has always been seen as belonging to philosophy. However, since the breakthrough in the formalization of mathematical reasoning, logical investigations have become more technical and less philosophical. Logic today is a technical discipline with the mathematics of deductive logic as its main subject [Flach, 1995]. This technical view on logic also explains the heavy use of logics in AI, e.g., formalisms for representing the properties of agents and reasoning over them.

In classical logics the denotation, or semantic value, of an expression depends solely on the denotations of its sub-expressions. However, this does not hold for logics to describe the use of language. As will be explained in chapter 2 and as Wittgenstein wrote "language is an instrument" and "for a large class of cases ... the meaning of a word is its use in the language" ([Wittgenstein, 1958]). The elements of language —words, sentences, propositions— cannot be treated only as things that represent things that are true or false. Speech act theory speaks about believes, intentions and commitments of speakers and hearers and classical logics are therefore not suitable in their standard form for reasoning about these intentional notions expressed by (elements of) language. One way logicians have overcome this problem is by introducing modal logics, which contain non-truth functional operators. Also other types of logics are devised (often by or on the request of AI researchers looking for a formal framework to define a set of properties describing intelligent behaviour), like non-monotonic logics, higher-order logics and meta-languages.

This thesis is not a philosophical one, nor contributes to the general question regarding the patterns underlying human reasoning. Here I follow the 'modern' use of logic, i.e., logic is used to give a uniform and formal semantics to the concepts and languages used, that can also be used to give the (software) agents elementary reasoning capability or intelligent behaviour. This thesis does describe work in modal logic, specifically deontic logic, and its combination with illocutionary logic.

1.3.4. INFORMATION SYSTEMS ENGINEERING

The research in this thesis is restricted to the class of ISs called Cooperative Information Systems. These systems are seen as the next step in the evolution of IS. The evolution

follows the need for more intelligence in the systems and the changed use of the systems within and especially between organisations. Of course I do not state that there is no use anymore for the other types of ISs, but for an organization to be flexible and support the coordination in and between organizations the new class of ISs is much more appropriate.

For the IS development process I restrict myself to the core stages: (requirements) analysis, design, and implementation. This thesis does not concern itself with project management, or testing and maintenance of the system, although these are important aspects of the development and success of an IS. In the analysis stage conceptual models of the discourse are made. The models are used as input for a design stage in which the architecture of the desired IS is determined and the internal working of the system is specified. Finally, code has to be generated (manually or automagically) to implement the system. A common problem with this process is that different techniques, based on different concepts or, even worse, different paradigms are used in the different stages. For verification purposes ("where does this object come from ?") and the possibility to give automated support (CASE tools) to the development process it is required that the concepts of models and specifications are formally defined. Although formal specification languages exist their use is usually limited to the design stage and is still not yet common practice. The development process can be improved upon by basing the different models on the same formally defined concepts, which is one of the goals of this thesis.

1.4. RELEVANCE

Besides scientific contributions I feel that (parts of) this work also have a practical relevance, especially in the fields of Electronic Commerce (and EDI), and Business Process Modelling (and Redesign).

1.4.1. ELECTRONIC COMMERCE

The notion of electronic or digital commerce is gaining widespread popularity. Much more fundamental improvements to (global) commerce are possible, but are presently being overlooked for lack of adequate formal theories, representations and tools. In both [Kimbrough and Lee, 1996] and [Covington, 1996] the role of language and communication modelling in electronic commerce is discussed, emphasizing the use of speech act theory. They also focus on the role of logic as a means to formalize the relationships in electronic commerce, with special attention to the use of deontic logic, in particular for modelling and formalizing (contractual) commitment. Also [Oliver, 1996] describes the possibilities of intelligent agents in electronic commerce, especially in negotiating contracts.

Although this thesis is not about electronic commerce, the concepts mentioned above (speech act theory, deontic logic and intelligent agents) are. Therefore, in my view the same principles that are developed and used for the design of cooperative information agents are applicable for the design of electronic commerce systems. Even stronger, by filling the knowledge and rule bases of the CIA with electronic commerce knowledge and business rules it becomes an electronic commerce agent. In this way this thesis can be seen as contributing to the development of formal frameworks for electronic commerce.

1.4.2. BUSINESS PROCESS MODELLING

In the IS society Business Process Redesign (BPR) is attracting much interest. The common theme is the effective usage that can be made of IT if the redesign and reengineering of a company's IS is performed together and in harmony with the redesign and reengineering of its business functions ([Dietz, 1994a, 1994b]). Although the notion BPR lacks a commonly accepted definition the claim that organizational change is necessary to maintain flexibility and competitiveness is clear ([Teng et al., 1992]). As in Dietz' DEMO method ([Dietz, 1992a]), the core of this thesis' approach is the modelling of tasks and relevant transactions between actors, independent from IT infrastructure, applications, and organizational structure, abstracting from the implementation or realization of information processes and flows, but focusing on the pragmatic purposes of these processes and flows (the 'core' of the business). A communication model and an information model are described, based on the LAP. These models give a better understanding of the 'core' of the business, which is an essential precondition for BPR. The approach taken here even goes further and provides not only models but also a formal framework for describing the business functions, and an architecture (CIA) for the ISs.

1.5. CONTRIBUTION OF THIS THESIS

Traditional IS development methods do not pay much attention to (design aspects of) communication and coordination. Their basic assumption is that ISs are used only to describe a current state of affairs. When viewing ISs as communication media they fall short. To remedy this I propose to take a Language-Action Perspective and focus on using speech act theory to describe this aspect of business activity. Important concepts in modelling for the class of ISs focused on in this thesis, Cooperative Information Systems (CIS), are the separation of Environment of Discourse and Universe of Discourse, and authorizations and obligations. In order to better describe the communication a three-level framework consisting of (interoperable) transactions, contracts, and tasks is proposed.

Taking a LAP gives us the opportunity to describe what it is that people and systems do while communicating and what it means to commit to the performance of some business activity. A formal framework is provided for the precise description of the concepts, based on the combination of deontic and illocutionary logic, which also serves as a basis for a formal specification technique for CIAs. Where the LAP was developed to better describe what humans do while communicating and support that with automated systems, with this approach I try to bring over the theory of communicative action to the communication of formal systems.

The communication supporting systems are not integrated but rather function autonomously. For this the notion of software agents (autonomous entities that perform activities in compliance with tasks delegated to it by a human user) is introduced. In AI-based agent theories little attention is paid to authorization specification so important for CISs. The agent architecture proposed here is not an architecture based on a theory of mental attitudes but instead based on a theory of interactions of the agents.

The goal is not to provide an agent theory as a metaphor for human activity, but to provide a system design architecture that supports the activities in an organization. The design methodology is based on the agent architecture. Such architecture gives an implementation that follows the specification closer than a traditional implementation.

The three topics "communication", "intelligent agents" and "CIS design" make for an interesting combination that in my view describes one of the future paths IS development is taking. Hence the title of this thesis: "a language-action perspective on the design of cooperative information agents".

1.6. OUTLINE OF THIS THESIS

The thesis contains the parts: Background, CIA Framework, Methodology, and Epilogue.

Background

In the first part the work in the LAP, and agent-oriented paradigm that is relevant for the investigations described in this thesis is reviewed (chapter 2). Speech act theory and its logical formalization illocutionary logic, but also the critique on speech act theory, are discussed. This includes a discussion how speech act theory is adapted when used for the modelling and design of communication between formal systems. The agent-part briefly discusses agent theories and gives specific agent architectures that resemble aspects described in this thesis.

CIA Framework

The second part makes up the core of this thesis. It consists of four chapters. In chapter 3 an overview is given of the business logic framework that describes communication structures in organizations. This chapter also includes the example of booking a business trip that is used throughout this thesis. Chapter 4 first introduces the communication framework consisting of transactions, contract and tasks. After this it discusses the CIA Architecture built up around this framework. The knowledge bases and the interpreter that make up the CIA are described. In the Communication Framework, chapter 5, the major elements Transactions, Contracts and Tasks are discussed in more detail. Also a formal specification language for these elements is worked out. In chapter 6 the logical formalisation of the communication framework based on the combination of dynamic deontic logic and illocutionary logic is given.

Methodology

This part consists of chapter 7 that describes the methodology for analyzing business communication and the design of Cooperative Information Agents.

Epilogue

The final part recapitulates the main achievements and conclusions of the research (chapter 8). Also pointers to improvements and further research are given.

CHAPTER 2

BACKGROUND AND RELATED WORK

It is not wise to do research without taking the bodies of work previously performed into account. Furthermore, many 'innovative' ideas are (at least partly) based on older ideas but often applied to different settings. That is true for this work too. To gain a better understanding of the ideas and concepts introduced in the next part, an overview of basic theories and related research are given. Section 1 describes work done in the Language-Action Perspective (LAP). This includes a discussion how LAP in general and speech act theory in particular can be used (after adaptation) for the design for automated communication supporting ISs. In section 2 the agent-oriented paradigm is discussed. I do not claim to give a complete overview of the subjects. Instead, this chapter concentrates on work that provides some of the foundations upon which this thesis is built.

2.1. THE LANGUAGE-ACTION PERSPECTIVE

In this section, first an overview of Speech Act Theory is given, the theory on which most work in the LAP is based. This is followed by a summary of the criticism on Speech Act Theory and the LAP. The overview ends with a description of several methods for business modelling and IS development rooted in the LAP, that influenced my work.

2.1.1. SPEECH ACT THEORY

The research in speech acts is discussed, starting with Austin who initiated it, followed by a description of the work of Searle, including an informal description of illocutionary logic, the logical formalization of speech act theory. This section ends with a overview of the work of Habermas, whose theory of communicative action is a valuable contribution to the speech act theory.

2.1.1.1. AUSTIN

Austin is widely regarded as the inventor of the speech act concept. He examined performative uses of language, the research into the differences between declarative utterances (constatives) and performative utterances: "The constative utterance ... has the

property of being true or false. The performance utterance, by contrast, can never be either: it has its own special job, it is used to perform an action. To issue such an utterance *is* to perform the action—an action, perhaps, which one scarcely could perform ... in any other way" (Austin in [Searle, 1971]). One of the classical examples of a performative utterance is: "I name this ship the Queen Elizabeth" (as uttered when smashing the bottle against the stem). "It seems clear that to utter the sentence (in, of course, the appropriate circumstances) is not to *describe* my doing ...it is to do it. ... [The performative] indicates that the issuing of the utterance is the performing of an action—it is not normally thought of as just saying something" ([Austin, 1962][1]).

The distinction between utterances which are *sayings* (statements, declarations; constatives) and utterances which are *doings* (e.g. promises, bets, warnings; performatives) was attacked by Austin himself. Both in 'Performatif-Constatif'[2] and [Austin, 1962] he shows that constatives turn out to be language acts as well. Making a statement or giving a description is as much performing a language act as making a promise or giving an order. This realisation led to the theory of **language as action**, or as Austin says: "a new doctrine, both complete and general, of *what one is doing in saying something,* in all the senses of that ambiguous phrase, and of what I call the **speech-act**, not just in this or that aspect abstracting from all the rest, but taken in its totality" [ibid.].

Austin considered how many senses there are in which to say something *is* to do something, or *in* saying something and even *by* saying something we do something. He summarized the different senses of 'the use of language as actions' as follows ([Austin, 1962, p. 109]): "We first distinguished a group of things we do in saying something, which together we summed up by saying we perform a *locutionary act*, which is roughly equivalent to uttering a certain sentence with a certain sense and reference, which again is roughly equivalent to 'meaning' in the traditional sense. Second, we said that we also perform *illocutionary acts* such as informing, ordering, warning, etc., i.e. utterances which have a certain (conventional) force. Third, we may also perform *perlocutionary acts*: what we bring about or achieve *by* saying something, such as convincing, persuading, and even, surprising or misleading". Austin claimed there were over a thousand expressions in English ([ibid., p. 149]) indicating speech acts, such as "assert", "warn", "comment", "order", "request". He classified the speech acts into five categories. However, this was criticized for overlap and too much heterogeneity of categories, ambiguous definitions of classes, and misfit between the classification of verbs and definition of categories ([Searle and Vanderveken, 1985], [Ballmer and Brennenstuhl, 1981]).

Austin took a different approach than most language philosophers at that time in that he emphasized the 'use' of expressions more than the 'truth' of the elements of language.

---

[1] The theory of speech acts is worked out in a series of lectures Austin presented at Oxford between 1952-54 under the title "Words and Deeds" and as the William James Lectures at Harvard University in 1955. After his death these were published as a book: "How to do things with words" which is thought of as the standard reference to Austin's work on speech acts.

[2] In [Searle, 1971] a straightforward English translation by G.J. Warnock of the paper Austin wrote in French and presented at an Anglo-French conference held at Royaumont in March 1958 is reprinted.

Not only questions like 'what is it for something said to be true (or false) ?' but also questions of 'how is it possible that when a speaker stands before a hearer and emits an acoustic blast such remarkable things occur as: the speaker means something; the sounds he emits mean something; the hearer understands what is meant; the speaker asks a question, or gives an order ?' are interesting.

Such questions form the subject matter of the philosophy of language. This (as Searle explains in [Searle, 1969]) has to be distinguished from linguistic philosophy which is primarily the name of a method which attempts to solve philosophical problems by attending to the ordinary use of particular words or other elements in a particular language, analysing the meanings of words, and analysing logical relations between worlds and natural languages. Philosophy of language is trend in philosophy and consists in the attempt to analyse general features of language such as meaning, reference, truth, verification, speech acts and logical necessity ([Searle, 1971])[3].

Without going into the history of the philosophy of language (see [Searle, 1971] 'The Philosophy of Language' for a summary of this) we now concentrate on the work of one of the most influential language philosophers: John R. Searle.

2.1.1.2. SEARLE

Searle, a student of Austin, published many standard works on speech acts, the most famous of which is 'Speech acts: an essay in the philosophy of language' [Searle, 1969]. As Austin, Searle states the hypothesis that "Speaking a language is performing speech acts, acts such as making statements, giving commands, asking questions, making promises, and so on; and more abstractly, acts such as referring and predicating. These acts are in general made possible by and are performed in accordance with certain rules for the use of linguistic elements" ([Searle, 1969]). He states that linguistic communication involves linguistic acts, the unit of which is not the symbol, word or sentence, but rather the production of the word or sentence in the performance of the speech act. Speech acts therefore are considered to be the basic or minimal units of linguistic communication.

Searle distinguished between different kinds of speech acts, which by way of example are described below. Imagine a speaker and a hearer and suppose that in appropriate circumstances the speaker utters one of the following sentences:

1. The airline makes the reservation.
2. Does the airline make the reservation ?
3. Airline, make the reservation.
4. I wished the airline made the reservation.
5. If the airline made the reservation, I will leave today.

In uttering 1 a speaker is making an assertion (or stating a fact), in 2 asking a question, in 3 giving an order, in 4 expressing a wish or desire, and in 5 a hypothetical expression of intention. In the performance of each of these five different acts the speaker performs also

[3] Both should also be distinguished from linguistics, which attempts to describe the actual structures—phonological, syntactical, and semantic— of natural human languages.

other acts which are common to all five: the speaker *refers to* (or mentions or designates) a certain object ('airline'), and he predicates the act of 'making the reservation' (or one of its inflections). In the utterance of all five the reference and predication are the same, though in each case they occur as part of a complete speech act different from any of the other four. The notions of referring and predication can be detached from that of complete speech acts as asserting, questioning (or requesting), commanding, etc. Searle calls the complete speech acts *'illocutionary acts'*. The common content (reference to some object and the predication of the same thing) is called ("for lack of a better word" as Searle says) a *proposition*. He describes this feature by saying that in the utterance of each of 1-5 the speaker expresses the proposition that 'the airline makes the reservation'.

Note that it does not say that the sentence expresses the proposition, but in the utterance of the sentence the speaker expresses a proposition.

In uttering any of the five example sentences a speaker is characteristically performing at least three distinct kinds of acts:

- the uttering of words (morphemes, sentences): performing *utterance acts*
- referring and predicating: performing *propositional acts*
- stating, requesting, commanding, etc.: performing *illocutionary acts*.

Utterance acts consist simply in uttering strings of words. Illocutionary and propositional acts consist in uttering words in sentences in certain contexts, under certain conditions and with certain intentions. Searle distinguishes between the illocutionary act[4] and the propositional content of an illocutionary act. Searle furthermore adds Austin's notion of *perlocutionary act*. Since these acts deal with subsequent effects it is not possible to linguistically determine that an utterance counts as the performance on an perlocutionary act, e.g. convincing or annoying. In the following only speech acts proper are discussed, i.e. illocutionary acts.

Illocutionary acts are characteristically performed in the utterance of sounds or the making of marks. One difference between just uttering sounds and uttering them in the performance of an illocutionary act is that the in the latter case one is said to *mean something* by uttering those sounds. Searle claims that the intended effect of meaning something is that the hearer should know the illocutionary force and propositional content of the utterance, not that he should respond or behave in such and such ways, i.e., not perlocutionary. Said differently, uttering something and meaning it are closely connected with intending to produce certain (illocutionary) effects on the hearer. Understanding the speaker's utterance is closely connected with recognizing his intentions. The bridge between the speaker's side and the hearer's side is provided by their common language (and knowledge of the rules of constructing meaningful sentences thereof).

[4]Searle employs the expression 'illocutionary act' which was first given by Austin, with some misgivings, since he does not accept Austin's distinction between locutionary and illocutionary acts. He shows that no sentence is completely (illocutionary) force neutral, i.e. every sentence has some illocutionary force-potential built into its meaning. Searle states that although the *concepts* of locutionary an illocutionary acts are different, the conceptual difference is not sufficient to establish a distinction between separate classes of acts, and in fact every locutionary act *is* an illocutionary act.

Searle distinguishes between the propositional content and *illocutionary force* showing how the proposition is to be taken. He describes rules for expressing propositions, rules for referring and predication (not discussed here, see [Searle, 1969] for an in-depth analysis), and rules for illocutionary force indicating. These rules can be extracted from a set of conditions that are necessary and sufficient for a speech act to be *successfully* and *non-defectively* performed in the utterance of a sentence. Illocutionary acts, like all human acts, can succeed or fail. E.g., an act of firing can be successful only if the speaker has the institutional power to fire someone by his utterance. Otherwise, it is a complete failure. There are various kinds of defects of illocutionary acts but not all of these are sufficient to vitiate the act in its entirety. Searle calls such an act *defective*. Ideally a speech act is both successful and non-defective. For each illocutionary force conditions can be given to determine if that type of speech act is both successful and non-defective. The conditions ([Searle, 1969], [Searle and Vanderveken, 1985]) are described in more detail below.

Summarizing the theory of speech acts we recognize that the minimal units of human communication are speech acts of a type called illocutionary acts. Examples are statements, questions, declarations, and promises. Whenever a speaker utters a sentence in an appropriate context with certain intentions, he performs one or more illocutionary acts. The meaning of an illocutionary act (in general consisting of a illocutionary force and a propositional content) is a function of the meaning of the sentence and the intended effect of meaning something is that the hearer should know the illocutionary force and propositional content of the utterance.

Important here is to note that the concept of *commitment* is important in Searle's work. When someone performs a speech act he or she commits to what he/she is saying. However, in contrast to the approach presented in this thesis the commitment is speaker-oriented only, and it relates to the success of the speech act.

2.1.1.3. ILLOCUTIONARY LOGIC

Illocutionary logic [Searle and Vanderveken, 1985] is a logical formalisation of the theory of speech acts. Its main objective is to formalize the logical properties of illocutionary forces. The illocution (= illocutionary force) of a speech act is what the contents of that speech act indicates that the speaker intends the hearer to recognize him to be doing in uttering the speech act. Illocutionary logic studies the properties of illocutionary forces (e.g. assertion, promise) without worrying about the various ways these are realized in the syntax of a natural language. No matter whether and how an illocutionary act is performed, it has a certain logical form which determines its conditions of success and relates it to other speech acts. Illocutionary logic tries to characterize that form independently of the various forms of expression that may exist in actual natural languages for the expression of that act.

Here, an (informal) overview of the main concepts of illocutionary logic is given. In section 6.3 a logical formalization of the communication between formal systems is given, describing the differences with the logic described here.

2.1.1.3.1. Illocutionary act, force and point

The basic concept in illocutionary logic is the *illocutionary act*. The illocutionary act consists of three parts:

- propositional contents
- illocutionary context
- illocutionary force

The *propositional contents* of the illocutionary act is the part that expresses what the speech act is about. For instance, the propositional content of the illocutionary act "I promise that I will go to the meeting" is "I will go to the meeting".

The *illocutionary context* indicates the relevant knowledge about the situation in which the speech act is made. This knowledge can be factual, about the place the speech act is performed, but also epistemic, about the intentions and believes of the participants in the speech act. It also includes the speaker and addressee of the speech act themselves. Formally, the context of an illocutionary act consists of five elements:

- speaker
- addressee
- time
- location
- circumstances (world knowledge)

The *illocutionary force* determines (for a large part) the reasons and the goal of the communication. The central element of the illocutionary force is the *illocutionary point*. It indicates the type of effect for which the act is performed (the point or purpose). The illocutionary point describes what it is for the speaker to mean the utterance. The speaker intends to produce a certain illocutionary effect by means of getting the hearer to recognize his intention to produce that effect. E.g., the point of statements and descriptions is to tell people how things are, the point of promises and vows is to commit the speaker to do something (it counts as the undertaking of an obligation to do something), the point of orders and commands is to try to get people to do things, and so on. Five different basic illocutionary points (see 'A taxonomy of illocutionary acts' in [Searle, 1979]) are distinguished[5]:

- assertives
- directives
- commissives
- declarations
- expressives

[5] Although the basic illocutionary types are assert, direct, commit, declare, and express ([Searle, 1979], [Searle and Vanderveken, 1985], [Lehtinen and Lyytinen, 1986]), many more can be distinguished (cf. [Habermas, 1984], [Balmer and Brennenstuhl, 1981]., [Chang and Woo, 1994]). We will not go into other taxonomies of illocutionary types here.

The distinction between the five different basic types is directly related to the 'direction of fit' of speech acts [Austin, 1962b]. Four[6] directions of fit are distinguished:

- *word-to-world*: The propositional content of the speech act has to fit an existing state of affairs in the world.
- *world-to-word*: The world is altered to fit the propositional content of the speech act.
- *double direction*: The world is altered by uttering the speech act conform to the propositional content of the speech act.
- *empty direction*: There is no relation between the propositional content of the speech act and the world. Success of fit is presupposed by the utterance.

An *assertive* speech act has a word-to-world fit. The assertive point is to say how things are. Such an act simply makes a statement about the state of affairs in the world, and commits the speaker to the truth of the expressed proposition. Therefore the propositional contents should conform with the (represent an actual) state of affairs in the world. E.g. "The ticket is on my desk".

Both *directive* speech acts and *commissive* speech acts have a world-to-word fit. They try to change the situation in which they are uttered to fit the propositional content of the speech act. The directives lay the responsibility of this fit with the addressee. The directive point is to try to get the addressee to do things (carry out a course of action represented by the propositional content), e.g. "Book me a flight, please". If successful and non-defective, the hearer commits himself to do it. The commissives lay the responsibility of the fit with the speaker. The commissive point is to commit the speaker to a future course of action, e.g. "I promise to pay you".

Declarations have a double direction of fit. By making a declaration the world is changed according to it, it brings about some new state of affairs of the world. The declarative point is to change the world by saying so, e.g. "Hereby you are fired". If performed by someone with the right authority this causes the addressee to be fired.

An *expressive* speech act has the empty direction of fit. The expressive point is to express the speaker's psychological state, feelings and attitude about the state of affairs. E.g. "I am glad you can make it to the conference".

Besides the illocutionary point, the illocutionary force contains six more elements. These elements are all dependent on the illocutionary point. They either indicate the strength of it or the effect of it in some way.

- degree of strength of the illocutionary point
- mode of achievement
- propositional content conditions
- preparatory conditions
- sincerity conditions
- degree of strength of sincerity conditions

[6] Searle asserts that there are only four directions of fit, with two possible agents for the world-to-word direction of fit (the speaker and the hearer), thus leading to five basic illocutionary points.

The *degree of strength* indicates how strong the direction of fit is made. E.g., one can command "book a hotel" or ask "can you please book a hotel". The first speech act has a stronger illocutionary force than the second one. There are different sources of degrees of strength. E.g., both pleading and ordering are stronger than requesting, but the strength of pleading derives from the intensity of the desire expressed, while the strength of ordering derives from the fact that the speaker uses a position of power or authority over the hearer.

The *mode of achievement* indicates that some conditions must hold for the illocutionary act to be performed in that way. E.g., a command makes use of a position of power or authority of the speaker, while an request does not. To be a successful command the speaker must not only be in a position of authority or power, he must be invoking his authority in issuing the command. Also a statement of a person as a witness testifying has a different effect than when that person makes the same statement somewhere else.

In many cases the illocutionary point forces some *conditions on the propositional content* of the speech act. E.g., if a speaker makes a promise the propositional content must be that the speaker will cause some condition to hold in the future, or the speaker will perform some future course of action. One cannot promise to have done something in the past or that someone else will do something.

There are basically three types of *preparatory conditions*. First, those dependent of the illocutionary point. E.g., requesting someone to do something while it is obvious that he is already doing or is about to do it independently of the request, is pointless and to that extent defective. Also, in promises something, it is presupposed that what is promised is beneficial for the addressee, the addressee wants the speaker to do it, and also that the speaker can fulfil the promise. Secondly, there are preparatory conditions that depend on the propositional content of the speech act. E.g., if I order someone to open a window I presuppose that the window is closed. The third kind of preparatory conditions state that the speaker should be in a position of authority over the hearer.

In the performance of a speech act the speaker presupposes the satisfaction of all the preparatory conditions. This does not imply that preparatory conditions are psychological states of the speaker, rather they are certain sorts of states of affairs that have to obtain in order for the act to be successful and non-defective (see next subsection).

In the performance of an illocutionary act, the speaker expresses some attitudes to the propositional content, e.g., in making a promise one expresses an intention. These attitudes, or psychological states, of the speaker are called *sincerity conditions* ([Searle, 1969]). If the propositional content of a speech act conforms with the actual psychological state of the speaker, we say the act is sincere. But, it is also possible to perform a speech act without actually having the expressed psychological state, called an insincere speech act. E.g., if I state that the sun shines while I know it is raining then the statement is insincere (a lie). Insincere speech acts are defective but not necessarily unsuccessful. The distinction between sincere and insincere acts is that, in the case of a sincere illocution, the speaker intends to do a certain predicated act by expressing something will hold in the future; in the case of an insincere illocution, he does not intend to do that act. Also, in sincere illocutionary acts, the speaker believes it is possible for him to do such act.

The last element of the illocutionary force is the *degree of strength of the sincerity conditions*. Similar to the strength of the illocutionary point, the same psychological state can be expressed with different degrees of strength. E.g., a request expresses the desire that the addressee performs a certain act. However, if the speaker 'begs' a stronger desire is expressed than if he just requests. Often the degree of strength of the illocutionary point and that of the sincerity conditions are related.

2.1.1.3.2. Successful and non-defective illocutionary acts[7]

Another issue of interest is the success of the performance of an illocutionary act. We therefore take to look at the conditions of performing a successful and non-defective illocutionary act. With the help of the definition of illocutionary force the *successfulness* and *non-defectiveness* of an illocutionary act can be described.

A successfully and non-defective performance of a speech act concerns more than just the speaker's intentions. It includes conditions on 'illocutionary uptake' [Austin, 1962] or 'input-output' conditions [Searle, 1969]. 'Output' covers the conditions for intelligible speaking and 'input' covers the conditions of understanding. Examples are that the hearer must be awake and paying attention, the speaker and hearer both know how to speak the language, both are conscious of what they are doing, and have no physical impediments to communication such as deafness. Since these conditions for understanding are of little theoretical interest we concentrate here on the conditions concerning the seven features of illocutionary force, which reduce to four different types of necessary and sufficient conditions for the successful and non-defective performance of an illocutionary act. These are given by example.

A speaker succeeds in issuing a successful and non-defective *command* to the hearer iff:

- the point of his utterance is to attempt to get the hearer to do an act *A* (illocutionary point). This attempt is made by invoking his position of authority over the hearer (mode of achievement), and with a strong degree of strength of illocutionary point (degree of strength).
- he expresses the proposition that the hearer will perform a future act *A* (propositional content condition)
- he presupposes both that he is in a position of authority over the hearer with regard to *A* and the hearer is able to do *A*. He also presupposes all of the propositional presuppositions if there are any. And all his presuppositions, both illocutionary and propositional, in fact obtain (preparatory and propositional presuppositions).
- he expresses and actually has a desire that the hearer do *A* (sincerity condition) with a medium degree of strength (degree of strength).

[7] although this describes only the definition of a successful and nondefective performance of an *elementary* illocutionary act, for reasons of simplicity I will take this as the definition for illocutionary acts in general and do not go into the more complex issue of complex illocutionary acts, see [Searle and Vanderveken, 1985] for this.

In Searle's view there are only two ways a speech act can be successful but defective: if some of the preparatory conditions do not obtain and yet the speech act might still be performed; and if the sincerity conditions not obtain, i.e., the speech act can be successfully performed even though it is insincere. The focus of speech act theory is on the speaker. The success of a speech act depends on the speaker's ability to perform a speech act that should be understandable and successful. The conditions require that the speaker presents a speech act that is valid in the context: he should be sincere, have the authority to perform the speech act, and his proposition should be possible in the context.

2.1.1.4. HABERMAS

Founded in the speech act theory, Habermas proposed an alternative theory of communication, called the Theory of Communicative Action ([Habermas, 1981], English translation: [Habermas, 1984], [Habermas, 1987]) that concerns linguistic coordination of action in social systems. Habermas critized Searle's work concerning three topics discussed below: communicative versus strategic action, validity claims for a speech act, and the classification of speech acts. For a more thorough discussion of the theory of communicative action and its differences with speech act theory see for instance [van Reijswoud, 1996] and [Dietz and Widdershoven, 1991].

Habermas identifies two mutual exclusive mechanisms for coordinating social actions: consensus and influence. His thesis is that consensus is the fundamental mechanism of social coordination. Consensus is connected with the idea of Verständigung. A notoriously difficult word to translate, Verständigung refers both to linguistic understanding and to the process of reaching agreement, extending across a spectrum of meanings ranging from comprehension to consensus ([Cooke, 1994]). English translations of Habermas favour "understanding", presumably because it may also be used to imply both comprehension and agreement. I will use it here too.

Corresponding with the distinction between influence and consensus, Habermas distinguishes between *strategic action* and *communicative action*.

In strategic action each agent has its private goals and plans. Depending on the reaction of other(s) they will try to compromise or strive to 'defeat' them Their motivation is empirical and they try to maximise their profit and minimise their losses. Coordination is based on empirical contingencies, especially on a claim to, or use of *power*.

In communicative action coordination is brought about by a mutual understanding of the situation and goals pursued. Communicative action should be interpreted as an 'interaction between subjects that engage in a social relationship' with the aim of reaching a mutual understanding of the situation in order to be able to coordinate their actions. Their motivation is rational: people respond to requests because they presuppose that these requests can be justified when asked for.

The success of communicative action depends on the hearer's agreement to three validity claims raised: truth, justice, and sincerity. The claim to *truth* entails that the speaker contends to represent the factual contents of the speech act as they are. The claim to *justice* regards the adequacy of the interpersonal relation between speaker and hearer. The claim to *sincerity* entails that the speaker is genuine in performing the speech act.

Validity claims define conditions for the creation of commitments. In principle each of the claims can be questioned. When the hearer disagrees on a claim, the speaker must supply reasons to support the validity of the claim. E.g., a hearer can challenge the truth claim ("what reasons do you have for saying *that* ?"). He can also challenge the speaker's right to say what he says, disagreeing with the legitimacy of the normative context of the utterance (claim of justice), e.g. by saying "what reasons do *you* have for saying that to *me* now ?". Another possibility is that the hearer questions the speaker's sincerity, e.g. "what reasons do you have for expecting me to believe you *mean* that ?" ([Cooke, 1994]).

In the claims three worlds of reference are distinguished: the *objective* , *social* and *subjective world* . The claim to truth refers to the objective world (how things are), the claim to justice refers to the social world (how the participants stand towards each other), the claim to sincerity refers to the subjective world (how the speaker perceives the world).

Habermas' critique concerns mostly Searle's earlier work ([Searle, 1969], [Searle, 1979]), which Searle later improved upon in [Searle and Vanderveken, 1985] where discussion on conditions for speech acts provide a means for dealing with validity claims. The basic critique remains the same however ([Habermas, 1991]): Searle's speech act theory puts in the foreground only the speaker's role in presenting a successful speech act and actually hides the negotiation about the validity claims, by this failing to see the principle which underlies and explains successful communication: the orientation of the participants towards *mutual agreement*.

Another point of critique is the speech acts classification. Habermas criticizes Searle in that he misses the distinction between speech acts based on power claims and those based on validity claims, and also the distinction between speech acts which express a claim to justice (such as promises) and those expressing a claim to sincerity (such as intentions). Habermas presents a classification based on one dimension only, namely the dominant claim put forward by the speaker. I will not discuss Habermas' classification here, see e.g. [Dietz and Widdershoven, 1991], where Searle's and Habermas' taxonomies are compared.

Habermas' distinction between empirical and rational orientation of communication means that direct orders can be distinguished from polite requests which are open to discussion. The distinction between strategic and communicative action is exemplified by the distinction between a claim to power and a validity claim. E.g., the success of an imperativa ("I want you to") is based on a claim of power (e.g. fear of sanctions), and the success of a directive (e.g., command) is based on a validity claim (e.g. prevailing social regulations). In this way only speech acts to which the speaker assigns criticizable validity claims motivate the hearer to accept the speech act offer, and because of this foundation do they become the mechanism for effective coordination of action ([Habermas, 1981]).

However, Habermas has also been criticized for his taxonomy. [Auramäki and Lyytinen, 1996] summarize some critique: "Despite the focus on dyadic relationships through the notion of validity claims Habermas' classification is too narrow in dealing with larger communication contexts. Moreover the classification of directives and commissives into regulativa speech acts is not always reasonable. It seems to be useful to know if we are commanding or promising. Use of power is a part of everyday action, and we do not want to exclude communication based on the use of power (strategic action) from the analysis of communication".

2.1.2. LAP CRITICISM

The Language-Action Perspective has received some critiques from areas like ethnomethodoloy, sociology and linguistics. Most of the criticism however concerned critique on Searle's speech act theory. This section describes some of this criticism.

The criticism the LAP received concerns mainly its claim to give a theoretical foundation for the coordination of actions through the use of language, thereby providing the possibility to reorganize work. Most of the criticism, however, is critique on Speech Act Theory, the basis most of the work in the LAP is built.

Representing and redesigning work is a complex and difficult task. The LAP is frequently related to BPR efforts ([Keen, 1991], [Medina-Mora et al., 1992], [Dietz and Mulder, 1996]). However, not giving enough attention to actual work practices and underestimating the complexity of these practices will lead to defective design. Examples of such failures are found in [Bowers, 1993], [Sachs, 1995], [Suchman, 1995]. As [Holm, 1996] summarizes: "The main argument is that taking a Language-Action Perspective provides a too simple picture of what really goes on in work, with respect to the contingent character of work practice. It imposes a categorization scheme upon a user community, ruling out that community's own scheme. Furthermore it is claimed to be based on a rationalistic view of communication, and work organization. It represents a managerial perspective, with discipline, surveillance, and control of workers on the agenda".

A number of arguments against the LAP that can be found in the literature (e.g., [De Michelis and Grasso, 1994], [Holm, 1996]) can be summarized as:

- the normative use of the illocutionary force (implicit in the idea of categorizing utterances by their illocutionary points and making intentions explicit) is the basis for developing tools for the discipline and control over organization members' actions and not for supporting cooperative work among equals ([Suchman, 1994]);
- the LAP supports the creation of tools usable only in some organizational settings, generally hierarchical, authoritarian organizations ([Robinson, 1991]);
- the LAP does not recognize that embedded in any conversation is a process of negotiating the agreement of meaning ([Robinson, 1991]);
- the LAP is too much oriented towards discourse analysis and not conversation analysis ([Allwood, 1977], [Levinson, 1983]);
- the LAP misses locality and situatedness of conversations, because it proposes a set of fixed models of conversations for any group without supporting the ability to design its own conversation models ([Bowers and Churcher, 1988])
- a taxonomy of speech acts from the illocutionary viewpoint is wrong in that it assumes a one-to-one mapping between utterances and illocutionary acts, which is not recognizable in real life conversations ([Bowers and Churcher, 1988]).

Much of the critique on the LAP is rooted in sociological and ethnomethodological traditions ([Suchman, 1987, 1994], [Lynch, 1995]) or influenced by it ([Bowers, 1988, 1993], [Kaplan et al., 1992], [Robinson, 1991]). As [De Michelis and Grasso, 1994] point out this is, in a certain sense, surprising, for ethnomethodology shares with the proponents of the LAP

the same theoretical grounds: a rejection of the Tayloristic approach to work analysis and design; an attention to the complexity (contextuality) of work processes; and a reference to 20th century European philosophy, in particular to phenomenology and hermeneutics (cf. [Winograd and Flores, 1986]).

The debate on the validity of speech act theory as a basis for computer systems for workflow and communication support reached a high point with the Winograd-Suchman debate ([Suchman, 1994], [Winograd, 1994], [CSCW, 1995]). Although I do not want to repeat that debate here I agree with Winograd who acknowledges Suchman for three observations: i) explicit representation of intentions and commitments are appropriate only in certain social/organizational situations, and applicable to certain work structures; ii) the people 'who live the situations being represented' must participate in the generation of the abstractions; iii) no abstractions will capture all meaning of a complex organizational situation. The categorization system used for making the abstraction of work, imposes a set of constraints on the workers, when they are describing and representing their own work activities. This can cause a dangerous form of blindness (as [Winograd, 1994] admits) but it is inherent in any form of method for structuring organizations, work and ISs.

In the next subsection is described how these remarks have been taken into account when using speech act theory for the design of cooperative ISs.

To resolve some of the differences there are attempts to synthesize the LAP with other approaches, e.g. ethnomethodology ([DeMichelis and Grasso, 1994]), conversation analysis ([Bowers and Churcher, 1988], [Ljungberg and Holm, 1996]). Also alternative communication models have been proposed as a foundation for the design of communication systems, e.g. [Shepherd et al., 1990], [Bowers, 1988, 1993], [McCarty and Monk, 1994], [Schiffrin, 1994]. Also attempts to enrich the LAP by including ideas from the Scandinavian tradition of participatory design have been made ([Kensing and Winograd, 1991]).

2.1.2.1. ADEQUACY OF SPEECH ACT THEORY FOR THE DESIGN OF IS

Much of the speech act theory criticism, given by linguists and language philosophers, concerns the question whether speech act theory is an adequate theory for the description of human communicative behaviour. However, here the task is not to search for an ultimate "true" philosophy or linguistic theory about communication or social interaction. Instead the focus is on what needs to be articulated about communication (and work) to improve current praxis in the development and usage of ISs. Where linguistic and philosophical theories are descriptive by nature, design is prescriptive. The basic premise here is that philosophers and also linguists are *passive* observers, describing social interaction, while IS developers are *active* designers of such interaction. In defence of using a LAP and speech act theory I follow [Holm, 1996] in that one must consider that we now are using both in a new context, i.e. that of designing ISs.

Since this work is based for a large part on speech act theory I must address the adequacy of using speech act theory (and that of the LAP in general) for the design of ISs. A *pure* communicative view may have shortcomings. However, a communicative oriented

view may be rewarding, since a large part of work is performed through language, and ISs are used to support communicative activities. Speech act theory has played an important role in the context of IS design. However, it appears that it needs to be used with caution and to be adapted for an IS context. The adaptations are described below.

[Ljungberg and Holm, 1996] and [Holm, 1996] give an extensive description of aspects of the shortcomings of speech act theory for the design of IS. This subsection follows their framework discussing the applicability and adequacy of speech act theory for designing IS. With every aspect the choices I have made or adaptations that are described in this thesis are given. The main points to address are: When ISs are viewed as mediator and support for communication, how should we apply communicative theories in modelling and design ? When is it appropriate to apply a speech act perspective and when not ?

2.1.2.1.1. How to apply speech act theory in modelling and design ?

This subsection lists aspects that should be taken into account when applying speech act theory in modelling and design of ISs for the support of communication in organizations. With every aspect the solutions chosen in this thesis are described.

Articulation of work

The situated character of work is discussed in the theory of articulation of work ([Schmidt, 1993]) developed to handle the fact that cooperating actors have to articulate (allocate, coordinate, schedule) who is doing what, where, how, by means of what, and under which constraints. Dimensions of articulation include actors, responsibilities, tasks, activities, conceptual structures, and information-, material-, and technical resources. It goes beyond a communicative approach and has a broader scope than speech act theory.

There is no real conflict between the two if the latter is viewed as one way in which communicative aspects of work can be articulated. However, I feel the other aspects are important too, and, as is shown in chapter 4 and 7 attention is paid to these aspects, in particularly tasks, responsibilities, activities, and conceptual structures.

Discourse vs. Conversation

Discourse analysis and conversation analysis are two different approaches to the study of language usage. When speech act theory is adopted for IS design the terms 'discourse' and 'conversation' are often used as synonyms. A discourse is viewed as a generic goal-oriented task (cf. [Flores et al., 1988], [Auramäki et al., 1988]). It is a globally managed sequence of speech acts forming a coherent and predetermined course of action leading to a goal. Conversation analysis states that conversational sequences are rarely structured. Instead, certain kinds of utterances go together in pairs, like question-answer, and offer-acceptance ([Levinson, 1993]). In many situations the natural response to a request is complying with it (or rejecting it) without any explicit promising in between.

While the course of action in a discourse is globally managed by means of constituting rules, that in conversations is locally managed by the participants, i.e., "who talks and what gets talked about is decided then and there, by the participants in the conversation,

through their collaborative construction of the conversation course" [Suchman, 1987]. An office procedure can be anywhere between strict globally controlled discourse and free unrestricted conversation. The contact with a customer may follow a predefined format (generic discourse) or there may be room for creativity and improvisation. In designing IS support for contracting and negotiation we need to handle both situations.

Here I follow a business logic based on the Business as Action game Theory ([Goldkuhl, 1995]) which describes a communicative action model of business relations and processes. In the framework (chapter 3) communicative actions between suppliers and customers can be set up with the goal of establishing a business relation. However, it is not a rigid framework in which all steps have to be taken. Instead, it should be regarded as an editable reference model in a design situation. The approach presented here is discourse oriented and I view communicative transactions (logical groupings of speech acts with constraints between them and a predefined goal) as the basic unit. Transactions are grouped together in contracts, that describe the communicative behaviour between two agents, and tasks that describe how the agent can reach his goals by performing private actions and initiating communicative transactions. Transactions, contracts and tasks and the relationships between them are described in section 4.1 and chapter 5.

Multi-functionality of speech acts

Searle's classification has been critized for not taking into account the multi-functionality of communicative acts. In the conversation-for-action schema a one-to-one mapping between specific messages and illocutionary acts is used. A message will either count as a request, or a promise, or a declaration etc. However, the one-to-one mapping is designed, it is decided beforehand how the actions should be interpreted. Again, there is the difference between describing and designing interaction. The discussion about multi-functionality of speech acts now turns into the question of what specific classification of speech acts is needed in a design situation.

Here I adopt the Searlian classification, however with one adaptation. Based on Habermas' ideas a claim (describing the relationship between speaker and hearer) is attached to the illocutionary acts. I distinguish between power, authorization and charity. The speech acts and claims are described in detail in chapter 6.

Context

Speech act theory is also critized for its limited possibilities of referring to the wider social context in which the conversation is embedded. The focus is on the performer of an idealized utterance. It has a sender perspective rather than a receiver or social-interactional perspective. The notion of context may be quite complex.

In this thesis the focus is on the supplier-customer conversation in the context of business transactions. The business logic framework mentioned above is symmetric in the way that both the supplier and customer role are emphasized. See chapter 3 for details.

Social roles

Our design view should also answer questions on social roles in organizations. According to [Flores et al., 1988] a typical office comprises a structure of recurrent conversation

patterns associated with formally declared roles, such as group manager, assistant etc. The roles and power relations among the users are assumed to be stable. This view leads to a notion of organizations as bureaucracies and away from the powerful view of organizations as networks of commitment (also put forward by [Flores et al., 1988]). The question is whether one should design for stable structures or change. If language is considered to be social, inter-subjective and a means through which a social reality is created, a language oriented view on design should have a dynamic concept of roles.

The business logic framework rests upon the assumptions about the social roles of supplier and customer, however some freedom is allowed by the use of the authorization concept with which new possibilities for communicative action can be created dynamically. This is described formally in section 6.3.2.

Cognitivism and individualism

A classical problem within philosophy of language is the relation between the private and socially public world. Beliefs and intentions belong to the private realm, conventions belong to the social. In speech act theory both intentions and social conventions play crucial roles, but the focus is on the intentions of the speaker. Although in philosophy of language it is often claimed that we should only refer to public items, many people find it counter-intuitive not to take intentionality into account. The question is whether appealing to intentions is relevant when using speech act theory in the design of IS.

[Winograd and Flores, 1986] state that social and conventional aspects of communication are more relevant to consider in designing ISs for organizational communication, and I agree. However, I feel intentions cannot be left out. This is conform the currently popular cognitivistic trend in AI. Agent design gives the possibility to include intentions. In contrast with AI-based approaches the focus is not on a theory of intentionality, leading to a narrow individualistic view, but see agents as social cooperating communicative entities.

Organizational agents

In IS design it is relevant to consider organizations as responsible agents. A promise may create a commitment for an organization or a department, not (only) for individuals performing speech acts. E.g., when a person acting "on behalf of" the organization makes a promise (or accepts a customer order) a commitment is made for the whole organization. In Action-Workflow, e.g., sub-commitments are created within the organization from one department to another. SAMPO ([Auramäki et al., 1988, 1992]) illustrates the relationship between commitments in the notion 'coordination of commitments'.

I agree with [Ljungberg and Holm, 1996] that this aspect needs further elaboration. In section 1.1.3.2 (delegation of authority) and section 6.3.2 (delegation of obligation and relations between contracts) some thoughts and examples are presented.

Propositional content

In the conversation-for-action approach the information content of speech acts is ignored. The schema focuses on *who* is communicating *when*, and not on *what* is communicated. However, in speech act theory propositional content plays a crucial role. The separation of the concern for information content and information context is often imposed by

modelling-administrative problems, e.g., redundancy. [Holm, 1996] points out that any approach combining models of information content with models of information context demands a non-conventional formalization of the used models.

I follow Searle in that the propositional content is important. The separation of content and context is accounted for by the distinction between, respectively, UoD and EoD (sections 1.1.2.3 and 7.2). Modelling techniques applying this are described in chapter 7, while the formal framework underlying the modelling concepts is given in chapter 6.

Conclusion

In the discussion about the applicability of speech act theory and LAP as a foundation for IS design it is important to be aware of the adaptations that are made to the original theory when applied in this field. The adaptations I propose deal with the formal description of obligations and authorization, and the different claims made on basis of the relationship between agents. Furthermore a new way of structuring the communication is presented.

2.1.2.1.2. When is speech act theory application appropriate ?

Related to this question is criticism on speech act theory concerning issues of power, control and rational design of work organization and IS support. "This criticism concerns problems with rigid work design versus needs for flexibility, and issues of power relations, such as authority and control versus autonomy. Designing for change and flexibility will entail possibilities of learning, while routinization may lead to deskilling and alienation" [Holm, 1996]. These issues raises the question to what extent it is possible and desirable to achieve a rational design of work ? The need for skill, flexibility and social responsibility may prevent the possibility of achieving a rationalistic work design.

One can consider all social activities to be designed. Everyone is affected by social conventions and rules in ones actions, in other words all activities are situated and performed with a certain freedom and responsibility. It is important to consider that in introducing new ISs in a change process, decisions must be made regarding the character of the structure, plans and control of work, i.e. how work is (re)designed. Some work characteristics, like utilization and development of human skills and continuous learning, utilization and development of social competence and responsibility, and a rich and diversified human interaction make a strict rationalistic design of work less desirable (and possible) ([Ljungberg and Holm, 1996]).

This thesis only looks at the design of ISs supporting the communication in organizations, thereby coordinating business activities. In this sense it is limited. One should take into account that this is only one aspect in the (re)organization of work.

When applied to situations that can be routinized and controlled, and when taking into account the adaptation of speech act theory, taking a language-action perspective to the design of business communication and the support of IS can be both appropriate and applicable (but as described above, only for that).

2.1.3. LAP APPROACHES

In 1996 the first international workshop on the LAP was organized ([Dignum et al., 1996c]). Existing applications were reviewed and also new developments were presented. This section describes and comments upon some of the older and newer approaches that bear similarities with the work presented here.

In the Netherlands, LAP research has been performed within the framework of the Dutch SION-funded LIKE (Linguistic Instruments in Knowledge Engineering) project. Previous publications this work is partly based on include the following. The use of linguistics for ISs was described in [Dignum, 1989], [Weigand, 1990], [Weigand, 1991a] and [Weigand et al., 1996]. Communication aspects have been described in [Weigand, 1990], [Weigand, 1993], [Dignum and Weigand, 1995a, 1995b], and [Weigand et al., 1995], [Verharen and Dignum, 1997]. Semantics and multimodal logics have been described in [Wieringa et al., 1989], [Weigand et al., 1995], and [Dignum et al., 1996a, 1996b], while modelling methods for modelling Information and Communication Systems based on linguistics were described in [Verharen et al., 1994], [Verharen and Weigand, 1994], and [Burg and vdRiet, 1995].

2.1.3.1. ORIGINS

The initial impetus of Flores and Ludlow ([Flores and Ludlow, 1980]) has resulted in a first wave of applications within the LAP. The most important results were laid down in the Coordinator, a communication supporting tool ([Winograd and Flores, 1986]). They developed a model of communication in a work environment, and the 'conversation for action' schema is a well-known model of commitment management. They point to the fact that every conversation is governed by rules, which constrain the actions of the participants, e.g. a request must be followed by an accept or decline, a question by an answer, and so on. The schema has been criticized and modifications of it have been suggested or realized, e.g. [Dietz and Widdershoven, 1991], [Medina-Mora et al., 1992].

Another (early) application of the LAP is the office communication analysis method SAMPO by Lyytinen and colleagues ([Lehtinen and Lyytinen, 1986], [Auramäki et al., 1988, 1992a,b]). Based on Searle's speech act theory and discourse analysis, it does not instigate any predefined model for communication, but is an approach for modelling communication enabled through ISs, for which it provides concepts and notations. Its view of commitments and conversations is broad, and not only covers conversations, speech acts and their perlocutionary effects, but also physical acts.

This thesis follows their recognition of the need to understand commitment negotiations and contracts between people, but the approach here offers a higher level of abstraction by concentrating on the essential communication. In [Auramäki and Lyytinen, 1996] a new theory on speech act understanding, acceptance and mutual commitment is proposed, based on the speech act theories of Searle and Habermas and looking into the conditions for speech act success, taking into account context. They emphasize the importance of listening and learning, and understanding of the work processes. The work however is still in its infancy. Like in this approach the concept of contract as set of mutual commitments plays a central role.

The increasing importance of communication, CSCW, and technological development, is responsible for the second wave of applications within the realm of the LAP.

The Milan Conversation Model ([De Michelis and Grasso, 1994]) resembles the CHAOS model ([De Cindio et al., 1988]) and even the Coordinator, however the commitment negotiation is made more explicit. The model includes the steps: request/offer, modified request/offer, counter request/offer, agreed on request/offer, delivered, accepted/refused, and cancelled. The approach taken here is based on a different negotiation pattern, based on Goldkuhl's BAT approach described below, and worked out in the next chapter.

[Chang and Woo, 1994] present the speech act based negotiation protocol SANP, informed by the speech act classification of [Ballmer and Brennenstuhl, 1981]. I agree with their critique on existing agent negotiation protocols as being inflexible and not based on existing negotiation models. However, SANP seems to be based on a strategic action approach in which agents strive after the accomplishments of their own goals. This is in contrast with our approach where the focus is on cooperation and coordination instead of trying to resolve conflicts. The approach in this thesis is broader, and negotiation between the agents is only a small (but not unimportant) part of the communication between them.

Another difference with the approaches above is that here the deontic aspects (authorization and obligation) are modelled explicitly and have one underlying logical framework (based on deontic logic) that describes the semantics of the models.

2.1.3.2. BUSINESS PROCESS MODELLING FRAMEWORKS

The Language-Action Perspective gives content to a new generation of Business Process Models ([Teufel and Teufel, 1995], [Goldkuhl, 1995]) and communication modelling ([Johannesson, 1995]). Examples are DEMO (also participant in the LIKE project), Action Workflow, and Business as Action game Theory, discussed below.

2.1.3.2.1. DEMO

DEMO (Dynamic Essential Modelling of Organizations) ([Dietz, 1992a,b, 1994a, 1994b]) is a business process modelling method based on social theory, grounded in the philosophy of Searle and Habermas. The motivation behind DEMO is the strongly felt need to have a theory about the dynamics of activities in organizations for IS analysis. The following description of DEMO is based on [Steuten and van Reijswoud, 1996] and [Dietz et al., 1996].

DEMO starts from the understanding that an organization is a social system, consisting of communicating actors. An actor is a particular function or activity to be performed by a social individual. We share the distinction between object world and subject system (conform the separation between UoD and EoD). The state of the subject system represents the progress made in performing activities, the state of the object world represents only the result of these activities (represented by facts). In studying organizations three levels of abstraction can be distinguished:

- the *documental* level. An organization is viewed as a system of actors that produce, store, transport and destroy documents;
- the *informational* level. Here one focuses on the semantic aspect of information. An organization is viewed as a system of actors that emit and receive messages

and reproduce and derive information. Here most current methods (e.g. DFD and ER) aim to be helpful, or as Dietz states, can only be helpful, in spite of farther reaching claims;

- the *essential* level. Here the focus is on pragmatic meaning of messages, i.e. their role in carrying on the business activities. An organization is viewed as a system in which actors perform performative conversations, i.e. conversations resulting into a subject system or object world state change. Because only performative conversations create original new facts, they are considered to represent the essence of an organization.

In initial analysis of an organization this essential level is modelled by an essential model, an integrated whole of four partial models: the Communication, Process, Fact and Action Model, discussed in more detail in section 7.5.1. In DEMO a communication act consists of a proposition, represented by a predication (description of the desired result of an utterance) and a time-for-completion. It also contains an action-coordinating indicator, composed of the verb of the utterance extended with the illocutionary class to which it belongs (e.g., declaration, promise, request, or order), and the action-type as proposed by Habermas' theory of communicative action (strategic, communicative, or discursive), possibly indicating the validity claim (truth, justice, or sincerity) under discussion.

According to Dietz communicative acts in business communication are related to each other according to a specific pattern, called the transaction pattern. The pattern consists of a communication part and an action part (see figure 2.1).

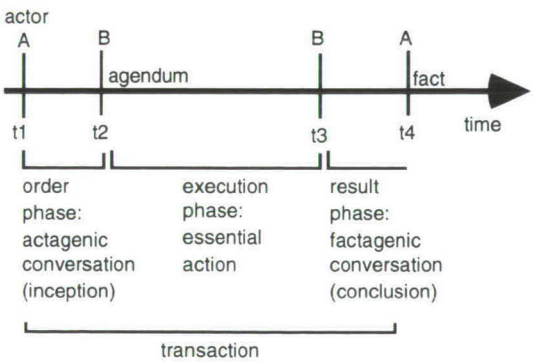


figure 2.1. Transaction pattern
(after [Dietz, 1994b, pp. 83])

The transaction starts with a request of the initiator A (at time t1). The participants involved in the transaction, called actors, reach (at t2) a commitment for a future action, called agendum (thing-to-do), added to the agenda for the actor involved. Next, the action agreed upon is executed by the executor (t2-t3). Finally, the parties try to reach an agreement about the result of the action. When the initiator accepts the result, the transaction succeeds and a fact is created (t4). The fact corresponds with the predication of the communication act as mentioned above.

According to Dietz, the essence of the behaviour of an organization consists of the continuous accomplishments of such transactions between actors.

Dietz starts the modelling without taking into account the existing ISs in an organization. This originates from the idea that essential conversations and actions can only be performed by responsible, authorised subjects and not automated systems. Other activities could (often more efficiently) be performed by artefacts. This holds especially for all actions that are purely and only informational. This are actions of reproduction (the actions usually performed by databases) and derivation (mathematical or logical computation, otherwise said, the processing of information).

The approach taken in this thesis does take into account legacy systems. Although CIAs fall under the responsibility of a user (subject), in my view more tasks than the purely processing and storing ones can be delegated to a CIA. There is no formal logic describing the DEMO models, and they especially lack deontic concepts. At the moment it is unclear how the DEMO models can be used as input for the design phase of an (automated) IS. Another difference is the grouping of transactions in contracts, describing their deontic effects. However, DEMO models can be used within the methodology, presented in chapter 7, in the analysis phase to model the business communication. Since they are based on the same principles of the LAP they can be taken as the input for the formal specification phase for an CIA.

2.1.3.2.2. Action Workflow

Action Workflow ([Medina-Mora et al., 1992], [Action Technologies, 1993], [Denning and Medina-Mora, 1995]) is a theory about the organization of work taking a LAP and relies on theoretical work of [Flores and Ludlow, 1980], [Winograd and Flores, 1986], [Winograd, 1988].

Action Workflow can be seen as generic business framework, or a business process and workflow analysis and modelling method, and is also the name of a supporting software tool. It uses the 'work is a closed loop' idea (figure 2.2).

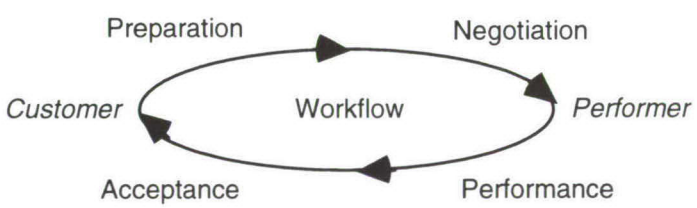


figure 2.2. Action Workflow
(from [Action Technologies, 1993])

Business processes are split up in elementary transactions between a customer and performer and consist of the steps: preparation, negotiation, performance, and acceptance. The first two steps aim at the establishment of a commitment of the addressee to perform an action. The last two steps aim at the establishment of the performed action. The action itself is not modelled, only its results. In both parts there is negotiation aimed at mutual

agreement of what has to be established. The Action Workflow theory (with its roles and phases) can be seen as a generic blueprint for the organization of work.

Action Workflow is a 'one-way-round model', it starts (generally) with a customer request, and through a performer's commitment and work it ends up with customer satisfaction. It is rather one-sided in actor (only customer) focus and action focus. The business logic presented in the next chapter emphasizes the mutual character of the transactions, it is an exchange process with mutual commitments, fulfilments and satisfactions, or a 'two-directions co-action model' ([Goldkuhl, 1996]).

2.1.3.2.3. Business as Action game Theory

[Goldkuhl, 1995] and [Goldkuhl, 1996] describe the Business as Action game Theory (BAT). BAT is a generic framework for business transactions between companies. It describes the roles of two actors (a supplier and a customer of products (goods or services)) and their communicative and material actions which build up a business logic. It is strongly influenced by communicative action theories and therefore a business transaction is not mere information transfer, but a truly communicative action.

For the modelling of business transactions BAT uses (enhanced) Action Diagrams from the SIMM method ([Goldkuhl and Röstlinger, 1988], [Goldkuhl, 1992]), a method for business modelling (BPR). However, there is not a direct correspondence between the underlying theoretical framework and the modelled work flow maps in Action Diagrams (as for instance is the case in Action Workflow and DEMO). The framework of BAT is not enforced in a strong way, which is a disadvantage.

This thesis follows BAT. In contrast to speech act theory and Action Workflow, the focus is on the satisfaction of both supplier and customer. The emphasis is on mutual commitment (laid down in contracts). There are differences in the models used (e.g., our contracts can describe the mutual commitments in all phases of the business relation, not only in the execution phase). Like DEMO, BAT is aimed at the analysis and redesign of business processes. It lacks the possibility of (formal) specification of information agents.

2.1.3.2.4. Discussion

Action Workflow (like DEMO) emphasizes communicative actions in every phase, also in the performance phase. I follow [Goldkuhl, 1996] which states: "this is an overemphasis on communication. The most important is the (often material) actions ... leading to the delivery of products. In many situations there is no separate message of delivery. The delivery itself has an informative character to the customer". In BAT the focus is on the wholeness of communicative and material actions and their business logic.

The techniques described in this thesis aim at providing a formal way of specifying business process models such as DEMO, Action Workflow and BAT. The approaches however have different scopes. E.g., DEMO offers a rigid methodology, but only on the transaction level. BAT is a generic task model that is particularly aimed at free market exchanges (and business transactions between two companies), whereas Action Workflow is a generic task model (and perhaps transaction model) that is more oriented towards an organizational context. The models do not pay much attention yet to failure handling.

E.g., Goldkuhl only recently added claim management in BAT ([Goldkuhl, 1996]), but it is not worked out in any way, especially the mentioning in the contract of what should happen in the case of claims is missing. Although the models recognize the importance of commitments they lack an underlying formal logical framework in which such deontic aspects can be described. Furthermore, not much attention is paid to authorizations. Section 7.5 contains a more detailed discussion on the modelling methods.

2.2. INTELLIGENT AGENTS

In this thesis ISs are considered to be intelligent communicating agents. Agents have gained considerable interest, especially from the Distributed AI community. In this section we look at what we perceive the most important issues related to the design and construction of intelligent agents. It describes aspects of multi-agent systems and gives an overview of some of the approaches taken and architectures developed in the agent research community. The structure of this section partly follows the excellent agent-paradigm overview of [Wooldridge and Jennings, 1995].

The *agent* concept has become important in both AI and mainstream computer science. In fact, the term has been used in such diverse ways that it has become meaningless without reference to a particular notion of agenthood. The term 'agent' certainly is not exclusive to computer science disciplines. In general, an agent is *a person, or entity, which performs some tasks, usually on behalf of someone or something else.* This original meaning still exists in e.g. intelligent-interfaces, where 'software agents' carry out the user's wishes, and also in the agency theory in economics. In this thesis the focus is on agents that intelligently perform their tasks. And although some of the theories presented here are applicable to agents in general, the focus is on artificial agents, i.e. agents within automated systems.

2.2.1. DEFINITIONS OF AGENT

This subsection is concerned with the question of what an agent is. As was (or, is) the case with Object-Orientation almost every researcher has a different definition of agent. [Wooldridge and Jennings, 1995] describe how Carl Hewitt remarked (at the 13th international workshop on Distributed AI) that the question "what is an agent ?" is as embarrassing for the agent-based computing community as the question "what is intelligence ?" is for the mainstream AI community. In AI the term 'agent' often refers to an entity that functions continuously and autonomously in an environment in which other processes take place and other agents exist. Although there is no single universally accepted definition of what an agent precisely is, there seems to be some consensus on the properties that make up an agent[8]:

[8] Another classification could be one along the lines of [Goodwin, 1993]; an agent should be: *successful* (it accomplishes the specified task in the given environment), *capable* (it possesses the effectors needed to accomplish the task), *perceptive* (it can distinguish characteristics of the environment, needed for its effectors), *reactive* (able to respond sufficiently rapid to events in the

- *autonomy*: agents operate without the direct intervention of humans or others, and have control over their actions and internal state ([Castelfranchi, 1995]);
- *social ability*: agents interact with other agents (and possible humans) via some kind of agent-communication language ([Genesereth and Ketchpel, 1994]);
- *reactivity*: agents perceive their environment (which may be the physical world, a user, a collection of other agents, the internet, or perhaps all of these combined), and respond in timely fashion to changes that occur in it;
- *pro-activeness*: agents do no simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative.

This, what is often called *weak notion* of agency, is also used in the emerging discipline of agent-based software engineering. Often agents following this description are called softbots (e.g., [Etzioni et al., 1994]).

However, for researchers working in AI the term 'agent' has a stronger and more specific meaning. The *strong notion* of agency usually starts from the notion of an agent as an entity "whose behaviour can be predicted by the method of attributing beliefs, desires and rational acumen" ([Dennett, 1987]), or *intentional system*, as he coined it. Searle saw speech act theory as leading to a theory of *intentionality* ([Searle, 1983]): "Intentionality is that property of mental states and events by which they are directed at or about objects and states of affairs in the world.....I [call this] feature of directedness or aboutness 'Intentionality'".

Dennett chose to refer to these attitudes as intentions instead of intentionality. He gave a good account of intentional systems as starting point in the explanation of the behaviour of complex systems ([Dennett, 1981]): "Intentional explanations have the action of persons as primary domain, but there are times when we find them and predictions based on them not only useful but indispensable for accounting for the behaviour of complex machines".

McCarthy ([McCarthy, 1979], quoted in [Shoham, 1993]), has argued that taking the intentional stance is appropriate and not just anthropomorphism: "To ascribe beliefs, free will, intentions, consciousness, abilities, or wants to a machine is *legitimate* when such an ascription expresses the same information about the machine that it expresses about a person. It is useful when the ascription helps to understand the structure of the machine, its past or future behaviour, or how to repair or improve it. ... Theories of belief, knowledge and wanting can be constructed for machines in a simpler setting than for humans.... Ascription of mental qualities is *most straightforward* for machines of known structures, such as thermostats and computer operating systems, but is *most useful* when applied to entities whose structure is incompletely known".

The standpoint 'agents as intentional systems' is taken by most agent theories developed today, however, mainly for pragmatic reasons. [Singh, 1994] also supports the intentional stance for describing agent systems with this in mind. He argues that: "the intentional stance makes available such abstractions as the intentions and know-how of

---

environment), *reflexive* (behaves in stimulus-response fashion); and the deliberative properties: *predictive* (its model of how the world works is sufficiently accurate to allow it to correctly predict how it can achieve the task), *interpretive* (it can correctly interpret its sensor readings), *rational* (it chooses to perform actions that it predicts will achieve its goals), and *sound* (an agent is sound if it is predictive, interpretive and rational).

agents, and the communication that takes place among them ... These abstractions no doubt have much conceptual appeal. Furthermore, there are simple pragmatic and technical reasons for considering them seriously. They (i) are natural to humans, who are not only the designers and analyzers of (multi)agent systems, but also the end users and requirements specifiers; (ii) provide succinct descriptions of, and help understand and explain, the behaviour of complex systems; (iii) make available certain regularities and patterns of action that are independent of the exact physical implementation of the agents in the system; (iv) may be used by the agents themselves in reasoning about each other".

The strong notion of agency, in addition to the properties given above, characterizes an agent using mentalistic notions, or *intentional attitudes*, like *knowledge, belief, desire, goal, intention and obligation, or commitment*.

[Kiss, 1992] classifies intentional attitudes into three groups:

- *cognitive*, referring to epistemic issues, such as beliefs, knowledge, awareness;
- *conative*, referring to action and control, denoting an attempt to perform an action. To this category belong intention, commitment, plan.
- *affective*, referring to those attitudes which correspond to the dynamics of an agent's behaviour, classifying goal, desire and preference.

A different classification is given in [Shoham and Cousins, 1994] where intentional attitudes are divided according to their relevance to computational applications:

- *informational*, they concern the information available to agents. Belief, knowledge and awareness belong to this category;
- *motivational*, "are in some sense directly linked to the agent's selecting one among the various possible actions available to it" [ibid.]. Attitudes include intention, choice, plan, goal, desire, commitment, preference, and wish. As the authors observe some are better understood than others, and the meaning of all of motivational terms is far less clear than those for knowledge and belief;
- *social*, these are related to the motivational attitudes but give social, moral and/or rational reasons for behaving in a certain way. To this category belong obligation, and permission.

The authors state that for other emotions such as fear, joy etc. their relation to computational applications is not yet obvious.

Other attributes sometimes discussed in the context of agency are:

- *mobility*: the ability to move around on an electronic network ([White, 1994]);
- *veracity*: the assumption that an agent will not knowingly communicate false information ([Galliers, 1988]);
- *benevolence*: the assumption that agents do not have conflicting goals, and every agent will therefore always try to do what is asked of it ([Rosenschein and Genesereth, 1985]);
- *rationality*: (crudely) the assumption that an agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved —at least insofar as its beliefs permit ([Galliers, 1988]).

There is an ongoing discussion about the rationality of agents. Most researchers follow Dennett's interpretation of rationality which refers to the quality of following the rules of logic, and provide a status for the beliefs that a rational agent holds.

2.2.2. AGENT THEORIES

This subsection is concerned with (logical) formalisms for representing and reasoning about agent properties. An agent theory is regarded as a specification for an agent, it captures the desirable properties represented by formalisms.

Although there is no consensus in the AI community about precisely which combination of *information attitudes* (related to the information an agent has about its environment, e.g. knowledge and belief) and *pro-attitudes* (attitudes that in some way guide the agent's action, e.g. goals, desire, intention, commitment, choice, ... [Bratman, 1990]) are best suited to characterize agents, a number of approaches have gained much support. Whatever attitudes are chosen, a complete agent theory defines how the attributes of agency are related, e.g., it will need to show how an agent's information and pro-attitudes are related; how an agent's cognitive state changes over time; how the environment affects an agent's cognitive state; and how an agent's information and pro-attitudes lead it to perform actions.

Cohen and Levesque ([Cohen and Levesque, 1990a]) developed a theory of intention which the authors required as a pre-requisite for a theory of speech-acts ([Cohen and Levesque, 1990b]).

Rao and Georgeff ([Rao and Georgeff, 1991a], [Rao and Georgeff, 1991b], [Rao and Georgeff, 1993]) developed a logical framework with three primitive modalities: *Belief*, *Desires* and *Intention*. They distinguish between goals and desires in that: (i) desires can be inconsistent with one another, but goals must be consistent. In other words, goals are chosen desires of the agent that are consistent; (ii) the agent should believe the goal is achievable. Although in [Rao and Georgeff, 1992], and [Kinny et al., 1992] the potential for adding (social) plans is considered, the BDI framework considers agents in isolation; it ignores communicative aspects that are central in this thesis approach.

Another influential approach is the one taken by Shoham ([Shoham, 1993]) in which he uses the mentalistic notions *belief*, *obligation* and *capability* to characterize an agent. He also describes a new programming paradigm based on this notion of agents: agent-oriented programming, see also section 2.2.4.

An interesting formal framework, consisting of the attitudes intention, belief, knowledge, know-how and communication, has been developed by Singh ([Singh, 1994]) and will be briefly discussed in section 2.2.5.4.2.

Yet another interesting framework is the Information-Motivation-Action-Social (IMAS) model ([Dignum and van Linder, 1996]) based on the intentional attitudes classification of [Shoham and Cousins, 1994], but with emphasis on the *capabilities* of agents as inherent attitudes of agents. [Thomas, 1993] and also [Belnap and Perloff, 1989], [Elgesem, 1993] support the view that capability should be taken as a primitive notion not reducible to other notions. In IMAS Information corresponds to knowledge and belief

attitudes, Motivation to the desires and intentions of the agent (represented by goals and plans), the Action component gives a description of the capabilities and actions of an agent (specified in a dynamic logic), and finally the Social component describes the communication and cooperation attitudes.

Agent theories and formalisms do not have to be rooted in logic as Werner shows. In [Werner, 1989, 1991] he laid the foundations of a general model of agency which draws upon work in economics, game theory, situated automata theory, situation semantics and philosophy. The properties of this model, however, fall beside the scope of this thesis.

See [Wooldridge and Jennings, 1995] for an overview of yet other approaches.

2.2.2.1. INTENTION, ACTION AND COMMUNICATION

Most problems with respect to logics that combine different attitudes relate to *intention*. [Wooldridge and Jennings, 1995] state that the relationship between intention and action has not been formally represented in a satisfactory way, although recent work of Singh and van Linder shows some perspective (see below). The problem to tackle is that having an intention to act makes it more likely that an agent will act, but does not generally guarantee it. While it seems straightforward to build systems that appear to have intentions (e.g., [Wooldridge, 1995]) it seems much harder to capture this relationship formally. Other problems include the management of multiple, possibly conflicting intentions, and the formation, scheduling and reconsideration of intentions. In sections 5.2 and 5.3 an alternative approach is taken, concerning contracts and tasks.

Several theories of *action* have been proposed, many including quite restrictive assumptions like: only one event happens at a time, events have precisely determined effect, events are necessarily associated with a state change. Dynamic logics provide a rich syntax for actions, but generally do no consider time. On the other hand temporal logics contain operators to deal with time, but do not explicitly consider actions. Only few consider the ability to perform an action as essential to an agent. Moore ([Moore, 1980, 1985, 1990]) studied the question of what an agent needs to know in order to be able to perform some action. He formalised a model of *ability* in a logic containing a modality for knowledge, and a dynamic logic for modelling action ([Harel, 1984]).

Singh ([Singh, 1994]) gives a dynamic logic variant in which actions are explicitly related with time. Intentions correspond to the courses of events preferred by the agent. Singh states that intentions are determined not only on the basis of the goal the agent is currently pursuing, but also on the basis of the actions it can perform (plans), its abilities and 'know-how'. Know-how refers to the knowledge of how to act and knowledge of skills, which Singh distinguishes from 'know-that' which refers to the knowledge of facts.

The KARO (Knowledge, Ability, Result, Opportunity) framework [van Linder, 1996] incorporates many of the aspects described above. Besides knowledge it emphasizes actions and events. Action are descriptions of causal processes, which upon execution turn one state of affairs into another. An event is the performance of a particular action by a particular agent. The focus is on three aspects of actions and events: result, opportunity and the agent's abilities. Based on this a logic of capabilities is developed in which non-deterministic actions can be considered.

Formalisms for representing *communication* in agent theory tend to be based on a simple interpretation of communication: the exchange of information. Work on agent communication has mostly been done in agent architectures and languages and not in agent theories. However recent multi-agent theories (e.g., [Singh, 1994], [Haddadi, 1996]) include formal theories of communication in the lines of Searle's speech act theory, and pay special attention to the commitments resulting from communication. See also section 2.2.5.4 for a short discussion of their work.

2.2.3. AGENT ARCHITECTURES

Agent architectures can be thought of as software engineering models of agents. They represent the move from specification to implementation.

Maes defines an agent architecture as: "A particular methodology for building agents. It specifies how the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to how sensor data and current internal state of the agent determine the actions and future internal state of the agent. An architecture encompasses techniques and algorithms supporting the methodology" [Maes, 1991].

The most common view to build intelligent agents is to see them as a particular type of knowledge-based system. Three approaches can be distinguished: the deliberative, reactive and hybrid architecture.

Deliberative architectures describe 'deliberative agents' ([Genesereth and Nilsson, 1987]) meaning a specific type of symbolic architectures. [Wooldridge and Jennings, 1995] define a deliberative agent or architecture "to be one that contains an explicitly represented symbolic model of the world, in which decisions (e.g. about what actions to perform) are made via logical (or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulation". It is clear that there is a close relationship between deliberative architectures and theories based on mathematical logic.

Examples of deliberative architectures are:

- IRMA (Intelligent Resource-bounded Machine Architecture) ([Bratman et al., 1988]), an agent architecture based on the BDI framework of Rao and Georgeff.
- GRATE* ([Jennings, 1993b]), a deliberative architecture that takes multi-agent worlds into account, based on beliefs, desires, intentions and joint intentions.
- DESIRE ([Brazier et al., 1996]). Although not truly an agent architecture it is mentioned since it focuses on the study of compositional multi-agent systems for complex (and distributed) tasks and development methods for these systems.

A *reactive architecture* is one that does not include any kind of central symbolic world model and does not use complex symbolic reasoning, and therefore usually reacts much faster to events in its environment. [Wooldridge and Jennings, 1995] list a number of reactive architectures. Examples are:

- the subsumption architecture ([Brooks, 1986, 1991a,b]), used frequently in robot approaches. A subsumption architecture is a hierarchy of task-accomplishing behaviours that 'compete' with others to exercise control over an agent.

- the situated automata paradigm ([Rosenschein, 1985], [Rosenschein and Kaelbling, 1986], [Kaelbling and Rosenschein, 1990], [Kaelbling, 1991]). An agent is specified in declarative terms which is then compiled down to a digital machine.
- the Agent Network Architecture ([Maes, 1989, 1990, 1991]). An agent is a set of competence modules, specified in pre- and post-conditions and an activation level giving a valued indication of the relevance of the module in a particular situation. The higher the level the more likely it will influence the behaviour of the agent.

Many researchers argue that *hybrid systems*, neither completely deliberative nor completely reactive, are most suitable for building agents. An obvious approach is to build an agent with two subsystems: a deliberative one containing a symbolic world model which develops plans and makes decisions in a symbolic way; and a reactive one capable of reacting to events in the environment without engaging in complex reasoning, preferably with precedence over the deliberative one to provide rapid response to important events.

Examples of hybrid architectures are:

- TOURINGMACHINES ([Ferguson, 1992a,b]).
- PRS (Procedural Reasoning System) ([Georgeff and Lansky, 1987]).
- COSY ([Burmeister and Sundermeyer, 1992], [Haddadi, 1995]).
- InteRRap ([Müller and Pischel, 1994], [Müller, 1994], [Müller et al., 1995]).

The main problem with hybrid systems is to combine the interacting subsystems (deliberative and reactive) cleanly in a well-motivated control framework. Often, hybrid systems tend to be ad-hoc: their structure is well-motivated from a design point of view, but it is not clear how to reason about them, or what their underlying theory is. Especially architectures that contain subsystems that compete to get the control over the agent seem to defy any attempt at formalisation [Wooldridge and Jennings, 1995].

2.2.4. AGENT LANGUAGES

Agent languages are software systems for programming and experimenting with agents. These systems may embody principles proposed by theorists and follow guidelines or design rules set out by agent architectures. Such languages often include (some of) the attributes of agency (mentalistic or otherwise) as described above. Beforehand we can say that most agent languages developed today support the weak notion of agency, delivering kinds of 'softbots', while languages supporting the strong notion of agency mostly are based on the logics developed for agents.

Nowadays agent languages gain much attention and new languages are designed almost on a monthly basis. Therefore no up-to-date overview can be given. Here I only want to mention an agent communication language, based on speech acts and the only standardized language today, and AOP, a new agent programming paradigm.

Most work in agent communication languages focused on developing protocols for communication (as message sending and information exchange) ([Genesereth and Ketchpel,

1994]). The best known is the ARPA knowledge sharing effort ([Patil et al., 1992]) to develop three related languages: KQML (Knowledge Query and Manipulation Language) ([Finin et al., 1993, 1994]) for communication to and between knowledge based systems, describing a standard syntax for messages and a number of performatives (e.g. tell, ask, reply) that define the force of the message, based on speech act theory; KIF (Knowledge Interchange Format) ([Genesereth and Fikes, 1992]) a standardized knowledge representation language, describing a syntax for message content (first-order predicate calculus recast in a LISP-like syntax); and Ontolingua, a language for specifying standard ontologies.

One of the most influential developments is the *agent-oriented programming* (AOP) paradigm ([Shoham, 1993]), as a successor to object-oriented programming (OOP), based on a 'societal view of computation' of multiple interacting agents. Whereas OOP views a computational system as a set of modules that can communicate with each other and that have individual ways of handling incoming messages, AOP specializes the framework by programming the state of the modules (agents) directly in terms of mentalistic, intentional notions. An AOP system has three components: (i) a logical system defining the mental state of agents; (ii) an interpreted programming language for defining and programming the agents; (iii) an 'agentification process', for compiling agent programs into low-level executable systems, and converting neutral devices into programmable agents.

A restricted formal language is provided with clear syntax for describing the mental state. The logic contains the modalities *belief* (beliefs about the world, about itself, and about one another), and *decision* (or *choice*). A third category which is not a mental construct per se is *capability* (things the agent can do, including private actions and sending messages). Instead of taking decision as basic, Shoham starts with the notion *obligation* or *commitment*, and treats decision as an obligation to oneself.

Shoham's first attempt was the AGENT-0 system. The mental categories described above appear in the language itself, and the semantics of the programming language is related to the semantics of the mental constructs. Although no formal semantics are given in [Shoham, 1993] it appears to be based on [Thomas et al., 1991].

Although Shoham's work has been an inspiring example (I used AGENT-0 to build a prototype implementation, [Verharen et al., 1994]), I feel that there are shortcomings to his approach of speech acts in general and commitments and obligations in particular. Agents communicate in different ways, and as said before, communication is more than information transfer or message sending. The speech acts Shoham uses (Request, Inform and Cancel) are too simple to set up good communication. Although in [Shoham, 1993] he stresses the importance of commitment as a result of communication he never enforces the obligations resulting from it. Instead of a logical semantics to his language, (pointers to) an operational semantics are given. The reason for this is that Shoham requires that the language be efficiently executable, and a logical semantics for the language would require a multimodal, temporal theorem prover, which would become notoriously inefficient.

However, AOP should be acknowledged as the first approach to make computational agents, paying attention to both the abilities of an agent and the importance of good communication structures (beyond mere message sending and network protocols). In that way it has been an inspiration for many agent research, including this one.

AGENT-0 was only intended as a prototype, to illustrate the principles of AOP. PLACA (Planning Communicating Agents) ([Thomas, 1993]) is a more refined implementation of the AOP ideas and overcomes one severe drawback of AGENT-0: the inability of agents to plan, and communicate requests for action via high-level goals. The development of plans means that multiple actions (private actions, message sending, etc.) are composed into plans, which are compared with the agent's environment. Once the agent has determined a suitable plan, it will make a commitment to execute it. The agents in PLACA are programmed in terms of mental change rules, and the logical component is similar to that of AGENT-0 but includes operators for planning and intentions (achieve goals, or a commitment to achieve a state of the world). In [Thomas, 1993] a detail semantics of the logic and its properties are examined. Like AGENT-0, PLACA is only an experimental language.

Another extension of the AGENT-0 system is the Agent-K system ([Davies and Edwards, 1994]) that integrates the ideas of AOP with the communication language KQML. The work is motivated by the limited set of the communicative messages in AGENT-0. The system includes a modified interpreter that is able to deal with all KQML messages, and is implemented in Prolog and a library of C-routines that allow an agent to send KQML messages using TCP/IP, HTTP or email.

2.2.5. MULTI-AGENT SYSTEMS

In most agent approaches (especially theories) described above only agents in isolation are considered. The last couple of years a tremendous amount of research has been done on Multi-Agent Systems (MAS) (see e.g. the many conference proceedings of agent related conferences and workshops (MAS, MAAMAW, AAAI, DAI, ... [O'Hare and Jennings, 1996]). When looking at societies of agents interesting problems occur, such as joined commitments, coordination, and negotiation. This section describes some of the approaches taken in solving multi-agent coordination.

2.2.5.1. COMMITMENT AND CONVENTION

If a society of agents is to function successfully, global constraints must be imposed, including social rules and social roles, both reducing the problem solving required by agents and the communication overhead. Examples of literature on computer societies are Minsky's informal Society of Mind metaphor [Minsky, 1986], Winograd's studies of societal roles ([Winograd, 1988]), and the work of Shoham and others [Shoham and Tennenholtz, 1992] investigating the off-line design of social laws.

The research in Distributed AI (DAI) focuses on distributed problem solving, and often DAI systems are modelled as distributed goal search problems. The key issue in joint problem solving is coordination, defined as the process by which an agent reasons about its local actions and the (anticipated) actions of others to try and ensure the community acts in a coherent manner.

[Jennings, 1993a, 1996] argues that commitments and conventions form the foundation of coordination for joint problem solving. Conventions are defined as general policies for governing the reconsideration of commitments, indicating whether to retain, rectify, or abandon commitments. In pursuing a joint goal cooperating agents must make joint commitments. Since a shared mental state of the joint commitment is impossible without sacrificing autonomy, social conventions are specified to keep all agents informed of changes in the joint commitment. Designing them is difficult, since it is important that relevant information pertaining to changes in commitment is disseminated as soon as possible. But, agents should not broadcast information about their commitments every time they change, but only those relevant to the joint commitment or action.

[Durfee et al., 1989] identifies three ingredients for successful coordination: (1) structures that enable the agents to interact in predictable ways; (2) flexibility so agents can operate in dynamic environments and can cope with their inherently partial and imprecise view of the community; (3) knowledge and reasoning capability to exploit the available structure and flexibility. Based on this Jennings ([Jennings, 1996]) argues that commitments provide the necessary structures for predictable interactions, conventions provide the flexibility needed to operate in dynamic environments, and social conventions provide the necessary degree of mutual support. Thus

coordination = commitments + conventions + social conventions + local reasoning

The agents described in this thesis are not joint problem solvers and do not have joint goals. They have their own tasks and only need to cooperate if a service is required from another agent (or if another agent requires a service) for fulfilling a goal. Coordination of activities is reached by mutual agreement on the services required. However, the mutual agreement, laid down in a contract, can be seen as containing social conventions, since the contract describes the mutual obligations (commitments) of the communicating parties and also how they change if one of the parties does not adhere to an obligation, or how they change in case a certain state is obtained. Furthermore, a contingency plan can be specified as part of the task that contains local knowledge of how the goals are affected if something unforeseen happens. Chapter 5 describes contracts and tasks in detail.

2.2.5.2. PROBABILITY AND UTILITY

In most work on knowledge and belief a very crisp notion of mental attitude is adopted; there is no representation of graded belief or commitment. This stands in contrast to game-theoretic work on rational interaction among agents in economics ([Aumann, 1976], [Geanakoplos, 1988]) and AI ([Rosenschein and Genesereth, 1985]) where uncertainty and utility functions play a key role. E.g., [Narazaki et al., 1995] and [Klusch, 1996] describe strategies based on utility functions for selecting communication structures (network use) in cooperative search. Much DAI research (especially that on agents as distributed problem solvers) base their models for cooperation among autonomous agents on these concepts. They start from the idea that cooperation may be mutually beneficial even if agents are selfish and try to maximize their own expected payoff ([Sycara, 1990], [Zlotkin and Rosenschein, 1991], [Kraus, 1993]).

2.2.5.3. NEGOTIATION

Negotiation is one of the notions most often stressed in DAI. The common idea in DAI is that agents use negotiation for conflict resolution and coordination.

[Müller, 1996] describes the general aim of negotiation to be modification of local plans, in the case of negative (harmful) interactions between agents, and identification of situations where potential interactions are possible. It is used for task and resource allocation, conflict recognition and solving, goal disparity resolution, the determination of organizational structure. See [Müller, 1996] for an overview, [Bond and Gasser, 1992] for an indexed bibliography of negotiation protocols, and [Rosenschein and Zlotkin, 1994] for a overview of automated negotiation. One of the oldest and widely used approach is the Contract Net Protocol ([Smith, 1980], [Smith and Davis, 1980]). Contract Nets use negotiation among a limited set of participants and a fixed protocol in order to select a course of action. The original contract nets were based on broadcasting contracts and soliciting bids, as opposed to the intimate communication in AOP, or the negotiation process from the business-as-action framework of Goldkuhl ([Goldkuhl, 1996]) that is adopted in this research. Furthermore contract nets have no other notion of mental state, no range of communicative speech acts, nor any design aspects.

In negotiation strategies often probability and utility functions, mentioned above, are used, both to assess the agent's own as well as the negotiator's standpoint.

2.2.5.4. MULTI-AGENT SYSTEM APPROACHES

Below the state of the art multi-agent approaches are described that take into account some of the concepts and techniques discussed above.

2.2.5.4.1. FCSI agents

[Klusch, 1994, 1995], and [Klusch and Shehory, 1996] describe an approach to cooperative information agents that has its basis in federated database systems, called FCSI (Federative Cell System for discovery of Interdatabase dependencies). It is an approach for recognizing interdatabase dependencies in an environment of decentralized and autonomous rationally cooperating information agents. It utilizes DAI-techniques and terminological knowledge representation and reasoning to develop agents as front-ends to autonomous databases. Each agent has a local information model (LIM) that entails a terminological description of a set of views of database schema-objects. Each FCSI agent is able to detect interdatabase dependencies by a mutual classification of terminological view descriptions in the LIM of another information agent and projecting that to the local schema of the uniquely associated database. This is similar to the idea of incrementally building and using some shared ontology for contextual interchange.

The architecture of FCSI agents is similar to that discussed in chapter 4. The main difference (except for different representations) is that cooperation between FCSI agents is performed via a decentralized, utilitarian coalition formation process, based on utility functions of the production-oriented approach of [Shehory and Kraus, 1993] that is informed by game theory, and a decentralized, bilateral coalition formation algorithm.

2.2.5.4.2. MultiAgent system framework

Singh developed a family of logics for representing intentions, beliefs, knowledge, know-how, and communication in a branching-time framework ([Singh, 1994]). The formalism is extremely rich, and considerable effort has been devoted to establishing its properties.

Singh based his protocols (as interaction specifications) on Searlean speech act theory. In the formal model speech acts are taken as actions and include a set of illocutionary forces (including permissives and prohibitives). His theory to capture an objective semantics of the messages is based on specifying the satisfaction conditions of speech acts. E.g., his theory specifies that "a directive is satisfied if (1) its proposition becomes true at a moment in the future of its being said, and (2) all along the scenario form now to then the hearer has the know-how, as well as the intention to achieve it". His theory is an objective criterion for evaluating the correctness of different scenarios (the possible runs of executions of a multi-agent system). Prescriptive specifications are required that tell agents what to do given their beliefs and intentions, so only correct scenarios can emerge.

Although Singh also recognizes the importance of permissives and prohibitives they are not linked to deontic states but are explained in increase in know-how of the agent: which actions can or cannot be performed. Nothing is said what should happen if the permissions or prohibitives are violated. Although there are similarities in the architecture the underlying framework of Singh's and the one presented here are different, both in the dynamic logic used, and the lack of deontic logic in his system.

2.2.5.4.3. Vivid agents

[Wagner, 1996] proposes a model of agents that is both logical and operational. The vivid agents model takes into account that besides the ability to draw inferences agents also need the ability to update their current knowledge state. They should be able to represent and perform (and simulate the execution of) actions in order to generate and execute plans, and to react and interact in response to perception and communication events.

The behaviour of the agents is represented by means of action and reaction rules. It is possible to specify a set of communication acts and associated communicative action rules, similar to Shoham's AOP approach. It extends this by basing the theory on knowledge systems (allowing for negation-as-failure in the query language), adopting a genuine action concept, which can account for epistemic effects, and allowing asynchronous message passing. Wagner uses the meta-logic programming formalism ([Kowalski, 1995]) to define an execution model for the agent specifications. The approach combines static knowledge in the form of a declarative knowledge base with dynamic knowledge in the form of action and reaction rules.

Wagner's approach is more elaborate than the one presented in this thesis in that he provides a theoretical foundation for hybrid agent architectures. However, he admits that more work should be done on the communication component, especially since he hopes the vivid agents model can serve as a basis for the formalization of high level concepts such as social structures, which is also the aim of our approach.

2.2.5.4.4. COSY

COSY (COoperating SYstems) ([Burmeister and Sundermeyer, 1992], [Haddadi, 1995]) is a hybrid BDI architecture. Haddadi describes cooperation protocols as a general means of designing and analyzing dialogues in pairwise interactions. She follows [Singh, 1994] in that to achieve meaningful dialogues, it is required to specify under what conditions individual agents may choose to perform a particular speech act. These conditions are specified in terms of beliefs, goals, and intentions of individual agents and their belief about the intentional states of their communicating partners.

Communicative action is considered in the context within which a dialogue may take place, and according to rules and conventions governing the interactions in that context. Reasoning about communication therefore not only involves reasoning about when and what speech act to perform, but also what is involved after a message is received (the perlocutionary effects). The message types supported are inform, query, reject, demand, command, request, offer, accept, propose and report. The perlocutionary effects are described by commitments an agent enters into. The communication seems to be a strategic one, potential for cooperation and commitments are made with the goal of task delegation or adoption. Commitments are defined by beliefs, desires and intentions.

Although there are some similarities in the architecture, the underlying framework of tasks and communication is different from the one presented here. COSY is broader in the sense that context is taken into account, which is not (yet) done in our approach.

2.2.5.4.5. Adept

Adept (Advanced decision environment for process tasks) ([Alty et al., 1994], [Jennings et al., 1996a]) aims at developing agent-based infrastructure for managing business processes.

The agents are negotiating, service providing, autonomous agents. Each agent is able to perform one or more services, which corresponds to some unit of a problem solving activity. Simple services can be combined to form complex services by adding ordering constraints and conditional control. Services are associated with one or more agents which are responsible for managing and executing them. Each service is managed by one agent, although it may involve execution of sub-services by a number of other agents. Since agents are autonomous there are no control dependencies between them, therefore, if an agent requires a service which is managed by another agent it cannot simply instruct it to start the service. Rather the agents must come to a mutually acceptable agreement about the terms and conditions under which the desired service will be performed (called Service Level Agreements (SLA), or 'contracts' in our approach).

The agent architecture is based on GRATE ([Jennings, 1993b, 1995], [Jennings et al., 1992]) and ARCHON ([Jennings et al, 1996b]) agent models. It involves an agent head responsible for managing the agent's activities and interacting with peers, and an agency representing the agent's domain problem solving resources. It allows for a nested (hierarchical) agent system in which higher-level agents realise functionality through lower level agents.

Our approach is similar to the ideas presented in [Jennings et al., 1996a], although different techniques are used (e.g., BAT as generic framework, and one formal language

to describe the actions of the agents). Especially the role of the situation assessment module corresponds to our task manager, and the role of the service execution module to the contract manager. For negotiation between the agents the negotiation model of ADEPT is adopted and is discussed in more detail in section 5.2.2. The main difference is in contract (SLA) specification and the use (and enforcement) of deontic constraints.

2.2.6. DISCUSSION

Research in DAI focuses on the knowledge of agents, rather than on their capabilities. However, the representation of the capabilities of agents (not what they think and know, but what they can and will do) is of great importance. Recent years have seen a shift to research in intentions and making commitments to act. However none of the approaches combines this with the theory of deontic logic to obtain a method to describe what it means that an agent enters into a commitment *and* be able to provide (formal) techniques to reason about and enforce deontic constraints, or at least describe what should happen if constraints are violated. Shoham recognizes the importance of obligations but focuses on single agents only and treats actions as facts.

Several approaches emphasize the importance of communication structures, and base their work on (mainly Searlean) speech act theory. However, most of them only use the basic illocutionary types to classify messages. They do not pay much attention to the rules of making speech acts work (or only very superficially, like Singh). The semantics of the speech acts usually are restricted and do not lead, like in our approach, to obligations of the agents. Communication is regarded as necessary for negotiation, and the negotiation protocols employed are mostly strategically oriented whereas in the approach taken in this thesis favours the communicative action cooperation over the strategic one.

The approach taken here has many similarities in agent architecture with those described above, especially the ADEPT system ([Jennings et al., 1996a]). In fact all deliberative agents that behave in a social way have a general architecture, of which a graphical representation is given in [Moulin and Chaib-draa, 1996]. The only differences are that some (hybrid) architectures contain more or less reactive components, especially for handling incoming data (via sensors) and performing physical actions.

Most differences between the work described here and the other approaches mentioned come from the adoption of a different logical framework. This bears consequences for any specification languages developed to support the basic constructs. Here a dynamic deontic logic approach is adopted. Although actions are a central concept, e.g., Haddadi's and Singh's approach do not allow for negation of actions. This is important if one wants to reason about what it means if an agent has *not* done an action. Other differences are the possibility to form longer transactions by combining actions and specifying constraints on them. The formal logic should give an explicit semantics to such constructs. Finally, semantics of speech acts are given by obligations and not by intentions to act.

Important to repeat here is that I do not provide a complete agent theory or model (as e.g. Singh and Haddadi) based on intentional properties. The agent architecture proposed here is mainly used to guide the design of cooperative information systems and is based on the communication framework developed.

CHAPTER 3

BUSINESS LOGIC FRAMEWORK

This chapter introduces a framework for describing business communication. It describes a generic business logic that can be used to model business communication, following the communicative action theory, between a supplier of services (or goods) and a customer needing this. The business logic is based on the Business as Action game Theory of Goldkuhl and is explained in section 1. Section 2 describes the example of booking a business trip that will be used throughout this thesis.

3.1. BUSINESS AS ACTION GAME THEORY

Goldkuhl proposes the Business as Action game Theory (BAT) ([Goldkuhl, 1995, 1996]) as a generic business framework describing a business logic. In the first subsection the work that led to BAT is described, especially the influence of the LAP. The second subsection contains the business logic framework itself, adapted to the approach proposed in this thesis. The final subsection contains a comparison of this framework with other frameworks used in business modelling.

3.1.1. MESSAGES

In the traditional infological approach ([Langefors, 1966]), communication is seen as the exchange of information only. It represent a *content view* of data and information, or referential aspect ([Holm, 1996]). This has been the basis for, for instance, the relational data model ([Codd, 1970]) the entity/relation approach ([Chen, 1976]), and the ISAC approach to IS design ([Lundeberg et al., 1981]). The main concept in Langefors' approach is the *elementary message* (e-message). It describes the object (what is talked about), property (something pointed out, a comment or proposition, on the object) and time (usually of the message sent) of the message. Instead of words or a sentence, the message is the smallest meaningful unit which communicates information. When taking a Language-Action Perspective two differences should be noted. Both concern the use and structure of elementary information units ([Goldkuhl, 1995]).

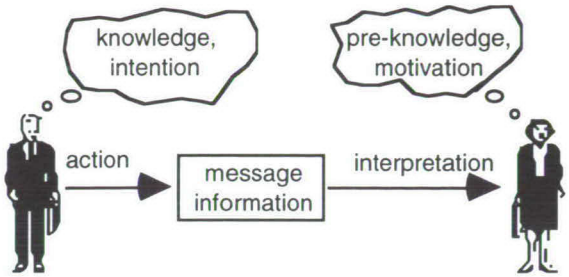


figure 3.1. Information in relation to action and interpretation
(after [Goldkuhl, 1995] fig.3)

The first difference comes from the interpretation of a message (figure 3.1). In the context of ISs the interpretation of information (and messages) seems to be 'linguistic sentences with the purpose to inform people'. Goldkuhl proposes a different conceptual usage. His interest concerns what it means to produce information (or a message). From the LAP we learned that to produce information (a message) means to act, to perform a linguistic action with a certain purpose and meaning (section 1.1.2). Pragmatic aspects are disregarded in many content oriented views of messages ([Goldkuhl, 1995]). Goldkuhl repeats that there are pragmatic aspects in the message itself and not only in the consequent actions. There is an action component in the message (figure 3.1).

This corresponds with the approach taken in speech act theory and LAP and as Goldkuhl explains "is a rejection of the classical 'objectivistic' information view in many database approaches (e.g. [ISO, 1982], [Martin, 1989]). The contents (information) in the database is seen as a mapping of an 'objective reality'...This objectivistic information conception seems to be an example of what Austin ([Austin, 1962]) calls 'the descriptive fallacy', i.e. the fallacy that language is only or mainly used for descriptive purposes" ([Goldkuhl, 1995]). In section 2.1 we saw that Austin, Searle and Habermas have shown many other uses of language, and especially that the production of information is considered to be a communicative action. The e-message is therefore extended with an action component, describing the speech act (illocutionary) type.

The second difference comes from the concept of infological equation ([Langefors, 1993]): to acquire knowledge, a receiver must perform an act of interpretation of data (the message), using his pre-knowledge (figure 3.1). However, in the infological equation only the role of the information user (receiver and interpreter of messages) is stressed. There is no explicit reference to the producer or formulator of the messages. On the other hand, in speech act theory the role of the speaker is emphasized (section 2.1). Speech act theory added to the infological equation makes it symmetrical. Goldkuhl expanded the e-message with a communicator role describing the actor responsible for the production of the message. This expansion to make the process more symmetrical also bears consequences for the interpretation of a business process: in a communicative business transaction not only the *customer* has to be satisfied, but also the *supplier* of the information.

These two differences lead to a model of symmetrical communication as action between a supplier and a customer, introduced in the next subsection.

Figure 3.1 is called the linear communication process model ([van Reijswoud, 1996]), based on the work of [Shannon and Weaver, 1949]. Van Reijswoud argues that human communication cannot be understood as being a linear process.

In the first place feedback has to be integrated in the process. Communication means that a speaker sends a message with a certain meaning expressing his mental state to a hearer that must interpret the message and extract the meaning from it (forming a mental state of the hearer). In order to be successful the expressed mental state of the speaker must be isomorphic to the generated mental state of the hearer ([Taylor, 1993]). The linear model does not provide an explicit way of checking whether this has been achieved. The model should be extended with a feedback communication from the hearer to the speaker, confirming the understanding of the message (meaning).

A second problem with the linear model van Reijswoud notices, is that communication is more than the utterance of isolated speech acts. Some speech acts are related to each other, e.g. a promise to act from a hearer as an answer to a request from a speaker. In terms of conversation analysis the two speech acts form a kind of adjacency pair ([Taylor, 1993]). It can even be argued that the first speech act is meaningless without the second.

In order to overcome these problems van Reijswoud proposes the cyclic communication process model as a model to understand the structure of human communication ([van Reijswoud, 1996]).

As already stated in section 2.1.2.1 we are not interested in analysing and explaining human communication, but instead focus on the design of communication between automated systems. For this the linear model as a technical model of communication, considering the source and destination of communication as rationally acting subjects (machines), can be used. Furthermore, communication does not always have to be two-way in that a response of the receiver of the message is necessary. In case the sender of the message has power or authority over the receiver the message (e.g. a command) has effect (e.g. the obligation of the receiver to perform an action) independent from its reply, as will be explained in detail in section 6.3.

3.1.2. BUSINESS LOGIC

In this thesis we are interested in communicative action in relation to business processes.

As Goldkuhl points out many business processes can be seen as having a *mixed communicative action* character, as opposed to the single category approach of Searle. E.g., when making an offer one performs a mixed communicative action; in this case both an attempt to influence the future action of the receiver (potential buyer) ("I want you to buy this product", a directive in Searle's classification) and a commitment of own future actions ("I will sell this product to you under these conditions", a commissive). The same holds for an order ("I want you to sell this product", and "I will buy this product from you under these conditions"). This means a communicative act cannot be viewed as an isolated entity. Different communicative acts are related to each other. "The different actions get their meaning from the business context: the roles and relations of the parties and the other business actions and the total *action logic* of the business transaction"

([Goldkuhl, 1996]). Wittgenstein's concept of 'language game' ([Wittgenstein, 1958]) is useful to describe such circumstances, since it describes communication as actions performed and interpreted according to different intersubjective rules. There are rules for single acts, relationships between different acts, and an institutionalized communication (consisting of different acts) as a whole. A business transaction is seen not only as a language game, but as an *action game*, since it involves both communicative and material actions. Material actions, e.g. the delivery and transportation of physical goods, have a communicative character, they are in itself informative of the delivery, but are not reduced to only communication. By this, Goldkuhl stresses that usually communicative acts are parts of an integrated wholeness of different actions.

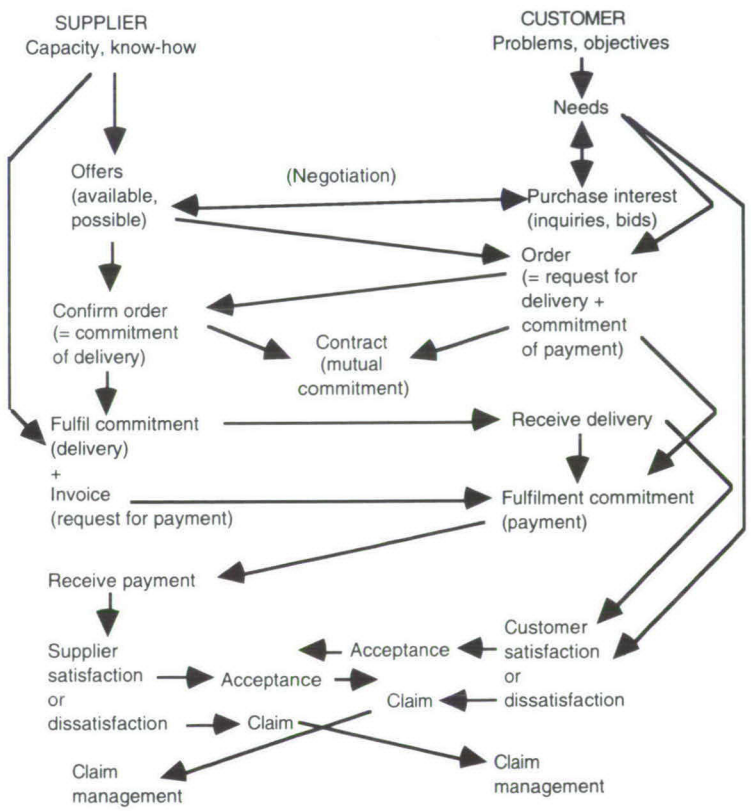


figure 3.2. Business as Action generic business framework
(after [Goldkuhl, 1996], fig. 3.1)

The *Business as Action game Theory* (BAT, [Goldkuhl, 1995, 1996]) is a generic framework for business transactions between actors. It describes the roles of two actors (a supplier selling, and a customer buying products (material goods or immaterial services, or a combination of both)) and their communicative and material actions that build up a

business logic. It is strongly influenced by the Language-Action Perspective and therefore communications of a business transaction are not mere information transfer, but truly communicative actions building up business relations between the parties. Figure 3.2 gives a graphical representation of the business logic framework.

Fundamental in this approach is the concept of *commitment*. A commitment is something that one must *fulfil*. From the work of Searle we learned that when someone performs a speech act he or she commits to what he/she is saying. As [Winograd and Flores, 1986] state it is "the specification of the meaning in terms of commitment entered into by speaker and hearer by virtue of taking part in the conversation", and it defines "the possibilities for what a speaker can do with an utterance, the possibilities for how words can be related to the world". A business relation and cooperation can only commence after the actors have come to an agreement and made a commitment to one another that they will indeed cooperate on the agreed terms. It is a mutual promise to the future conduct of the actors involved. The commitments are laid down in a *contract*, which is the central construct of the theory. Another basic principle steering the action game is the *business transaction*: "the business transaction is an interchange process between supplier and customer and it involves the creation and sustainment of business relations" ([Goldkuhl, 1995]).

Goldkuhl identifies four phases in this interchange process ([Goldkuhl, 1995, 1996]):

1) *proposal phase*. This in turn can be divided into three sub phases:
   a) business identification phase: on the supplier side, the development of offers (and corresponding products) together with the identification of potential market and customers; on the customer side, identification of problems and needs leading to purchase interest;
   b) exposure and contact search phase: (supplier side) the offers (capacity to sell actual products) must be communicated (exposured) to potential customers (search for contact); (customer side) the customer can try to get into contact with potential suppliers;
   c) contact establishment and negotiation phase: after finding each other the supplier and customer must establish contact and exchange their business possibilities and expectations (offers and inquiries).

   There are certain action conditions necessary for establishing a business transaction. There must be an *offer* by the supplier and an *order* by the customer. The supplier's offer is based on a *capacity* and *know-how* to produce and sell offered products. The customer's order is based on some *needs*, which in turn can be based on problems and objectives within the customer, and a purchase interest in the supplier's offer. Often there is a *negotiation* process between supplier and customer, including for instance a bidding with offers and counter-offers, or more complex, an elicitation of customer needs and also an investigation of the available and possible supplier product capacity and know-how. The negotiation ends by a definitive rejection of the offer or an acceptance of an offer (an order), in the latter case resulting in a contract between supplier and customer expressing the mutual commitments.

2) *commitment (contractual) phase.* This phase consists of the communicative actions of setting up the contract (the order and confirmation);

   The basic notion in BAT is the *contract*. The offer and order together form a contract. Such a contract is a mutual commitment, i.e. a supplier commitment of delivery and a customer request for delivery and commitment of payment, and possibly other commitments of other related business conditions.

3) *fulfilment phase.* This phase includes on the supplier side the delivery of goods and sending of invoice, and on the customer side the payment;

   This part of the business process involves *fulfilment of commitments* (such as delivery and payment) and related communicative action like invoicing (which is a request of customer fulfilment of the payment commitment).

4) *completion (acceptance/claim) phase.*

   In the final phase the delivered products can be accepted by the customer, leading to *customer satisfaction or dissatisfaction.* On the supplier's side the satisfaction consists of acceptance of the fulfilments of the customer, e.g., payment. In business settings this is a necessity for carrying on the business activity! The supplier satisfaction can also be a learning effect in performing business activities, improving know-how of the supplying organization. If the customer is not satisfied (i.e. there is perceived difference between the supplier's commitments and fulfilments) there might be claims towards the supplier. If the supplier does not get paid or is otherwise dissatisfied, actions for a claim can be undertaken. Claims are handled by the claim management at either side. It should be noted that no specific messages or actions for the satisfaction stage are given in the model. This is perhaps because satisfaction is often implicit, and only dissatisfaction leads to communicative acts, such as appeals to warrant.

It is important to notice that the pattern of different business acts is ordered in a generic pattern. In specific business transactions the order can be altered, e.g., there might be payment in advance. Also, the different acts do not need all to be made explicit. In many cases there will be no explicit writing of a contract, the order and its confirmation can together form the contract, or it can be oral mutual commitments, followed by a hand shake. Or in more simpler cases, like buying a newspaper, the contract is made implicit in the order process, or made implicit in the fulfilment of commitments (taking out a newspaper and putting money on the counter). In simple cases some of the generic acts within the business transaction are made implicit and taken for granted (but nevertheless are made, either by convention, or higher level contract, see chapter 5). This is also a way of decreasing the transaction costs on both sides.

In general it can be said that the doing-tasks of a contract are actions of two types: physical and linguistic actions. E.g., physical actions in the performance of a contract include the transport of goods. Linguistic actions are performative acts, e.g. the formation of a contract and the offer, acceptance, retraction etc. Physical actions can also be part of the contract formation phase, e.g. deliverance of goods means the acceptance of the goods if no other acceptance communication is made. Also linguistic actions may be part of the contract performance, e.g., an investment broker may buy/sell stocks for investors ([Lee, 1996]).

In my view the Business as Action game Theory gives the best description of business oriented communication, and is therefore adopted for the research presented in this thesis. However, I feel that the BAT not only is applicable in business settings, but in every situation where one agent (system or person) needs a service from another agent, even if these agents belong to the same organization or organizational unit, i.e. in every communication a supplier and customer role can be distinguished. Also, every communication can be seen as consisting of the four phases of communication: negotiation, contracting, fulfilment and satisfaction. As indicated above not all phases have to be explict, especially within organizations there will exist pre-arranged contracts (or agreements) on how to provide the service and how to fulfil the commitments following from it. But providing a framework in which it is possible to specify all phases gives more meaning to the communication process and possibilities for changing it.

The adaptation of the framework made in this thesis is that the contract not only describes the order and acknowledgement of it, but also (and especially) the following phases: fulfilment and satisfaction. This is in accordance with real-life contracts that describe the role of both supplier and customer regarding the business transaction. The contracts as specified in this thesis also include clauses about what should happen if the customer is not satisfied after using the product (warranty), or if something goes wrong along the way (e.g., not obeying delivery times or conditions). Also special contracts can be set up that regulate the negotiation process as is described in section 5.2.2.

In the communication framework proposed in this thesis we also specify relationships between the agents that influence the type of contract that can be negotiated between agents. Three relation types can be distinguished: power, authority and charity (for relations between peers). Only if there exists a peer relationship between agents the symmetric contracts can be used. This aspect is discussed in more detail in chapter 6.

The approach in this thesis goes further than BAT in respect that a communication framework describing transactions, contracts but also tasks of the agents is given (including a specification language, chapter 5) and a formal framework for these concepts is provided (chapter 6). Furthermore an implementable agent architecture is described (chapter 4). Finally a modelling methodology is given (chapter 7) that uses different modelling techniques. Goldkuhl only provides the theory, he does not give an underlying formal framework. In [Goldkuhl, 1996] he proposes to use SIMMs for modelling the process. Comments on this are made in section 7.5.2.

What makes the agent system as proposed here different from most other cooperating systems described in the literature is that also in the fulfilment phase of communication the agents are autonomous. They can at any moment decide not to honour the contract they agreed upon. Of course this will have repercussions, but an agent might decide they are less damaging to it than following the contract. So, even though an agent agreed upon disclosing some information to another agent it might decide later to withhold some of it because it is too secret.

3.1.3. COMPARISON WITH OTHER FRAMEWORKS

This model can be distinguished from other BPR models on three accounts:

1) the focus on *interactivity*, rather than input-output transformations. A business process is not only an input-output transformation process (as defined e.g. in [Hammer and Champy, 1993]). Its interchange character should be emphasized. When describing and redesigning business processes different business acts (of generic communicative character) should be recognized.
2) its *generic communicative logic* . E.g., once an order is defined as a mutual obligation (to deliver and to pay, respectively), the fulfilment is a logical consequence, as is the (undesirable) possibility of non-fulfilment. The commitment must be mutual because of the very nature of a business transaction. It is also quite natural that mutual commitments can only be established after a stage of negotiation.
3) this model with clear emphasis on the actor roles of both supplier and customer is called a *two-directions co-action model*, since it emphasizes the mutual character of the business transaction. It is an exchange process with mutual commitments, fulfilments, and satisfaction. Goldkuhl stresses that it should not be conceived as a denial of the need for customer focus. However, the need for the *mutual* satisfaction of the supplier and the customer should not be disgarded. It is important to let the supplier be visible too. To disregard the commitments of the supplier is to cut away necessary parts of the business transaction, and hence, reduce it from its generic business character. "The business transaction is built from the business interchange relations and the different business acts which must be formed in a communicative congruent pattern"[ibid.].

In comparison, Action Workflow for instance is rather one-sided, since it only emphasizes the customer role. As described above, a contract can be "symmetric", but also establish an asymmetric power relationship, in the same way as an employer can contract a new employee. We then arrive in a situation in which the critique on Action Workflow - that it is rather one-sided in its customer(initiator) emphasis - is no longer valid. The one-sided Action Workflow loops fit well in an organizational setting with existing power and authorization relationships, whereas the symmetric business process model fits well in the free market context of autonomous negotiating agents. So both models have their value, depending on the context.

The four phases: proposal, commitment (contractual), fulfilment, and completion (acceptance/claim) are rather similar to the Action Workflow model which also has four phases: preparation, negotiation, performance, and acceptance. The differences are in the division in the first two phases and the different definition of acceptance in the fourth phase, which in Action Workflow is defined as only customer acceptance/satisfaction.

Also two major differences between the infological and communicative action approach to systems development can be distinguished.

First, the infological approach distinguishes a managing and an operational system.

The managing system gives directive information (control signals) to the operational system and from the operation system status information is fed back to the managing system. This is a closed loop approach as in BAT and Action Workflow, however with the assumption that there is only one managing system. This means that in an organization only the management is responsible. In today's business process models these loops reoccur at several places. This is a big difference between organizations in the seventies and nineties concerning organisational ideas and decentralised responsibility.

A second difference between the infological and communication oriented approach is that in the latter there is room for negotiation. An order is more than a trigger with which a receiver can be put to work. It is possible the receiver refuses. The sender of the directive has to try to convince him. According to [Habermas, 1981], communication always requires a common background, or 'life world' of the participants, which consists of common knowledge, shared institutions and mutually known competences. Since the life world is at the same time the basis for and the outcome of the communicative action, none of its parts is principally excluded from negotiation and debate.

The distinctions made by Habermas gives us the means to distinguish between different information and communication theories. It seems the classical organisation and information theories are based on the theory of strategic action. Also in other closed-loop approaches it is not clear what stance is taken (strategic or communicative). Although they do distinguish negotiation, as said above the basic transaction is rather one sided and aimed at the satisfaction of the customer. The satisfaction of the supplier is not taken into account. Here a true communicative action perspective is taken, emphasizing the roles of both supplier and customer and the need for mutual understanding and commitment.

3.2. RUNNING EXAMPLE

The example that is used throughout this thesis is the well-known example from interoperable transaction literature: planning a business trip (figure 3.3).

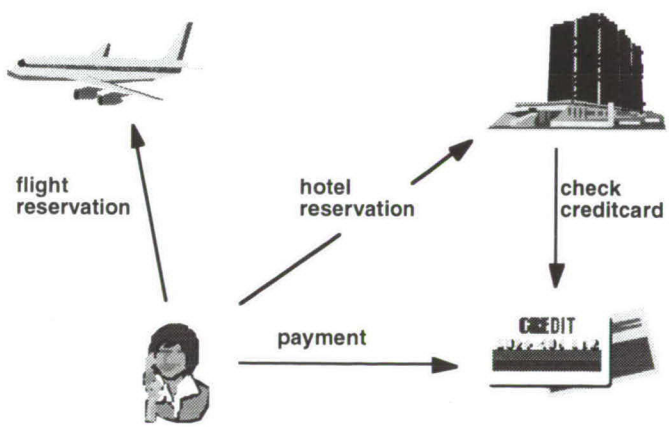


figure 3.3. Planning a business trip

The actor at the left bottom has the task to plan a business trip. He/she has to communicate with an airline company about ticket reservation and with a hotel about hotel reservation. Furthermore he/she must pay for both ticket and hotel, e.g. by using a credit card. In the process of paying, the hotel must check with the credit card company for authorization. Many aspects of the framework presented above and discussed in more detail in the next chapters can be illustrated with the business processes of this example.

The interaction between actors can most easily be seen in the cases where a particular service is rendered from one actor to the other. A (prototypical) example of such a service is illustrated in the following picture (which contains a simplified version of the framework presented in figure 3.2).

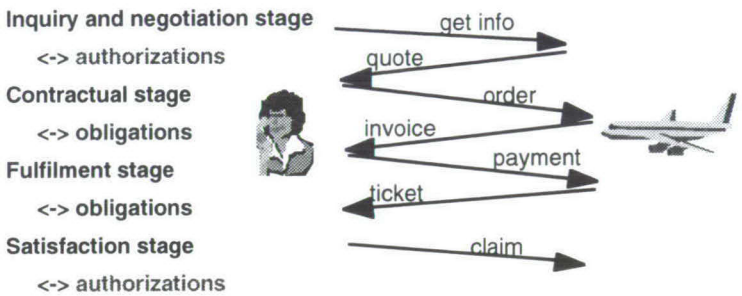


figure 3.4. Communication in a (prototypical) business procedure

We recognize the four phases of communication. The first phase is the inquiry about services and negotiation about the terms of the contract. In this phase authorizations can be established on the basis of which actions can be performed in the following phases. In figure 3.4 this corresponds to the requesting and sending of a quotation (proposal). Sending a quotation implies authorizing the customer to order some products on some specified conditions. The second phase is the contractual phase, the establishment of the contract, i.e. the obligations (and authorizations) pointing out mutual commitments. In the example this is included in the order, which implies an acceptance of the quotation, and also the sending of the invoice that creates an obligation for the client to pay. The third phase is the fulfilment of the contract, i.e. following a protocol according to the terms agreed upon in the contract, trying to live up to the obligations. In the example this corresponds to the delivery of the ticket and payment sequence (in non-specific order). The last phase is the completion phase. This contains the acceptance of both partners of the fulfilment of the commitments by the other. This phase usually describes the authorizations on basis of which the parties can put forward claims in case of dissatisfaction (but might also include rewards in case of great satisfaction). Often there are no explicit messages in this phase. In the example above the client can, for instance, claim some compensation from the airline company if the flights get cancelled.

In the remainder of this thesis the focus is on such ,what are called, contracting discourses.

CHAPTER 4
CIA ARCHITECTURE

In the introduction was described how information systems in my view evolve into Cooperative Information Agents. Also was pointed out that providing an architecture for such systems based on a theory of communicative action can support the design of such systems. In this chapter this architecture is given. First the basic terminology is given on which the communicative action theory and parts of the architecture are based. In section 2 the knowledge bases are described that contain the knowledge about the information context and information content, the agent uses for functioning. How these are operated upon by the agent-interpreter and its functional components is described in section 3.

4.1. BASIC TERMINOLOGY

In section 1.1.3 an agent was already defined as "an autonomous computational entity with certain tasks and capabilities (action it can perform) that communicates with other agents by means of messages and whose behaviour is not predefined but based on commitments to other agents". Based on what we learned from the Language-Action Perspective and agent theory we can now give a more detailed description of what an agent and its behaviour are (the structure follows the agent definition of the Open Distributed Processing group ([ODP, 1992]):

- an agent is an application independent concept used from an organizational viewpoint of businesses, or business processes. This means that an agent can help to define organizational work.
- an agent is an object which can initiate a 'performative' action (as opposed to an artefact which does not initiate actions). It does not only reply to requests from other agents, but it should be able to make a decision to execute an action. Since an agent is an object, it encapsulates a state and offers an interface to other agents.
- a 'performative' action is an action that changes policy for one or more objects involved in the action. Performative actions that can be distinguished consist of private and communicative actions according to the language-action framework:

- an agent incurs an obligation to another one
- an agent fulfils an obligation to another one
- an agent requires permission from another agent to perform some action
- an agent is forbidden (and forbids) to perform an action
- an agent performs some private (physical) action.

• Obligation, and authorizations are types of policies. A policy is a prescriptive relation between two or more agents which establishes a norm for the correct behaviour of these agents. Agents deal with such policies, so it is performing actions according to an established policy. It has to deal with its own obligations (commitments it made) and authorizations, and contractual relationships.

It is important that an agent can prove its behaviour is compliant to the policies that are specified (e.g., for users to trust their agents), and therefore needs a formal framework. The CIA framework proposed is based on the theory of communicative action and uses dynamic deontic logic extended with illocutionary concepts for its formal logical system (chapter 6).

Before a description of the architecture can be given first the different levels that can be distinguish in the communication framework must be explained.

The two basic activities that an agent can perform are:

• An *action* is a non-communicative activity that is performed by an agent. It is treated as an atom and not further analyzed.

• The atomic communicative action is the *message*. A message is a speech act that describes the illocutionary force of the message (whether it is a directive, assertive, commissive or declarative) together with its authorization claim (be it power, authority or charity), and a content (a proposition or action).

The model consists of three levels, which are shown (from top to bottom) in figure 4.1. The figure gives from left to right the name, discriminating parts and how they are related, e.g., a contract is described by 'deontic states', and 'transactions' describe the transition between the states. The concepts are explained in detail below.

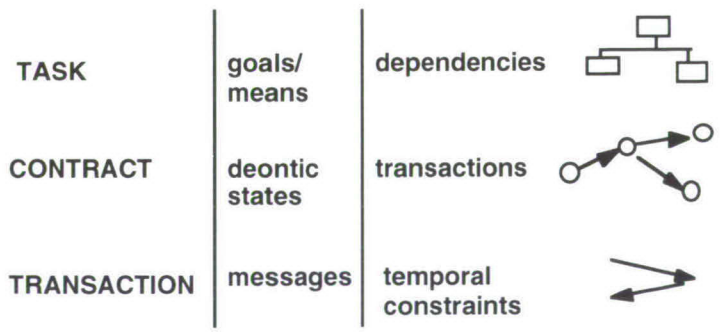


figure 4.1. Three-level communication modelling

CIA behaviour can be described by means of *interoperable transactions* defined as a logical unit of work (on communication and authorization level, not on technical database transaction level), involving different autonomous systems. Interoperable systems cooperate by means of message passing. The specification of an interoperable transaction consists of a set of communicating subjects (called agents), elementary actions, the possible message types provided by each agent to support its role, and temporal constraints on the synchronisation of the actions (communicative and non-communicative). Furthermore a deadline before which the transaction has to end can be specified. Here it is sufficient to mention that a message (speech act) can be implemented by embedding it into a traditional ACID transaction ([Gray, 1981], [Özsu and Valduriez, 1991]), and that their execution is serializable. Transactions, as defined here, however, are not database transactions and cannot be seen as an ACID unit of work. Especially failure handling is done differently and handled at the contract and task level. Transactions are described in more detail in section 5.1.

The specification of interoperable transactions needs structure for the sake of complexity reduction and reusability. Flexibility and extensibility in specification is even more important in a cooperating IS than in traditional isolated ISs, because the transactions typically span several organizations. To achieve a desirable level of flexibility of interoperable transaction specification, I propose:

- the use of an additional concept, called *contract*, to specify and manage the failure due to agents not honouring their obligations in interoperable transactions;
- the separation of transactions and contracts from the agent's task.

Figure 4.2 gives a graphical representation of the relation between the three concepts task, contract and transaction.

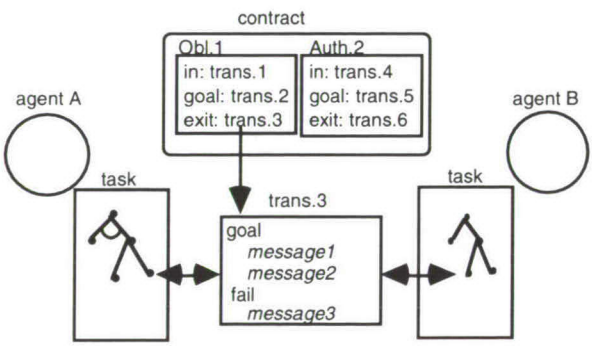


figure 4.2. Tasks, contracts and transactions

A *task* is a meaningful unit of work assigned to an agent. A task is specified as having a goal, it can be decomposed into subtasks, and can include alternative means for achieving the goal. Furthermore a deadline before which the goal should be reached, and

constraints on the execution of the subtasks (dependencies) can be specified. Elementary subtasks are either executed by the agent as an internal procedure (an action) or involves initiating communicative transactions with another agent. Tasks are managed by the Task Manager of the agent (see section 4.4).

An important part of the task specification concerns the failure handling method, or compensation and contingency. As part of the task specification the dependencies between results created by subtasks can be given. E.g., in the case of a business trip it can be specified that a hotel reservation should be cancelled (compensating action) if a flight reservation (made in a separate subtask) is cancelled. Also update relations can be given, e.g. if the flight is delayed, the hotel reservation should be updated. As a separate part a contingency plan can be given, describing actions that can be taken in order to reach the goal if a subtask fails, e.g., book a flight with a different airline company if the original flight is cancelled. Only if this fails dependent results (such as the hotel reservation) should be compensated (rolled-back). In the case of a business trip one might also specify alternatives, e.g., that the destination can be reached also by train instead of by airplane.

Tasks are described in more detail in section 5.3.

A *contract* is specified as a set of deontic states. It specifies obligations and authorizations between different parties about services provided to each other. Following the business logic framework a contract describes the interaction (possible transactions), together with the obligations and authorizations, but also the consequences if either agent does not adhere to its obligations.

For instance, in the case of the business trip example, the ticket-order contract can specify the obligation for the airline company to answer to an inquiry by the travel agency, either by giving a quotation for a ticket or a refusal. It furthermore specifies the obligation for the airline company to deliver the ticket if the travel agency orders it under the conditions specified in the quotation. The contract also specifies the obligation of the travel agency to pay for the ticket (within a certain time) after it has ordered it.

Based on the agreement reached between the airline company and the travel agency, the contract can also specify what should happen if the ticket is not delivered in time, or if the travel agency does not pay in time, or what should happen if the flight for which the ticket is issued is cancelled. In the last case, the contract may specify that a fine should be paid by the airline company. But, following the business logic concerning satisfaction, the contract can also specify what course of action to follow if the travel agency is not satisfied with the delivered ticket.

This last note also means contracts have a longer lifespan than tasks. After the product (e.g., the ticket) is received the goal is reached for the customer (travel agency) and the task succeeds, however the contract might have options that describe the course of action if the customer becomes dissatisfied at a later stage or if the result is invalidated. So reaching a goal does not mean the contract can be forgotten. Invalidation of established results also has the effect that the goal becomes invalidated and the contingency plan should be triggered.

Contracts are described in more detail in section 5.2.

The task specification and updates thereof concern *the agent in question only*, they are the responsibility of the user or agent only, whereas changes in the possible transactions (involving other agents) can only be made by *consent of the other agent*. For this reason, a distinction is made between task and contract, where the contract corresponds to the agreements between the two agents and the task draws on this potential for fulfilling an agent's individual goal.

As said above, contracts specify the obligations of agents about services provided to each other, it is a mutual promise to the future conduct of the agents involved. If an obligation is not fulfilled (e.g. the order is not payed), it is possible to reason over this violation and take a remedial action without forcing the whole task to abort. It is the job of a Contract Manager to impose these violation policies. It would complicate the task specifications and lower the reusability if the communication about obligation violation is included in the task. It is a matter concerning the agreements *between* the agents and should therefore be part of the contract specification. The agent's Task Manager should only be responsible in ensuring that a task is brought to its goal, not how violation of commitments is being dealt with. In this way, the process is more reactive to failures.

Working out the business trip example further, the task of the travel agency might contain subtasks of flight-reservation and hotel-reservation and constraints between them (e.g. hotel-reservation after flight-reservation, meaning we want to be sure to have a flight to the destination before booking a hotel there). The subtask flight-reservation initiates a transaction between the agent and an airline company. The interaction is part of the contract between the travel agency and the airline describing the procedure of ordering tickets (the result of a flight-reservation). The selling of tickets can be part of the task of the airline company. If, at a later stage, the flight gets cancelled the result achieved becomes invalidated. In the contract can be specified that the airline in that case is obliged to pay the travel agency a fine. In the contingency plan of booking a business trip triggered by the violation can be specified that the hotel-reservation should be cancelled too (the hotel booking task is started after the flight-reservation task succeeded before). Also, since the result is invalidated it may be retried, either by booking another flight with the same airline, or a different one, or choosing an alternative, e.g. travel by train.

4.2. ARCHITECTURE

Figure 4.3 gives a graphical representation of the basic architecture of a CIA. In the next subsections the knowledge bases, and the functional components of the interpreter are explained in more detail. Here first an informal description of the components is given.

As stated above an agent is typically a piece of software that has certain capabilities, actions that it can perform (described as *services*) that can be requested for by other agents, and certain *tasks* and responsibilities (delegated to it by a human agent). The actions can be database actions such as updates, or communicative (transactions), such as providing some piece of information, or sending a request to another agent. An important component is the *contracts*-base, which contains standard contracts or contracts that resulted from negotiation with other agents, and that govern the interaction between the

agents. The interaction itself is described by *transactions*. Each agent also has an *agenda* containing the actions to be performed by the agent, instantly or at some designated time. The *Lexicon* is the system that stores and manages the terminology of the domain, it corresponds to the Conceptual Model describing a particular Universe of Discourse and also the Environment of Discourse (e.g. communication rules). The instantiation of this model is stored in the *database*.

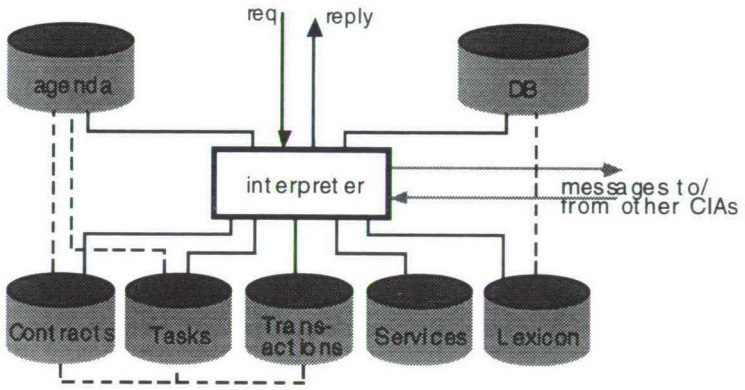


figure 4.3. CIA architecture

The following section describes the knowledge bases of the CIA in more detail, and in section 4.4 the interpreter (and its functional components) is described.

4.3. KNOWLEDGE BASES

A CIA has local autonomy and its behaviour is not predefined but based on commitments to other agents (created either statically at design time or dynamically during execution). In order to be able to function properly the CIA needs knowledge about the domain. Taking a LAP this not only includes knowledge about the objects in the domain (UoD) but also knowledge about the communication processes and structures in the discourse (EoD). Three kinds of knowledge can be distinguished:

1. contractual knowledge
   Knowledge about tasks, contracts, and authorizations and obligations resulting from them, organized and managed in the agenda. It also includes some *self* knowledge and knowledge of peer CIAs and their services. The knowledge bases containing this knowledge are Tasks, Contracts, Agenda and Services.
2. semantic knowledge
   Knowledge about the object types dealt with. Their structure is defined in the Lexicon, and the instantiations are recorded in a database. Since both UoD and EoD are modelled, it includes knowledge about the speech acts and meaning of the illocutions made (knowledge about the communication structures).

3. communication knowledge
   This constitutes pragmatic knowledge about communicating. When the CIA receives messages, e.g. a request, it should somehow answer, either a result, based on his knowledge or "don't know", or a discussion on the appropriateness and validity of the request. This knowledge does not have a separate knowledge base in the architecture, but is partly contained in the Services knowledge base and the Communication Manager. The CIA should also know how to send messages to (initiating transactions with) other agents. This is captured in the transactions-base.

4.3.1. AGENDA

Intelligent agents have an agenda that is monitored continuously to decide what action is to be performed. Formally, an agenda is a set of deontic temporal constraints. It is deontic since the agenda specifies what the agent should do (its obligations). It also has temporal properties since the obligation is usually to be performed before a certain deadline, or as soon as possible. An example of an agenda item might be the obligation to pay $635,- for the ticket before next week. In chapter 6 the logic that facilitates reasoning about obligations and deadlines is presented. The logic is an extension of deontic dynamic logic, in which one can specify that an obligation starts at a certain time or event, that it must be done immediately, as soon as possible, before a deadline, or periodically.

We assume the agenda is not fixed but can be manipulated by the agent. The agent can add new obligations to the agenda (typically done on the request of another agent) and can reason about them. Obligations can be the result of the (sub)task of the agent, describing what (private) actions have to be performed or what transactions have to be initiated (e.g., ask the airline company for a flight-schedule), but can also result from the contract (the mutual obligations agreed upon, e.g., payment of the bill for the ticket within two weeks) or the transactions that have to be followed according to the contract (e.g., acknowledge the receipt of the ticket). A special agenda-item, put on the agenda by the contract manager, is a check whether the other party has fulfilled its commitments and thereby followed the contract (see section 4.4.1).

The agent can also remove items from the agenda by performing actions or it may violate an obligation. In the latter case it usually has to perform some compensatory action (e.g., payment of a fine).

Taken from this viewpoint the agenda defines the normative space, describing all temporal deontic aspects that influence the behaviour of the agent. This differs from most other agent approaches that use the agenda only to record tasks the agent has to perform.

In section 5.3.2.1 the structure of the agenda is described in more detail. Here it suffices to say that with every agenda item the specific subtask or contract that created it is recorded. This is important since the agent can be engaged in interaction with several agents at the same time. Also, for planning purposes the agenda items can be extended with priorities, representing preferences in goal satisfaction.

4.3.2. CONTRACTS

The Contracts-base not only contains the contracts between the agent and other agents, but also contains some standard contracts, describing a preferred way of doing business, that the agent can offer to others that want to make use of the services the agent provides.

Also contracts about previous interactions are stored, as they can be reused to conduct the same business again (possibly changing some of the conditions). As is described in section 5.2.2 the process of setting up a contract with another agent (by negotiating) is in itself a service the agent can offer. Existing contracts are then instantiations of a standard contract, or perhaps an instantiation of a subtype of a standard contract. Sometimes conditions have to be changed, or if the other party does not want to work according to a standard contract a new one has to be set up. The Contracts-base therefore also is the place to store knowledge about negotiating processes, that are used in setting up, or changing a contract. The contracts are consulted and managed by the Contract Manager. Contracts are specified in the contract specification language, described in section 5.2.4.

4.3.3. TASKS

The Tasks-base contains all tasks of the agent. A task has a goal, can be decomposed into subtasks, and describes the actions that have to be performed to achieve the goal. Elementary tasks are either executed by the agent as an internal procedure or involves communication transactions. It can also specify (temporal) constraints, and precedence relations between the subtasks. The task specification thereby becomes a plan for achieving the goal. The preferred order of execution of subtasks (partial plans) can be stored separately for reuse purposes. More importantly, one can specify alternatives for subtasks that can be tried if the original subtask cannot be satisfied. A task specification can also include a contingency plan that describes what should happen if an already committed result becomes invalidated e.g. by cancellation. The tasks together with the task specification language are described in section 5.3.

4.3.4. TRANSACTIONS

The Transactions-base is an auxiliary database that contains all the transaction descriptions, i.e. the messages and the constraints between them. Both the contracts and tasks refer to the transactions in the Transaction-base.

4.3.5. SERVICES

The Services-base consists of four parts:

1. knowledge about the services the agent can provide to others, and the conditions under which these can be provided. A service is a publicly accessible interface of the agent, comparable to a method-interface in OO-programming. A supplier agent can for instance provide the service of giving an offer for a specific

product. Another service can be the offering of a standard contract according to which two (or more) agents can conduct their business.

2. the 'private' services or elementary actions. This is a specification of the private actions the agent can perform. These do not have to be the same as the services provided to other agents. They are usually more low level. E.g., an agent can offer to provide hotel reservation information to others, however the internal database actions to get that information are private. This is especially the case if the agent acts as a broker. It offers services to others, but for the execution of them, it needs the knowledge and capabilities of other agents.
3. knowledge about services provided by other agents. E.g., in the case where one wants to conduct business with a specific travel agency (maybe because one has done business with it before) and knows that it can offer good alternatives for flight and hotel reservation. This information can be used in the contact-search phase as it was called in section 3.1.2.
4. knowledge about the authorizations, or what may or may not be done. E.g., the authority to order another agent to book a business trip.

4.3.6. DATABASE

The Database contains the population of the domain model and can be used to store intermediate results of calculations. As was described in section 1.1.3.1 the CIA can also contain an existing database that has been encapsulated (the CIA functions as a wrapper for the original database, extending it with the functionality of the CIA). Following [Shoham, 1993] I call this process 'agentifying'.

4.3.7. LEXICON

As described above, the Lexicon is the system that stores and manages the terminology of a certain domain. It describes the information of the discourse including the message types and the static and dynamic structure of the domain. The Lexicon can be compared with a data dictionary in traditional ISs.

In the remainder of this subsection the Lexicon is described in more detail. While the other knowledge bases are implemented in the prototype the Lexicon is not (yet). The following is a description of the Lexicon as it is foreseen.

From the linguistically motivated point of view taken in this thesis the Lexicon is interesting. In general a Lexicon has two major functions. First, it explicitly represents the *mutual* understanding about a certain concept of all agents involved. This can be used in the initial phase (negotiation) of setting up the contract. The second major function of a Lexicon is the input it can provide to the software development process. The more formal structure the Lexicon exhibits, the more can be derived from it in terms of entity types, relationships, actions etc. One research project in Tilburg University investigates to what extent NIAM object models can be derived automatically from a lexical specification.

In [Weigand, 1990] a linguistic approach is presented in which the lexical definition of concepts as well as the lexical structures in the Lexicon are based on linguistic primitives. The Lexicon defines the possible predicate frames describing the domain. I.e. grammatical information[1], such as gender, word category, stem, etc.; structural information, in particular the taxonomic relation, designating subsumption or subtyping, but also semantic sets and pre- and postconditions for dynamic predicates; and conceptual information (predicate schemata, i.e. 'stereotypes', including essential but not necessary characteristics of a concept). This information can be specified in for instance the linguistic representation formalism Functional Grammar ([Dik, 1978, 1989]).

In such a Lexicon, no distinction is made between entities, relationships, actions, attributes, identifiers, in so far as all of them have a lexical entry. Following linguistic theory, the concepts are organized in a taxonomy and around prototypes. The higher levels of this taxonomy form a basic ontology, including primitive concepts like 'entity', 'event', 'state'. The lexical entry abstracts from the various word forms and inflections of a certain word, as in normal dictionaries. In the case of complex concepts, such as actions, the lexical entry includes a frame containing the roles of the participants, such as 'agent', and 'recipient'. The definition of a concept is in principle simply the (natural language) dictionary entry. It can be parsed and stored internally in a linguistic representation formalism like FG, which allows formal treatment. Moreover, lexical research has shown that definitions typically make use of a small array of basic structures, such as 'isa', 'partof' relation and 'purpose'. This allows for even more formalization and hence computational processing, while the definition can still be expressed in natural language.

An example of some lexicon definitions is the following. The italicised noun phrase indicates a taxonomic link to a superconcept, arguments are given between parentheses:

ticket: a printed proof or *reservation* for a certain *flight*
registration number(ticket): a *string* of 4 alphanumeric characters and 2 digits uniquely identifying the *reservation* for which the ticket is made
serial number(ticket): a *string* of 12 digits uniquely identifying a ticket of a certain *airline company*
issuer(ticket): the *airline company* that issued the ticket with a specific serial number

Below is an example in which the semantic structure of a concept is explicit. *sell* is a transfer action with three semantic roles (the object has an empty label, ag stands for agent and rec for recipient). It describes pre- and postconditions using predicates from the same Lexicon, such as *own*. The incondition says that the action includes a payment action of the recipient.

sell(ag X:human)(P:thing)(rec Y:human)
`isa` transfer
`pre` own(ag X)(P), price(P) = D
`post` own(ag Y)(P)
`inc` pay(ag Y)(D)(rec X)

[1] Note that lexical definitions should not be considered as exhaustive characterizations of a concept. A definition is made relative to a certain context. Within the context, the definitions must differentiate different concepts, and provide a *basis* for mutual understanding.

In circumscribing the terminology for a particular application domain, the knowledge engineer might draw on available terminologies for the generic domain. Such generic terminologies might draw in turn on more general dictionaries. Hence, it seems useful to organize the Lexicon as a collection of related sublexicons. A *Lexicon Management System* is a system that can handle multiple lexicons. In this way, it is possible to set up application architectures in a consistent way.

We assume that the Lexicon is filled first by a knowledge engineer (backed by a domain expert). Dynamic negotiations between CIAs about concepts is still an idea to be worked out in the future. However, when more ontologies, ISO standards and domain lexicons become available, it is possible for negotiating CIAs to set up a mutual understanding by making a reference to such a given set of concepts. It could also use a bilingual (certified) Lexicon that translates Dutch business concepts to French, for example. It should be clear to which definitions both CIAs commit themselves .

4.3.8. COMPARISON

When we compare this architecture with the standard concepts knowledge, belief and intention from the agent literature, it can be observed that the knowledge of the agent is contained in both the Lexicon (domain and world knowledge) and the Services-base (the self-knowledge), the beliefs are contained in the Database (which contains all instances of the agent's domain knowledge) and partly in the Services-base (the beliefs about the possibilities of others). The intentions are related to the items on the agenda, which is described in more detail in section 5.3.2.1.

The framework can be related to Habermas' theory of communicative action (section 2.1.1.4). When involved in communicative action the agents are oriented towards mutual agreement, and it is considered essential that they achieve a common definition of the situation in which they find themselves. The CIA framework materializes this common definition in the form of the Contract (agreement on authorizations and obligations) and the Lexicon (agreement on concepts). With regard to the contents of a speech act, Habermas distinguishes between three worlds of reference: the objective, social, and subjective world. Although CIAs do not take part in the human discourse and we do not want to blur the distinction between the human and the artificial, the same principles of rationality apply. A CIA also refers to an objective world (Lexicon), a social world (Contracts) and a subjective world (Tasks).

4.4. INTERPRETER

The interpreter or agent-engine (the central component in the architecture of figure 4.3) consists of a number of functional components responsible for each of its main activities (see figure 4.4): task management, contract management, communication and interaction management, and service execution. These components are explained in more detail in the following subsections.

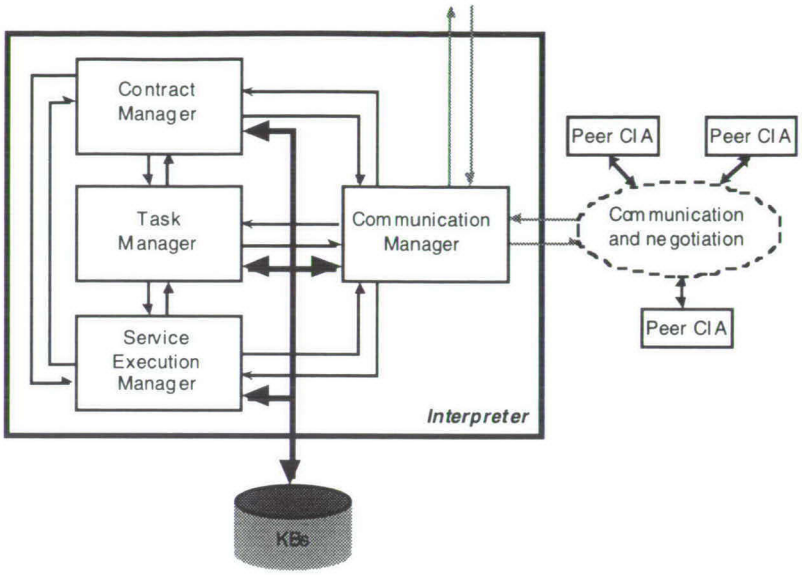


figure 4.4. Functional CIA architecture

Here we give an informal description of the operation of an agent concerning tasks, contracts and transactions. Figure 4.5 gives a graphical representation of the working, based on figure 4.2. The figure takes the viewpoint of agent A, agent B has a similar view (also a Task and Contract Manager). In the text the numbers between brackets, e.g. (2), correspond to the numbers in the figure.

An agent has a certain task that can be split up in subtasks that the agent tries to fulfil. In doing so the agent initiates the necessary transactions ((1)). In case of an unrecoverable failure ((2)) the Task Manager of the agent should be notified so that it can pursue an alternative, if necessary by rescheduling (sub)tasks.

The communication behaviour between the agents concerning some business relation and process is described by a contract. The contract also specifies what should happen in case of violation of one of the obligations or cancellation (cancel) by one of the agents (by notifying the Contract Manager), possibly leading to other obligations or authorizations ((3)) described by another transaction in the contract and triggering a 'contingency' plan ((4)), describing what should happen in order to get the subtask fulfilled, managed by the Task Manager.

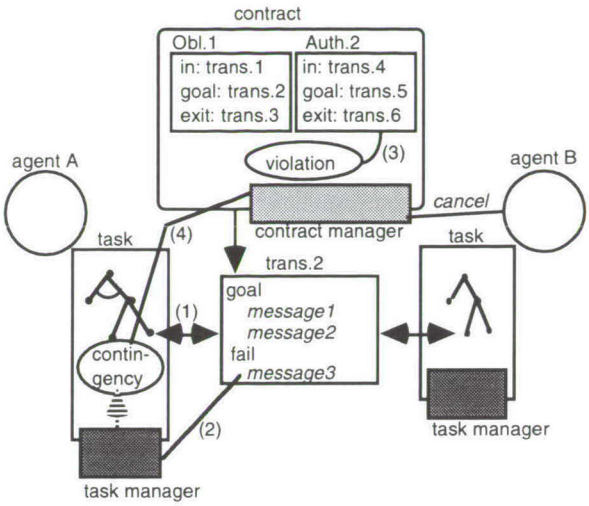


figure 4.5. Task and Contract Manager

4.4.1. CONTRACT MANAGER

The contracts are managed by the Contract Manager. The following gives an overview of the processes supported by the Contract Manager.

(a) creating new contracts, and adapting existing ones.

The agent can play two roles, either as customer (receiver of services) or supplier (provider of services). In the role of customer the Contract Manager can consult the services-base to check if it knows of a provider of the service required. If so it sends out a request for the service to that other agent (possibly including already some conditions under which it wants the service to be provided). If not it can see if it knows a broker agent that can find out who provides the service, or it can broadcast the request to all peer agents. The request can be answered with a refusal (from the directed agent or all agents) after which the Task Manager should be notified that the subtask can not be fulfilled, or with an offer (possibly with a standard contract offer). The agent can then try to negotiate some conditions of the contract (including making counter-offers). If both agents agree on the conditions the Contract Manager sends out the order, which should be acknowledged by the other agent. The above scheme follows the business logic (described in section 3.1), but different negotiations protocols can be implemented, see section 5.2.2.

In case the agent plays the supplier role and a request for a service comes in, the agent checks if there already exists a contract with the requesting agent about the service. If so, it replies to the request with an offer and this contract. If not, the agent checks whether there exists a standard contract for the requested service which is returned. Otherwise the agent replies with an offer (and possibly some conditions), and also offers the service of setting up a new contract.

Another situation that may occur is the adaptation of an existing contract. This usually concerns contracts the agents have been engaged in before, and that should be adapted to the new situation (this specific order). Normally the agents do not return on the mutual agreements reached earlier in the negotiation phase (unless that possibility is explicitly expressed in that phase of course). In this way the negotiation phase can be shortened (usually saving time and communication costs).

(b) transaction management, including contractual commitment.

The protocol that is specified is independent from applications, but in contrast to traditional communication protocols, captures the complete communication logic, not just a (ordered) set of messages. Aspects of faulty message delivery on the technical level do not have to be specified in the contracts themselves, e.g., what should happen if a message does not arrive through some hardware failure. Handling this kind of communication problems is typically a task for the underlying infrastructure and can be specified (generically) in that place. (However, if because of this failure the task cannot be completed, the Task Manager should be notified).

A contract describes authorized communication behaviour among the receivers and providers of services. In initiating transactions obligations are formed, based on the illocutionary force of the messages (speech acts). E.g., an authorized request yields an obligation for the receiver of the request to give an answer concerning the proposition of the message. If the receiver does not adhere to the obligation, it is the job of the Contract Manager to detect the violation and trigger the violation policies. This is done by consulting the contract with the other agent, and recording the obligations of other agents in connection with the CIA itself. For example, an order contract can specify that the airline company is obliged to deliver a ticket for a fixed price before the end of the month.

The contract also specifies that the travel agency has to pay the bill before a certain date. This is an obligation of the trave; agent, that of course should record and maintain it. But since the travel agent has to check if the other agent fulfilled the commitment (i.e. if it delivered the ticket before the date specified), the travel agent also has to record it. It therefore adds this check (with the appropriate deadline) to the agenda. It is then possible to undertake actions if the other agent violates the commitment and thereby the contract. Furthermore we want to be able to reason about what to do if something happens that is not supposed to happen (in the case of prohibitions). The contract may describe what should happen in case a violation occurs. E.g., in the case above, cancel the ticket, or asking for a fine. In case of a fine, this leads to new obligations of the client agent.

In case the agent is dependent on the fulfilment of the obligation for reaching a goal, violation of the obligation by the other agent means the goal cannot be reached through this transaction. In that case the Task Manager is notified, so it can pursue alternatives.

A reached goal state can become invalidated if the other party cancels a process, e.g. the airline company cancels a booked flight. In this case again the contract should be consulted what remedial actions should be taken. However, cancellation also means that any partial results becomes invalidated (e.g. the ticket is no longer valid). This means that also the Task Manager should be notified, triggering a contingency plan that can contain actions on how to revalidate the results (e.g. book a new flight).

c) claim management.

The final task of the Contract Manager concerns the final phase of business communication, the acceptance/claim phase (see section 3.1.2), dealing with satisfaction of the agents. Contracts live longer than tasks. Although a goal can be reached and the (sub)task is finished, at a later stage the customer (or supplier) can become dissatisfied with the service. For instance, if upon arrival a booked hotel is no good (not according to the conditions negotiated and laid down in the contract before) the customer can appeal to the warranty agreed upon.

The communication about the warranty is not setting up a new contract or performing a new subtask. It is directly related to the communication performed when reserving a room. Although the task of booking a hotel is finished, the contract still holds, and the supplier should take care of the problem and try to satisfy the customer (again). If this is not possible (or is not done by the supplier) the Task Manager should be notified that the (partial) result has been invalidated, triggering the contingency plan of the original task. In the case of the hotel reservation this now becomes a new task, since the customer is already at the place of destination and not finding a hotel is now a vital part (as it was not when trying to book a business-trip).

In section 5.2.2 the negotiation process is described in more detail.

4.4.2. TASK MANAGER

The Task Manager has three main roles:

a) scheduling and planning.

The Task Manager collects all the subtasks specified in the task specfication and puts them in the right order for execution. Based on the constraints, deadlines and priorities specified, a plan is formed and the subtasks (including the communication transactions and private actions) are put on the agenda and called one by one. This step continues recursively.

b) agenda management.

As described above obligations stemming from contracts are put on the agenda. The Task Manager (in its role as Agenda Manager) forms plans of action to bring about what is specified by the obligation. In this conflicting obligations can arise (e.g., according to plan A action α should be executed, but an obligation states that action α may not be executed). It is the Task Manager's role to resolve such conflicts (e.g., based on the strength of the obligations, or on priorities given to the actions).

Before executing the action on top of the agenda the Task Manager has to check if all conditions (still) hold, including authorization checks and checks for changes in the environment. If the state of the world (the environment of the agent) has changed it might not be necessary anymore to perform that action (or other actions on the agenda). The Task Manager should then reorganize the agenda.

c) failure management.

The Task Manager handles failures due to subtasks that cannot be fulfilled, and due to invalidating actions like cancellations. In many cases it is notified by the Contract Manager (see above). In the first case alternatives are tried, if no alternatives are specified or if they fail, it means the subtask fails and backtracking is tried (seeking alternatives for the parent task). In the second case a contingency plan is triggered that specifies how to revalidate the result of the subtask again, or what should happen if it cannot, e.g. compensation of dependent results.

The working of the Task Manager is explained in more detail in section 5.3.2, and algorithms for these processes are given in appendix B.

4.4.3. SERVICE EXECUTION MANAGER

The Service Execution Manager is responsible for managing the services the agent provides throughout their execution. This includes both services rendered to other agents, as well as private actions. It involves three main roles:

a) service execution management.

The Service Execution Manager start executing services if notified by the Task Manager (that controls the agenda).

b) exception handling.

The Service Execution Manager monitors the execution of tasks and services for unexpected events and then react appropriately. In case the execution of the action depends on constraints being satisfied, the execution can be halted and the constraints can be checked at a later time (however, before any possibly specified deadline) and if satisfied the action can be retried. Alternatively, if the action fails the Task Manager should be notified, so an alternative can be tried. This will usually mean monitoring the database that stores information about run time application/service specific information, e.g. the services which are currently active and the current number of invocations of each active service, or the up-to-date information about the upper limit the agent will pay for a service that is negotiated.

c) information management.

If the agent is implemented using traditional database techniques the Service Execution Manager can also be seen as the database management system for the Services database, routing information between and assisting the Task and Contract Manager if they want to consult the Services database.

The Service Execution Manager is not specified further, since they are of less interest to the communication process. Appendix C describes a prototype in which the Service Execution Manager is implemented.

4.4.4. COMMUNICATION MANAGER

The main task of the Communication Manager is routing messages between:

(i) the agent and its peers, e.g. between the Contract Manager and other agents during negotiation. It also distributes incoming messages to the appropriate manager, e.g., a cancellation message is forwarded to the Contract Manager. In order to be able to communicate successfully with other agents (the messages must be understood by both sender and receiver) a communication protocol and a information sharing language is needed. Such a language must consists of a number of semantically grounded speech acts that specify the intention of the message (illocution). The meaning of the objects refered to comes from a domain specific reference ontology (e.g. the General Transport Conditions for passengers and luggage from an airline company) that is described in the Lexicon. This meaning can also be negotiated in the negotiation phase.

(ii) the agent and its responsible human counterpart. In this role the Communication Manager is the interaction manager of the human-computer interface. This is not worked out in this thesis any further.

Both types of communication draw from knowledge about communication (e.g., a request should always be answered, this can be seen as a politness rule), including knowledge about predefined protocols (e.g., the negotiation protocol mentioned earlier). E.g., if a request for a service comes in, the Contract and Service Managers should be consulted, and based on their answer a reply to the request is formulated. This can also be a counter request for more specific information about what is needed.

4.5. AGENT SPECIFICATION LANGUAGE COLA

For the specification of an agent program the language CoLa, short for Communication and cOoperation Language, is defined. Following the three-level communication model, the transactions the agent can take part in, the contracts governing the communication and also the agent's tasks can be specified. For easy reference these separate parts of the specification are called Trans, coLa and TaLa respectively and are described in section 5.1.4, section 5.2.4 and section 5.3.1. Parts of the grammar are given there, accompanied by examples from the business-trip situation. The full (EBNF) grammar of CoLa is given in Appendix A.

CHAPTER 5
COMMUNICATION FRAMEWORK

This chapter describes the three-level communication framework consisting of the transactions (section 1), contracts (section 2) and tasks (section 3) of the agents. In all sections the characteristic elements of the concepts are given, together with the specification language part of CoLa. In the contracts and tasks section also the Contract and Task Managers responsible for the management of respectively contracts and tasks are described. Finally, section 4 gives a comparison of the proposed framework with other transaction and workflow specification frameworks.

5.1. TRANSACTIONS

As described in chapter 1, more applications are being developed that access different independent resources (databases, knowledge bases), or are designed as being composed of different cooperative components. In many cases, integration of the various resources might not be possible, which motivates the support of interoperability, i.e. the technology necessary for combining multiple resources in a cooperative fashion using explicit communication facilities. Such a system that supports interoperability was called Cooperative Information System (CIS) before. Behaviour of a CIS is described by means of *interoperable transactions*. Below first a description of transaction is given. In subsection 2 the failure handling of failing transactions is described. Subsection 3 describes deadlines, an important point in specifying interoperable transactions. After this the transaction specification language part (Trans) of CoLa is given, using examples from the business trip case.

5.1.1. DESCRIPTION

An important aspect of an interoperable transaction is the specification of the coordination between operations in different autonomous agents. Coordination can be modelled as a communication process. Interoperable agents cooperate by means of *message* passing. Each agent participating in the process has certain obligations to fulfil depending on the role it is playing. The obligations of agents are made visible in the messages each agent

agrees to provide in the context of a particular interoperable transaction. The agents exchange messages in the order specified by communication constraints. These include sequences of actions that must occur together, conflicting actions, and triggers causing other actions to be taken. Through the messages, true cooperation within an interoperable transaction is achieved without violating the autonomy of the agents. It is for that reason that the specification of communication is emphasized, and focus on the synchronization of messages rather than operations in CISs.

It is a characteristic of messages that they seldom stand on their own. For example, a request is typically followed by an acknowledgement, commitment or refuse message (figure 5.1). Therefore messages can be organized in transactions. The request and confirm (or refusal) together form a transaction that results in an obligation of the service provider. E.g., the request to reserve a room can be confirmed, creating a commitment of the hotel to reserve a room, or can be refused.

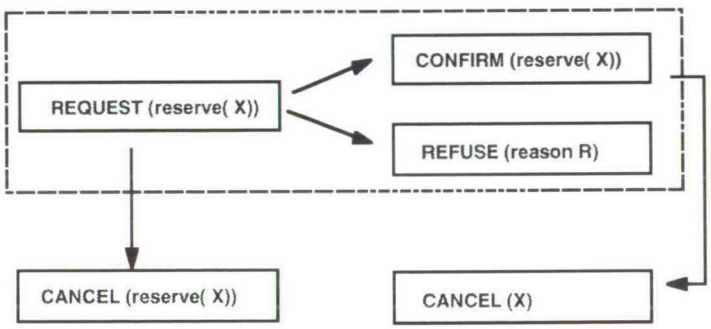


figure 5.11. Transaction

In contrast to other approaches (e.g., DEMO and BAT, section 2.1.3.2) that take material actions into account, this work focuses only on communicative actions. Material actions, like the delivery of goods, still exist but are outside of the view of the agents. Only the performance of a material action can be recorded. The agents do not have sensors that can observe changes in the world other than communicative ones. This means that real-world events, such as 'flight landed' do not play a role directly, but only via messages such as 'assert(flight landed)' by means of which a particular agent communicates the fact that the flight has landed. Moreover, since the message interface encapsulates the local database actions of the agent, the latter do not play a role either. Provided that there is agreement on the semantics of the messages exchanged, the specification of these messages is the only concern for a designer of an interoperable transaction. To put it sharply, for an agent sending a flight reservation request to an airline company, what counts is that the airline company replies by a positive confirmation, and not what it does in its database. This can and must be sufficient to let the transaction succeed. However, in order to get the actions done, the invocation of the actions is recorded as a message to the agent performing the action itself (this places the action on the agenda).

A specification of an interoperable transaction consists essentially of a set of communicating agents, elementary actions, for the communicative elementary actions (messages) the possible message types provided by each agent to support its role of either customer or supplier of services (or goods), constraints on the synchronisation of the actions (communicative or non-communicative), and the goal and exit states. According to speech act theory all messages are specified as an illocution function, such as request, and a propositional content (an action or proposition).

A transaction execution leading to a goal state means that the transaction succeeds, whereas an execution leading to an exit state means that the transaction fails. "States" are identified here by message occurrences.

Interoperable transactions can be nested. Higher-level transactions are aggregations of subtransactions, and can specify (temporal) constraints on the subtransactions. The constraints are propositional temporal logic formulae based on [Ngu et al., 1994]. It is also possible to allow more fine-grained constraints between subtransactions, e.g., that one subtransaction should occur after a certain message specified in another subtransaction. However these issues are not pursued here.

5.1.2. FAILURE HANDLING

The interoperable transactions used here differ from traditional database transactions in their longer time-span and larger structure. They introduce a unit of work and control that does not consist of individual database state transitions, but they define a control mechanism above ACID transactions.

Classical transactions are seen as a control mechanism that has the well known ACID properties ([Gray, 1981], [Haerder and Reuter, 1983], [Özsu and Valduriez, 1991]): Atomicity, Consistency, Isolation, and Durability. Despite their advantages for many straightforward database applications there are aspects of classical ACID transactions that limit their use in non-standard applications. As [Wächter and Reuter, 1992] describes: "they only perform well when the controlled units of work are small, access only a few data items, and therefore have a short system residence time".

For one thing it means that if there is a unit of work that has a structure which needs to be maintained by the system, it cannot be modelled as a transaction. However, the most fundamental drawback of traditional transaction systems in the context of long-lived applications is their notion of transactions being concurrent and completely unrelated units of work. As a consequence, any existing interrelations between individual transactions, like control flow dependencies and other semantic connections cannot be implemented by the system.

ACID transactions control concurrency by isolating atomic state transitions against each other in order to create a serializable schedule. To achieve this, nearly all concurrency control methods delay updates of the database until the commit of a transaction. However, this is not feasible for long-lived transactions because it leads to tremendous performance degradation because holding long locks block other activieties. It also leads to a high rate of transaction aborts due to conflict and deadlock resolution ([Wächter and Reuter, 1992]).

The elementary actions (communicative or non-communicative) that make up the interoperable transaction as defined here can be implemented by embedding them into a traditional ACID transaction, and their execution is serializable. The non-communicative elementary actions have all of the ACID properties but they preserve only local consistency for the manipulated objects. A whole interoperable transaction however is not an ACID unit of work. Although the interoperable transactions break isolation and atomicity because of there long-livedness, they do maintain consistency (and even on a larger scale) and durability (the global effects installed at the end of the transaction are durable and can be undone only by running another transaction). In this respect the transaction approach as described here is similar to the ConTracts approach ([Reuter, 1989], [Wächter and Reuter, 1992]).

Since transactions cannot be seen as ACID units and therefore the failure handling and recovery mechanisms based on this cannot be used, we must say something about the failure handling in our system.

Interoperable transactions are prone to more types of failures, caused by the longer time-span and the multiple parties being involved. Here we do not discuss (traditional) system failures such as crashes and failures of the network, but we focus on other semantical failure types (dealing with the meaning of the transactions) that cannot be ignored. These are:

- *cancellations* - one of the parties undoes an achieved effect, which necessitates either failure of the (sub)task or a rescheduling.

  A notorious problem with interoperable transactions, is that isolation cannot be maintained because of the long life-span. This means that partial results can become invalid later. E.g., where a flight reservation has been made, but the airline company cancels the flight. Alternatively also the original requesting agent can cancel, e.g. cancel the hotel reservation because no flight can be booked (and there are no other alternatives to make it to the hotel on time). See figure 5.1. The cancel messages are not part of the transaction, but undo the result obtained. Since they are part of the communicative behaviour between two parties the contract should specify remedial actions if such cancellations occur.

- *violations* - where the other agent does not comply to the agreements, which may cause "sanctional" actions on the one hand, and rescheduling of the subtask on the other.

  "Weak conflicts" (succeeded subtransactions that later fail, [Nodine et al., 1994]) usually originate from a violation of an obligation. For that reason, this approach insists that obligations and authorizations of service providers and receivers are modelled. However, here it is assumed that the cooperating agents will always try to honour their obligations, i.e. the communication is always reliable. In this way it is possible to reason over any non-cooperative action rather than classify them as a true failure (leading to unnecessary rollback or recovery actions of transactions).

Specifying the obligations and the way violations are handled in the interoperable transactions will not only complicate the specification, but also hinder the reusability of the transaction specification. Therefore the framework posits a separate layer on top of the transactions in which the deontic effects of transactions are described and these kind of failures are dealt with. Since these deontic effects are of immediate concern to both communicating parties, this specification is seen as a bilateral agreement. In order to ensure flexibility of interoperable transaction specifications, the use of a separate concept *contract* for modelling the obligation of an action in action-oriented communication processes is proposed. The contract is not the specification of a task of one of the agents. Rather it provides a basis for task specifications that involve cooperation with other agents. The user in most cases is only interested in the fact that the goal of the transaction is being achieved. Contracts and tasks are described in the next sections. The failure management itself is a task of the Task Manager of the agent. Besides a contingency plan that describes what steps to take in order to reach the goal of a subtask (again) also compensation tasks can be specified, that 'roll-back' results obtained so far.

5.1.3. DEADLINES

An aspect that plays an important role in flexible transactions is deadlines. In situations where several systems have to cooperate, deadlines are a means to specify expectations of the behaviour of the other parties. E.g., if a company delivers a product it expects a payment of the customer within a certain time. It is not difficult to develop a program that checks whether deadlines are met. The main idea is to wait until the deadline has passed, which usually can be checked easily, and then check whether a certain action has taken place. However, many difficulties arise when one tries to transform this procedural account of deadlines into a formal one.

In an intelligent agent system one would like to be able to reason about deadlines in order to check whether they can be fulfilled at all. This holds especially for combinations of different deadlines. Deadline constraints can also be used to influence the behaviour of the agent. The combination of deadlines can be used to plan the actions of an agent. Of course, this is only possible if the system has some formal description (besides a procedural one) of the deadlines.

A related issue is the occasion that a certain deadline is not met and what the consequences of such failure are. If one sees the failure to meet a deadline as a constraint violation, in some systems this would mean that the system reaches an inconsistent state. In these systems (most database systems currently in use) the constraints have to be fulfilled at any moment in time. Of course this can easily be enforced for static constraints. Any update of the system that violates a constraint is rejected. However, deadlines are constraints with a fundamentally different nature. Whether a deadline is met depends on an action that must have taken place. If the action is a local (database) action the planning system of the intelligent system can make sure the action is performed before the deadline. The performance of the actions is enforced and the deadlines would not have to be checked afterwards because they would be met by default (if possible of course).

Several problems can arise using this method. Take for instance the specification that an order should be placed before the stock falls below a certain level. It is not known at what point in time the stock falls below that level. It is therefore difficult to plan the action (ordering). We argue that the enforcement of the deadline and the planning problem are two separate issues and the enforcement of deadlines should not be implemented by a planning procedure. Of course, the deadlines do influence the planning of the actions of the intelligent agent, however they should not be enforced by it.

A second problem that arises with the enforcement of deadlines is that the system is not always capable of enforcing the performance of a certain action (if it is not a local action). E.g., upon delivery of the product the customer has to pay the bill within 30 days. The system can base its plans on the fact that the customer has paid within 30 days, but it has no way to enforce this payment (directly).

A last problem is the case where no specific deadline has been set. A certain action should take place "as soon as possible". E.g., after an accident has been reported, the ambulance has to go to the place of the accident as soon as possible. However, it might be that the ambulance first has to deliver another patient at the hospital or that the accident is not very serious and the ambulance does not switch on its siren. In these cases there is no definite point in time where one can check whether the action has been performed or not.

These examples show that deadlines cannot always be enforced. Not keeping a deadline should not result in the system being in an inconsistent state. Rather it should arrive at a state in which it is clear that a deadline has passed, but other (corrective) actions are still possible. (In case of the customer the system could send a reminder or a court order for payment). In the next section (5.2) is discussed how a contract can specify what should happen if a deadline is not kept.

5.1.4. TRANSACTION SPECIFICATION LANGUAGE (TRANS)

The communication between agents is described by messages, transactions and deontic states. Messages are defined using (speech act) primitives such as request, assert, and authorize. In this section the first part of the communication language CoLa, the transaction specification language Trans is defined. Section 5.2.4 describes coLa (contract specification language) for specification of the deontic states.

Messages that are aimed at establishing (or adapting) a certain deontic state (i.e., a conjunction of obligations and authorizations) are grouped together in transactions. The agents that are involved in the communication and the messages that each of them uses as explicit speech acts during the execution of the transaction are specified at the start of the transaction. For each transaction, there is a successful termination, indicated by the goal of the transaction, and non-successful terminations or exits. A transaction execution leading to a goal state means that the transaction succeeds, whereas an execution leading to an exit state means that the transaction fails. "States" are identified here by message occurrences. Furthermore temporal constraints on the synchronisation of messages can be specified. As we have seen above, deadlines can play an important role in the coordination between agents and optionally can be specified in the transaction.

The following gives an EBNF definition of a transaction declaration in Trans.

```
transaction_decl:   TRANSACTION trans_head trans_body
                    END-TRANSACTION ';'
trans_head :        trans_name ['(' arg ':' type [{',' arg ':' type}] ')']
trans_body :        AGENTS ':' {agent ':' type ';'}
                    {agent CAN SEND MESSAGES ':' {message_spec ';'} }
                    CONSTRAINTS ':' {trans_constr ';'}
                    [DEADLINE ':' {trans_deadline ';'}]
                    GOAL '=' message [{(AND|XOR) message}]
                    EXIT '=' message [{(AND|XOR) message}]
message_spec:       message TO (agent | self)
```

The messages correspond to the speech acts that are used in the communication. All messages are specified as an illocution function, such as request, and a propositional content (an action or proposition). This is in accordance with common practice in the Language-Action Perspective. The advantage of using these illocution functions is that otherwise the illocution is hidden in the message name (e.g. req_room), so that no generalization is possible over the semantics.

Each message type also includes the claim to authority, indicating the relation between the communicating agents. The three relations between agents that can be distinguished are: power, authority and charity. Power means that an agent A has some strong authority relation over agent B (e.g. boss and secretary). Authority means that an agent A can perform a speech act of which the intended effect is accepted by agent B based on the authority granted to agent A (either by agent B itself or by an agent that stands in a power relation with agent B). Charity is chosen as term for all 'free' peer-to-peer communication, where no relationship between agent A and B exist (or is not appropriate), e.g., a "request_quotation", where no relations have been established between agent A and B before, is defined as a request_C. Since we assume the agents to be benevolent, agent B will have the obligation to answer somehow, be it a quotation sent to agent A or a refusal, or a "don't know" which means agent B does not understand the request.

In section 6.3.2 these relations are defined formally, including the deontic effects the different speech acts in combination with the different claims have.

Besides the standard message types, like request and assert, some special message types are introduced, e.g., `AUTHORIZE` used by agent B to authorize agent A to perform some act. Also the reverse of authorize, `RETRACT`, the retraction of a granted authorization is included. Since the obligations of agents play such an important role in this approach some explicit message types indicating these deontic effects are included, such as `FORBID` and `PERMIT`. Furthermore abbreviations of some special message types can be given, e.g., the `CONFIRM` which is an abbreviation of the assertions of the special proposition 'received and understood'. Lastly, explicit messages to the agent self are included. Following [Shoham, 1993] these can be used for reasoning about its beliefs and actions. Finally, an `'action' (to SELF)` message can trigger the performance of some private action.

These message types all have pre-defined semantics in terms of obligations and authorizations, as is described in section 6.3.

```
message:    REQUEST '(' action ')'                    /* DIRc */
      |     COMMAND '(' action ')'                    /* DIRa */
      |     ORDER '(' action ')'                      /* DIRp */
      |     REQUEST '(' AUTHORIZE '(' action ')' ')'
      |     COMMIT '(' action ')'                     /* COM */
      |     SUGGEST '(' proposition ')'               /* ASSc */
      |     ASSERT '(' proposition ')'                /* ASSa */
      |     CLAIM '(' proposition ')'                 /* ASSp */
      |     ASSERT '('REFUSE-TO'('action [',' reason]')''')'
      |     ASSERT '(' ACCEPT '(' (action|proposition ) ')' ')'
      |     NOMINATE '(' proposition ')'              /* DECLc */
      |     DECLARE '(' proposition ')'               /* DECLa */
      |     ESTABLISH '(' proposition ')'             /* DECLp */
      |     AUTHORIZE '(' message ')
      |     RETRACT '(' action ')'
      |     FORBID '(' action ')'
      |     PERMIT '(' action ')'
      |     CONFIRM '(' action ')'
      |     action
```

For the specification of the temporal relationships in the constraints, Linear Time Logic can be used. Since we want to specify constraints on the possible execution paths of actions and constraints (things that have to be done) a small subset of the logic is used, specifically only those operators that deal with (possible) future actions, and leave out the operators that specify history. This means only the operators NEXT and UNTIL are used.

Definition 5.1. (Temporal precedence relations)

NEXT ϕ	ϕ should be initiated next
$\phi 1$ UNTIL $\phi 2$	execute $\phi 1$ UNTIL $\phi 2$ is initiated

And the derived operators:
SOMETIMES ϕ = true UNTIL ϕ
ALWAYS $\phi = \neg$SOMETIMES $\neg\phi$

The operator NEXT guarantees an event to occur after the current one. SOMETIMES conveys the message to honour the event at a later state. The unary operator ALWAYS is used to express cyclic sequences. E.g., ALWAYS T3 means that the formula T3 will always have the value true, this is because the condition T3 is always being regenerated. In [Ngu, 1990] it is shown that the composition of the various operators provides all the basic control constructs required in transaction modelling. The following example shows two mutually exclusive events A and B specified in the propositional temporal logic:

ALWAYS (A => NEXT ($\neg$(A OR B) UNTIL A_trigger))
ALWAYS (B => NEXT ($\neg$(A OR B) UNTIL B_trigger))

Instead of the frequently used: $\neg$T2 UNTIL T1, that expresses that event T2 cannot happen until event T1 has happened, the paraphrase: T1 BEFORE T2 is used.

In [Ngu et al., 1994] a temporal logic specification style for the precedence relations is used and it is shown how constraint specifications can be verified by constructing a dependency graph by applying tableau decomposition rules ([Wolper, 1981]). This is not

restricted to a directed graph structure, since the temporal operators provide the means to specify both partial order and cyclic dependencies. The paths in the graph correspond to possible execution sequences if the original specification is correct. Thus, the execution path provides the information as to the execution order of operations within an interoperable transaction. It provides a modelling mechanism which has a polynomial algorithm to prove the correctness of the specification.

```
trans_constr :          message_list BEFORE message_list
                |       ALWAYS '(' message '=>' temp_tr_constr ')'
message_list :          message
                |       '(' message {(AND|XOR) message} ')'
temp_tr_constr :        (NEXT|SOMETIMES) '(' trans_constr_el ')'
trans_constr_el :       message [{(AND|XOR) message}]
                |       message_list BEFORE message_list
                |       temp_tr_constr
```

As described in the previous section, deadlines are an important part of transaction and task specification. The deadlines that can be specified are the general deadline (something must be done after something else has been done or holds, including a specific time, and before something else is done or holds, including a specific time), the immediate deadline (something should be done now, as the next action), the deadline that specifies something should be done as soon as possible, and finally the periodic deadline (a reoccurring deadline, specifying something should be done periodically if some condition holds).

```
trans_deadline:     ['P'] [deadl_cond '<<'] message '<<' deadl_cond
                |   message '!'          /* immediate */
                |   message ASAP         /* As soon as possible */
deadl_cond:         proposition|message|time|message '+' time|now '+' time
```

Below we specify some transactions that can be found in the business-trip situation.

5.1.4.1. BUSINESS TRIP TRANSACTION EXAMPLES

We consider again the communication between agents involved in planning a business trip; this consists of transactions involving the travel agent and agents from an airline company, a credit card company and a hotel. The agents can come from different databases/systems such as an airline database or a hotel chain's database. The example below is from the specification of the hotel-agent and describes the hotel-reservation transaction with the messages that the other agent can send and the possible replies from the hotel-agent. Furthermore the constraints on these message exchange are given.

```
transaction hotel_reservation
agents:
      u: user; /* all the potential customers in the network */
      h: hotel;
u can send
      messages:
            request(reserve(room)) to h;
h can send
      messages:
```

```
            reserve(room) to self;
            confirm(reserve(room)) to u;
            assert(no_available(room)) to u;
            assert(refuse-to(reserve(room))) to u;
constraints:
      request(reserve(room)) BEFORE reserve(room);
      request(reserve(room)) BEFORE assert(refuse-to(reserve(room)));
      reserve(room) BEFORE confirm(reserve(room));
      reserve(room) BEFORE assert(no_available(room));
      ALWAYS(request(reserve(room)) => NEXT(SOMETIMES
            (confirm(reserve(room)) XOR assert(no_available(room))
            XOR assert(refuse-to(reserve(room))))));
Goal = confirm(reserve(room)) XOR assert(no_available(room))
Exit = assert(refuse-to(reserve(room)))
end-transaction;
```

The ALWAYS constraint states that whenever the hotel agent receives a request for a reservation, sometimes later it will send a message back to the customer which is either to confirm the reservation, or that there are no available rooms, or that he refuses to take the request into consideration.

As shown by figure 5.1 both the hotel and the customer can cancel the reservation (at a later stage). The cancel message is not part of the transaction, but instead is specified in the contract. In the contract is described what should happen if one of the agents violates an obligation resulting from a commitment made (the confirm of the reservation and acceptance by the customer, the last one being implicit here).

The examples specify respectively: a flight-reservation transaction for a travel agent, including the messages than can be exchanged, the triggering of a private action (check(flight-schedule)) and the specification of an 'as soon as possible' deadline; a hotel-reservation transaction from the travel-agent points of view; and a creditcard payment transaction, including the specification of an immediate deadline.

```
transaction flight-reservation
agents:
      a: airline;
      t: travel-agent;
a can send
      messages:
            confirm(reserve(ticket)) to t;
            assert(no_available(ticket)) to t;
            assert(flight-schedule) to t;
            assert(refuse-to(reserve(ticket)),reason) to t;
t can send
      messages:
            request(reserve(ticket)) to a;
            request(get(flight-schedule)) to a;
            check(flight-schedule) to self;
constraints:
      request(get(flight-schedule)) BEFORE check(flight-schedule);
      check(flight-schedule) BEFORE request(reserve(ticket));
deadline:
      request(reserve(ticket)) ASAP;
Goal = confirm(reserve(ticket))
Exit = assert(no_available(ticket)) XOR
            assert(refuse-to(reserve(ticket)),reason)
end-transaction;

transaction hotel_reservation            /* for the travel-agent */
agents:
      h: hotel;
```

```
        t: travel-agent
h can send
        messages:
                confirm(reserve(room)) to u;
                assert(no_available(room)) to u;
                assert(refuse-to(reserve(room))) to u;
u can send
        messages:
                request(reserve(room)) to h;
constraints: [OMITTED]
Goal = confirm(reserve(room))
Exit = assert(no_available(room)) XOR
                assert(refuse-to(reserve(room)))
end-transaction;

transaction creditc_payment
agents:
        u: user;
        c: card company;
u can send
        messages:
                request(pay_with(creditcard)) to c;
                request(increase(credit)) to c;
c can send
        messages:
                validate(creditcard) to self;
                pay_with(creditcard) to self;
                assert(accept(pay_with(creditcard))) to u;
                assert(not_enough(credit)) to u;
                assert(invalid(creditcard)) to u;
                assert(refuse-to(increase(credit))) to u;
constraints:
        validate(creditcard) BEFORE
                assert(accept(pay_with(creditcard)));
        [OMITTED]
deadline:
        validate(creditcard) !;
        assert(accept(pay_with(creditcard))) !
Goal = assert(accept(pay_with(creditcard)))
Exit = assert(not_enough(credit)) XOR assert(invalid(creditcard))
                XOR assert(refuse-to(increase(credit)))
end-transaction;
```

Note that we take a functional approach to the specification of messages. For example, `request(reserve(room))` and `request(get(flight-schedule))` use the same generic action `request()`. This gives the provision to state general constraints, such as that commit(α) cannot precede request(α), for any action α.

5.1.4.1.1. Special cases

Another example is a more general one and describes the transactions in the case of an ordering procedure. It states that the transaction `quotation` is a specialization of the transaction `get_authorization`. This transaction is defined as a request to get authorization, followed by an authorization or a refusal. Specialization means inheritance of the message set, the constraints, the Goal and the Exit. All these parts can be extended (not overruled) in the specialization. The message `authorize` is predefined and creates an authorization for a certain action. Note that the constraint here is so self-evident that it is better built-in in the language as an axiom. See also the next subsection about instant and delayed services.

```
transaction quotation
isa get_authorization(order(Partno,Quantity,Price),self,provider)
end-transaction;

transaction get_authorization(a:action,c,s)
agents:
      c: customer;
      s: supplier;
c can send
      messages:
            request(authorize(a)) to s;
s can send
      messages:
            authorize(a) to c;
            assert(refuse-to(authorize(a))) to c;
constraints:
      request(authorize(a)) BEFORE authorize(a);
Goal = authorize(a)
Exit = assert(refuse-to(authorize(a)))
end-transaction;
```

The `order` transaction below is defined for the cases where no quotation is given before. The request to deliver a product is therefore based on charity. If answered the company commits itself to deliver the product (if available) against a certain price after which the price has to be accepted by the customer.

```
transaction order-product
agents:
      c: customer;
      s: supplier;
c can send
      messages:
            request(deliver(Partno,Quantity,Price)) to s;
s can send
      messages:
            commit(deliver(Partno,Quantity,Price)) to c;
            assert(not_available(Partno,Quantity)) to c;
constraints:
      request(deliver(...) BEFORE commit(deliver(...));
      request(deliver(...) BEFORE refuse(not_available(...));
Goal = commit(deliver(...))
Exit = assert(not_available(...))
end-transaction;
```

The following example shows a typical transaction, in which the two partners express agreement about a situation by means of two messages. In most cases, the accept is essential, since delivery is more than saying the order is delivered, it is also more than just putting the containers in front of the entrance. Only when the customer accepts the goods as such (a speech act), the delivery succeeds.

```
transaction order-delivery(O: order)
agents:
      s: supplier;
      c: customer;
s can send
      messages:
            assert(delivered(O)) to c;
c can send
      messages:
            assert(accept(delivered(O))) to s;
Goal = assert(accept(delivered(O)))
end-transaction;
```

In the following example, the request to transfer money is a request based on authority following from the relation between bank and customer. The result is that the bank must commit to the money-transfer unless there is some other condition that violates the agreement (such as not enough credit). The refuse message is extended to include the reason for the refusal.

```
transaction payment(s:supplier,$:amount)
agents:
      c: customer;
      b: bank;

c can send
      messages:
            command(transfer_money(s,$)) to b; /* =DIR_a */
            request(increase_credit) to b;
b can send
      messages:
            commit(transfer_money(s,$)) to c;
            assert(credit(c,$)) to c;
            assert(refuse-to(transfer_money(s,$)), "not enough
                  credit") to c;
constraints:
      assert(refuse-to(transfer_money(s,$), "not enough
            credit")) BEFORE request(increase_credit);
      request(transfer_money(..) BEFORE commit(transfer_money);
Goal = commit(transfer_money(s,$))
end-transaction;
```

5.1.4.2. TRANSACTION TYPES

As described above a transaction is a set of messages and constraints between them. This section starts with an analysis of the example business trip transactions of the travel-agent. The goal of this analysis is not to improve on the syntax, or extend the expressivity, but to obtain a better understanding of what is going on, and how flexibility and modularity can be improved. Three observations can be made, the first regarding the transaction process, the second regarding prototypical transactions, and the third regarding modularity.

1. transaction process

Although transactions often take the form of a request followed by a commit, or an assert followed by an accept, sometimes intermediary messages are allowed, as with `increase_credit` in the credit-card payment example, and sometimes the response is superfluous, because the request/assert was already authorized. In DEMO ([Dietz, 1994a,b]), communication is modelled exclusively by means of so-called actageneous and factageneous conversations, creating an agendum and a fact respectively (see section 2.1.3.2.1). Between those conversations the essential action, the act that is requested in the actageneous conversation and that the addressee committed itself to, is performed. Each of these conversations is required to have at least two messages (request/commit, assert/accept). A limited set of additional messages, to be used in the negotiation, such as refuse, and counter offer, and discussion about validity claims (based on [Habermas, 1981]) are allowed as well. The structure of this transaction process can be represented by the Transaction Process Model (TPM) of van Reijswoud ([van Reijswoud, 1996]).

Actageneous and factageneous conversations are not specified explicitly in the transaction specification given above, but the basic idea is incorporated and worked out in the formal framework where the intended effects of certain speech acts are defined, e.g. the obligations that arise for the addressee after an authorized directive of the speaker, or the belief about a fact that is created in the addressee after the assertion of the fact by the speaker. In an agent-oriented setting the obligations of an agent are put on his agenda as things to do or bring about.

We do not force the specification of all reply messages, unless the acceptance of some specific service is described (see the order delivery example above). The reason for this is the focus on the contract between two parties. A contract gives rise to a certain communication process or protocol and these protocols are based on the essential speech acts given in the contract. So in the transaction specification we only focus on those transactions and communicative actions that have some deontic effect.

The transaction specification given here mainly deals with the success layer of the TPM. Failures due to agents not obeying their obligations are being dealt with in the contract specification (section 2). Failures due to cancellation are dealt with in task specification (section 3) and are not described on the transaction level.

Discussion about validity claims often boils down to discussion about the conditions under which certain acts can be performed. This is therefore part of the negotiation process of setting up (or modifying) a contract describing the mutually agreed actions of the partners in the communication. However, in analyzing a situation the TPM can be used to model the different transactions that are going on and which ones should be accounted for in the contracts and which one in the task's manager failure handling.

2. prototypical transactions

A second observation that can be made is the close similarity between the `flight-reservation` and `hotel-reservation` transaction. Both contain `request(reserve())`, `confirm(reserve())`, `assert(not_available())`, where the only difference is in the object, be it a ticket or a room. There are some non-essential syntactic differences in expression, but the basic structure is the same. The Goal and Exit conditions of both subtransactions are similar as well. What this suggests is that we can have a generic case, a "parameterized" transaction that is characterized as "reservation". This generic transaction is customized according to the object of reservation. From a software engineering point of view, the parameterized transaction would be a good candidate for reuse, to be handled as a kind of generic superclass in an object-oriented framework. Derived subclasses are then allowed to specialize by adding constraints or other messages, such as, those about the flight-schedule in the case of flight-reservation.

Although this is only one example, the number of different parameterized transactions is rather limited, making it worthwhile to look for a basic set that could cover 90% of the cases. Below two prototypical transactions are modelled, called DELAYED_SERVICE and INSTANT_SERVICE. The INSTANT_SERVICE transaction type is applicable when the service is offered by the other party instantly. In that case, reply coincides with the offering of the service, as for example the providing of a flight-schedule.

```
transaction i_sv(service(object), agentA, agentB)
agents:
      agentA: customer;
      agentB: supplier;
agentA can send
      messages: request(service(object)) to agentB;
agentB can send
      messages:
            service(object) to self;
            assert(object) to agentA;
            assert(no_available(object)) to agentA;
constraints:
            service(object) BEFORE assert(object);
            service(object) BEFORE assert(no_available(object));
Goal = assert(object)
Exit = assert(no_available(object))
end-transaction;
```

The DELAYED_SERVICE transaction is applicable if the service offered by the other party is an external event, and the essence of the transaction is the commitment of the other party to deliver that service. An 'order' of goods is a typical example. An interesting subtype is when the service is a reservation. In that case, the action contians granting of an authorization with respect to another service. The object of the reservation might be a hotel room, a flight seat, a book rental, a transport. An example is the `get_authorization` transaction given above. The generic DELAYED_SERVICE describes the possible messages and their synchronization. In real-life examples, the specialization of such transactions will typically include extra conditions on these actions.

```
transaction d_sv(action, agentA, agentB)
agents:
      agentA: customer;
      agentB: supplier;
agentA can send
      messages: request(action) to agentB;
agentB can send
      messages:
            action to self;
            confirm(action) to agentA;
            authorize(service) to agentA;
            assert(refuse-to(action),reason) to agentA;
constraints:
      request(action) BEFORE confirm(action);
      request(action) BEFORE assert(refuse-to(action),reason);
      request(action) BEFORE authorize(service);
Goal = confirm(action) XOR authorize(service)
Exit = assert(refuse-to(action), reason)
end-transaction;
```

Applying the parameterized transaction framework to the flight-reservation transaction, we can simplify it to be an aggregation of an INSTANT_SERVICE with object `flight-schedule`, and a DELAYED_SERVICE with object `reserve(ticket)`. The resulting representation below models the transaction as a subclass of the delayed service (inheriting its constraints and goal), and including the instant service (inheriting its constraints, but not its goal). The notation `d_sv` is used for DELAYED_SERVICE and `i_sv` for INSTANT_SERVICE. Both `d_sv` and `i_sv` are interoperable transactions.

```
transaction flight_reservation
      isa d_sv(reserve(ticket),t,a)
agents:
      t:travel-agent;
      a:airline;
include:
      i_sv(get(flight-schedule),t,a);
constraints:
      i_sv(get(flight-schedule),t,a) BEFORE
            d_sv(reserve(ticket),t,a);
      check(flight-schedule) BEFORE d_sv(reserve(ticket),t,a);
deadline:
      d_sv(reserve(ticket),t,a) ASAP;
end-transaction;
```

Notice the simplified representation of this transaction as compared to the original `flight-reservation` transaction. In the flight-reservation, it seems likely that the temporal constraint between `check(flight-schedule)` and `request(reserve (ticket))` and also the deadline is specific to the transaction, and hence should not be included in the generic `d_sv` and `i_sv` transactions.

Although important from a software engineering point of view, parameterized transactions are not yet implemented in the specification language.

3. modularity

A third observation is the modularity in the specification. Instead of one big business trip transaction, different transactions that each deal with a bilateral relation ((travel agency - airline company), (user - credit card company)) are specified. This modular approach is useful because the agreements/obligations between travel agent and airline company can change now without affecting the other agents' parts (such as tasks). Another potential advantage is that the same subtransaction, e.g. `creditc_payment`, can be used in quite different applications.

In fact, the transaction `creditc_payment` can be remodelled as a contract between user agent and credit card company. It describes a number of services offered by the credit card company, which are made use of in the task of planning a business trip. We can think of a cooperative environment as having an indefinite number of applications, identified by interoperable transactions. Each of these transactions depends on the action-oriented communication by drawing on several contracts to achieve its goal.

5.2. CONTRACTS

This section describes contracts, and its basic concepts: authorizations and obligations. We start of with a desciption of contract and contract creating, followed by a discussion about relationships between contracts. After this the contract specification language part coLa of the communication specification language is described.

5.2.1. PROPERTIES OF CONTRACTS

In this thesis a contract is defined as a set of related authorizations (permissions and prohibitions), and mutual obligations, together with conditions on the relationship between them, and rules governing violation of authorizations and obligations.

A contract is set up by two or more agents when communicating, usually through a process of negotiation (see below). It describes the authorized communication behaviour among the receivers and providers of services in terms of interoperable transactions. The purpose of the contract is to clarify the position of the communicating (business) partners and should be consulted whenever there are problems with or questions about the authorizations and obligations of one of them. The contract does not address problems about the meaning of the objects the communication is about (semantic problems), for this purpose the lexicon should be consulted. The actual obligations are created by individual messages, which are usually justified by authorizations laid down in the contract. But a contract can also cause the creation of an obligation in the absence of a directive message, e.g. because a certain state has arrived. The specifications of the authorizations and obligations are modelled using dynamic deontic logic. E.g., if a message is directive, the source of the directive is supposed to be authorized to command these actions, and the receiver is committed to follow the resulting obligation. If the receiver does not adhere to the obligation, it is the job of the Contract Manager to impose violation policies.

5.2.2. CREATING CONTRACTS

Whenever an agent requires a service provided by another agent it cannot simply instruct it to execute that service, since agents are autonomous entitites and there are no control dependencies between them ([Jennings et al., 1996a]). This is exactly one of the main differences between agent systems and more traditional forms of distributed processing ([Smith and Davis, 1980]). Instead, the service requesting and service providing agents must come to a mutual agreement about the terms and conditions under which the desired service will be performed. Such an agreement is defined here as a contract. As we have seen in the business logic framework (section 3.1) a contract is the result of a *negotiation* process. [Müller, 1996] defines negotiation as: "a joint decision making process in which the parties verbalise their (possibly contradictory) demands and then move towards agreement by a process of concession or search for new alternatives".

In order to be able to negotiate a protocol is needed describing the role of the current message exchange, i.e. whether the agent is making a proposal, or if it is accepting or rejecting a proposal ([Jennings et al., 1996a]). Based on the business logic a model of the negotiation process can be developed. This is similar to the approach taken in the ADEPT framework ([Jennings et al., 1996a], and [Alty et al., 1994] for more detail). The protocol is based on speech-act performatives (section 2.1).

The protocol consists of a initial phase of finding out what services an agent can provide, by simple requesting and answering (asserting) speech acts. This information can be stored in the Services-base for reuse, in which case this phase can be skipped.

The following phase is the phase of coming to an agreement. In the ADEPT framework four illocutions are used: `propose`, `counterpropose`, `accept` and `reject`. [Dignum, 1997] shows how these basic illocutions can be modelled within our logical framework (described in the next chapter). It describes how each of the speech acts mentioned above can be expressed in the basic speech act types adopted here: commissives, declaratives and directives. Furthermore it specifies precisely what permissions and especially what obligations are raised for the negotiating agents.

Each type of speech act (and its effects) should be interpreted within the background of the relationship between the speaker and the hearer of the speech act. We distinguish three types of relations between agents that have an effect on the negotiation: a charity relation (or a peer relation), a power relation, and authorization relation. The first two are similar to the ones used in the ADEPT framework.

The *power* relation is used to model hierarchical relations between agents. We assume that these relations are fixed during the lifecycle of agents. Within such a relation less negotiation is possible about requests and demands. E.g., if there is a power relation between one agent and another the last one cannot reject any proposals from the first one outright. This reduces the amount of communication and therefore increases the efficiency of agents.

The *peer* relation exists between agents that have no prior contract or obligations towards each other (with respect to the present communication). This relation permits extensive negotiations to allow a maximum of autonomy for the agents. The agents must negotiate in a cooperative (rather than in a competitive) manner.

The *authorization* relation is a type of temporary power relation that can be build up by the agents themselves, e.g. one agent explicitly permits another agent to direct it from now on to accept any offer that agent makes. The relations are described in more detail in section 6.3.2.

The protocol furthermore contains a management phase of actually invoking the agreement and making sure the service is provided, which is guided by both the Contract Manager and the Service Execution Manager, as described in section 4.4.

[Müller, 1996] describes that, besides the communication protocol, a negotiation model consists of a decision component describing the reasoning of the negotiators. This component usually contains utility functions, preferences, and negotiation strategies. Existing work on this aspect of negotiation can be divided into two camps.

On the one hand, much theoretical work (e.g. [Nash, 1950], [Raiffa, 1982], [Rosenschein and Zlotkin, 1994]) is based on game-theoretic and economic notions, providing insight into how agents should negotiate to produce 'optimal' solutions. However, as [Jennings et al., 1996a] state, a number of unrealistic assumptions are common in these models, such as: the availability of complete action descriptions, a utility function that can order all alternatives in all contexts, and perfect rationality of the agents when selecting actions.

On the other hand, more practical work typically adopts a superficial approach to negotiation. For instance in the Contract Net protocol ([Smith, 1980], [Smith and Davis, 1980]) the Manager sends out a request to a number of potential contractors to provide a given service with a given degree of quality. The potential contractors return a bid if they are capable of fulfilling all the requirements. The Manager then selects the best bid. However, this model fails to capture many intuitive and important aspects of the negotiation process, e.g. bidders cannot counter-propose better options, and cannot modify any of the contract parameters, and it is one-sided in that it only emphasizes the responsibility and satisfaction with the requesting agent.

Therefore Jennings ([Jennings et al., 1996a]) propose a deep and explicit model of the negotiation process. The model contains declarative and procedural knowledge.

The declarative knowledge models what is being negotiated for and why negotiation is taking place (it sets the negotiation context), e.g., negotiation over the start-time of a service if a proposal conflicts with an agent's existing commitments This knowledge draws on previous contract knowledge and task knowledge.

The procedural knowledge is represented as a set of strategies, and mechanisms for selecting between them. It specifies which actions should be taken given the declarative knowledge, e.g., if the agent has a long time to reach an agreement or if there are many potential suppliers of a service the Contract Manager may indicate that Boulware (a strategy in which the negotiator makes a reasonable initial offer and then sticks firm with it throughout the negotiation [Raiffa, 1982]) can be adopted.

5.2.3. RELATIONSHIPS BETWEEN CONTRACTS

In this subsection we will take a look at different relationships that can exist between contracts. We examine contracts not only from a technical viewpoint but also from an organisational one. From a technical point of view, a contract is nothing but a protocol binding different parties to their commitments by explicitly specifying the type of services agreed upon, the obligations resulting from it and the violation recovery methods. From an organizational point of view, however, a contract between interoperable systems involving different organizations also has the purpose of laying down a formal (and legal) agreement. This means for instance contracts should be "grounded" in other 'higher-level' contracts or (international) business laws. Below these ideas are described.

It should be noted that more research should be done on this aspect and in the next subsection where the specification of contracts is described we are only dealing with the technical view of a contract, i.e. the specification of the lowest level of contract, the agreements between two communicating parties.

Contracts, in the general sense of agreements between commercial partners, can be more or less elaborate and defined on two levels.

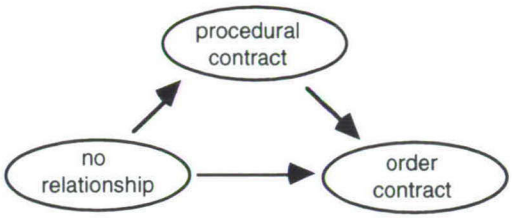


figure 5.2. Contracting on two levels

Starting from a situation where no (business-)relation between the agents exists (figure 5.2), agents have the possibility to offer services. A service can be seen as a publicly accessible interface of the agent, comparable to a method interface in OO programming.

In the first subsection, a contract was defined as a set of services that specify authorized messages (in transactions) and their deontic effects (authorizations and obligations of the communicating parties). Nowadays it is considered an economic advantage when business relations can be tied closer with some preferred supplier or customer. In that case the two parties can make agreements beforehand, e.g. about a guaranteed delivery time. We assume that such a contract is set up by the two partners only once and then frequently used. We call this a *procedural* contract.

In the business logic model a contract is the result of negotiation, and in a business setting implies a commitment to deliver a product and a commitment to pay. A supplier agent can provide the service of giving an offer. When a customer agent requests for this service, the supplier can refuse or accept. If he accepts, he sends an offer, that is, an authorization to order the product with the obligation to pay for it. If the customer agent uses this authorization a contract has been accomplished. We call this an *order* contract.

If the agents have done business before then in setting up the order contract the agents can take into account the agreements made before and laid down in a procedural contract.

Composition of contracts corresponds to traditional composition of modules, and contract specifications should have the possibility to "include" other contracts. For instance, in a contract between a travel agency and airline company about the ordering of tickets a generic contract about the order procedure, delivery and payment could be included so that only the agreements have to be specified on getting flight-schedules, changing requests (e.g., non-smoking seat instead of smoking-seat) and what should happen in case of cancellation by one of the partners.

Another relationship between contracts, mentioned above, is that between the business contract and a *grounding* contract, i.e., a contract that states the value of the contract actions in a higher-level context. By grounding the contract in other contracts or business law, it can obtain legal status. For organizations it is important that the highest level of contracts are ultimately grounded in (inter)national (business) law. E.g., laws of

commerce usually contain an article that if a commercial contract imposes a penalty on a party for non-compliance with the contracted obligations, the affected party may demand fulfilment of the contract.

There are basically two ways of defining a "grounding" relation: bottom-up, where a higher order contract or institution stipulates the legality or validity of lower order contracts; or top-down, where lower order contracts or transactions state explicitly that the obligations and authorizations from this level are validated or have a legal status by some higher order law or contract. Which one should be preferred is not clear at the moment, but a software engineering point of view is preferred where lower order contracts explicitly point to higher order contracts for the validation of the obligations and authorizations. These higher contracts can be used in case of conflicts when a party that violates an obligation questions the sanctions that are laid upon him.

Contracts describe the obligations and authorizations of the communicating agents. As mentioned in chapter 1 (and worked out in chapter 6) deontic logic is used for the specification of these concepts. Also the relations between contracts can be described using deontic logic. Some work on this aspect has been done in [Royakkers and Dignum, 1994]. For describing relations between contracts two logics can be distinguished:

(i) the intra-deontic logic, common to all contracts. This logic is described in section 6.1.1;

(ii) the inter-deontic logic that is about relations among deontic units. [Castaneda, 1982] describes four different inter-deontic structures:

1. deontic adjunction

This is the simple union of two contracts that are not in conflict with each other, e.g., the procedural and order contract relation.

2. sanctional dovetailing

A dovetailing contract contains actions that can or must be undertaken when the other party violates an obligation (the punitive type) or what must be done when an obligation of the other contract is fulfilled (the rewarding type).

It seems useful to have a 'sanction' link between contracts. If, as the result of a violation, a new obligation is created, e.g., give a refund if the original flight is cancelled, there is no way to enforce the fulfilment of the new obligation. However, we do not want to clutter the contract specification with several levels stating what should happen if the other party does not adhere to the new obligation. Instead it should be possible to point to a higher level contract (law) that specifies what sanctions can be taken (e.g. go to court if the fine is not paid, the laws of commerce under which both parties work specifies what should happen).

Sanctional dovetailing has some analogy in the use of exception handlers. It has the advantage that the process remains more transparent, although problems may arise how the process should continue after the exception handler has returned. In the contracts described in this thesis we try to specify what actions are necessary to return to the success-line leading to a preferred goal state.

3. delegational expansion

A delegational expansion contract specifies that certain other agents can perform actions that have normative power for the contracting agents. E.g., by the contract between an airline company and its sales-department employee, the employee is delegated the power to set-up contracts between the airline company and a travel-agent. All obligations arising from the contract are not obligations for the employee (although he/she might be the one performing actions to adhere to the obligation), but formally they are the obligations for the airline company. See also section 6.3.2.3 for matters concerning the delegation of authorizations. More about this type of contracts can be found in [Dewitz, 1991].

4. conflict-resolving expansion

Such a contract contains rules that say what must be done in the situation where another contract contains conflicting duties for the contracting agents.

Conflict-solving expansion occurs in an IS environment when conflicts cannot be solved by adapting the systems (e.g. because the systems are legacy systems). E.g., if there exists a contract between an airline company and a travel agency about the ordering of tickets that specifies the policy of the airline to offer alternative flights (and not pay fines) in case of cancellation, and a new contract is set up for certain flights that contains the obligation of the airline company to pay a fine if the flight gets cancelled. In case a flight gets cancelled the question is which contract prevails. A third contract might specify rules for deciding which obligation holds for the airline company.

In the present framework, the "dovetailing" contract is included in the contract itself. Other inter-contract references are not yet taken into account.

Overall there are three ways agents can interact in business communication: directly (e.g., the request for a service); following a standard or permanent contract (e.g., for preferred business relations); and, by setting up a new contract (e.g., for a one-time service under some conditions). Figure 5.3 gives a graphical illustration.

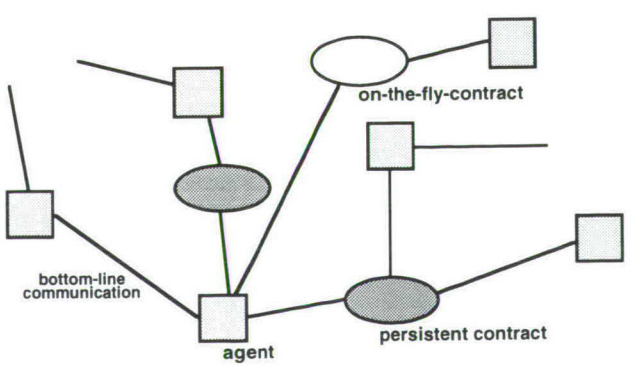


figure 5.3. Three types of communication lines

5.2.4. CONTRACT SPECIFICATION LANGUAGE (COLA)

The reader may have noted that the term 'contract' is sometimes used for the specification of the transactions plus the way failures are dealt with, and sometimes for the latter part only, separated from the transactions. This is to avoid inventing yet another term. In the following, the restricted meaning is intended.

Contracts are defined as units of cooperation between two or more agents. coLa (the contract specification language part of CoLa) provides a means to describe the communications between agents and the resulting obligations and authorizations in a structured way. A contract is therefore specified as a set of deontic clauses. A deontic clause describes the status of an interaction of the two partners in terms of obligations, authorizations and accomplishments. Logically speaking, this is a complex deontic logic formula. This formula can be put into a conjunctive normal form, where ach clause is a disjunction of literals. The contract describes the deontic clauses and their dynamics, that is, a clause is created by one or more transactions, and is removed by other transactions. The EBNF grammar of coLa is as follows:

```
contract :          CONTRACT contract_name contract_body END-CONTRACT ';'
contract_name :     ident
contract_body :     AGENTS ':' {agent ':' type ';'}
                    CLAUSES ':' {clause ';'}
clause :            clause_name ':'(OBL|AUT'('agent','action_spec')')|
                                ACC '(' agent ')'
                    IN ':' transaction [{',' transaction}]
                    [DEADLINE ':' obl_deadline]
                    [GOAL':'{action_spec'=>'clause_name [{'&'clause_name}]}]
                    [EXIT ':' {(cancel '(' action ')' | transaction)
                            '=>' clause_name [{'&' clause_name}]}]
                    [MODIFIED-BY {(action_spec|message)}]
                    END clause_name
action_spec :       transaction | action
```

The status of the interaction can be represented by a *set* of clauses, as in a Petri Net, where a state is identified by a set of token placements. A Petri Net consists of a set of places, a set of transitions, an input function and an output function from the set of transactions to the set of places. Petri Nets have been widely used in process modelling ([Murata, 1989]), but the interpretation of the states can vary. In this case, states are identified with a deontic clause and the transitions are interoperable transactions that create or remove the deontic clause.

The clause is identified by a clause name (unique within the contract). Its content is specified as a deontic formula. The IN part refers to the transactions that lead to this state, provided they have been closed successfully. The GOAL and EXIT transactions have the effect of moving to another deontic clause. The current state is then no longer valid. The difference between GOAL and EXIT is that one involves the fulfilment of the obligation whereas the other involves a violation or cancellation.

Optionally, a deadline can be specified on the obligation, using the specification style of the transaction deadline (section 5.1.4). Violation means the deadline has passed without fulfilment of the obligation.

```
obl_deadline :      [(action_spec|time|action_spec'+'time|now'+'time) '<<']
                    action_spec '<<'
                    (action_spec|time|action_spec'+'time|now'+'time)
                |   action_spec '!'
                |   action_spec ASAP
```

The contract also allows the specification of update transactions (indicated by MODIFIED-BY) that do not invalidate the deontic content but only modify certain parameters (e.g. a smoking seat instead of a non-smoking seat).

The following example illustrates deontic states for a flight_reservation. It is a contract between the airline company and the travel agency.

By a flight-reservation states S1 and S2 are reached. S1 denotes that the airline company has the obligation to fly (alternatively it can be specified as the authorization of a person to board the plane), and S2 that the travel agency has the obligation to pay. If the ticket is cancelled by the travel agency, state S8 is reached where the agent should pay a fine to the airline company. Alternatively the airline might cancel the flight, leading to state S6 specifying that the airline has to pay a fine, or state S5 if the ticket was already paid for. Finally, the travel agency's goal state S3 can be reached if the airline flies to its destination and/or the airline paid the fine. The airline's goal state S4 is reached if the travel agency has fulfilled all its obligations (to pay the ticket or the fine if the ticket is cancelled). If both parties are satisfied (reached their goal states) the contract ends.

```
contract airline
agents:
      a: airline;
      t: travel agency;
clauses:
/* obligation of airline after transaction flight reservation */
S1: obl(a,A.fly)
in    T.flight_reservation(flight-schedule)
goal  A.fly => S3
exit  cancel(reserve(ticket)) => S7 & S8
      cancel(flight) => S5 & S6
end S1;

/* obligation of travel agency after flight reservation */
S2: obl(t,T.pay(ticket))
in    T.flight_reservation(flight-schedule)
goal  T.pay(ticket) => S4
exit  cancel(reserve(ticket)) => S8
      cancel(flight) => S6
end S2;

/* accomplishment of travel agency */
S3: acc(t)
in    A.fly,T.pay(fine),T.payback(ticket),cancel(reserve(ticket))
goal  end-contract
end S3;

/* accomplishment of airline */
S4: acc(a)
in    T.pay(ticket), cancel(flight), T.pay(fine)
goal  end-contract
```

```
exit  cancel(reserve(ticket)) => S7
      cancel(flight) => S5
end S4;

/*obl of airline in flight cancellation after travel agency paid*/
S5: obl(a,T.payback(ticket))
in    cancel(flight)
goal  T. payback(ticket) => S6
end S5;

/* obligation of airline after cancellation */
S6: obl(a,T.pay(fine))
in    T.payback(ticket), cancel(flight)
goal  T.pay(fine) => S3
end S6;

/*obligation of airline after cancellation and payment of ticket*/
S7: obl(a,T.payback(ticket))
in    cancel(reserve(ticket))
goal  T.payback(ticket) => S3
end S7;

/* obligation of travel agency after cancellation of ticket */
S8: obl(t, T.pay(fine))
in    cancel(reserve(ticket))
goal  T.pay(fine) => S4
end S8;
end-contract;
```

Although read sequentially the contract does not look complicated, section 7.4.1.2 shows that modelling deontic states and satisfaction for both parties is a complex process.

The contract does not only specify the "success line" of the interaction, but especially the exceptions. In some cases, it is possible to return to the success line. E.g., (not specified above), the airline company might have the obligation to offer an alternative flight when it cancels a flight. If this is acceptable (if the transaction succeeds), we are in state S1 again (this can be handled by the Contract Manager). This is not a true cancellation, but a "revalidate transaction". If the airline does not offer an alternative flight, or when it is not acceptable, it is no longer possible to reach the final state, and the interaction fails (the reservation transaction fails, and true cancellation is performed). Such a failure has to be handled by the Task Manager. Note that this is done only when the contract has no solutions to offer anymore.

During the execution of the interaction, other obligations may be instantiated, e.g., the obligation to pay a fine or the authorization to warrant. These obligations can be represented explicitly in the framework. However, the further processing of these "side-effects" should not interfere with the task execution. To achieve this independence, the execution of the interactions should take place in an agent environment. When the execution of the interaction leads to an obligation of the travel agency to pay a fine, this obligation is automatically put on the agent's agenda. Since by definition the agent recurrently checks its agenda and executes its tasks, the fine will also be paid in due time. On the other hand, if it is the airline company that has a liability, it is up to the agent to see to it that the payment is actually made.

A final remark on the contracts is that not all actions are specified that are used in the contract, but only the speech acts that have some deontic result (obligation or authorization). For instance, the action 'deliver' is not specified at this place.

5.3. TASKS

This section describes tasks, the task specification language TaLa (subsection 1) and also the working of the Task Manager is described in more detail (subsection 2).

A *task* is a meaningful unit of work assigned to an agent. Performing the task often involves executing some of the possible communication transactions, besides the execution of private actions. The transactions and contracts level specified so far are neutral with respect to the tasks to be executed. They only describe the *possible* messages and message sequences, as well as the deontic effects. They are the same for both parties. On top of the contract level, a task level is defined that specifies the task(s) that the agent wants to perform. The task level is not shared, the task specification and updates thereof do concern the agent in question only, whereas changes in the possible transactions (involving other agents) can only be made by consent of the other agents.

As an example consider the business trip domain. The task layer of the airline company specifies its marketing and sales efforts. At the task layer of the travel agency, we might specify the task of planning a business trip, including the decomposition into subtasks such as "flight reservation". This includes communication with the airline company (as laid down in the contract), but presumably also with hotels, clients, and banks or credit card companies.

The separation of transactions and tasks achieves a desirable level of flexibility in interoperable transactions specification. A task can be considered as a meaningful work unit which schedules appropriate sequence of transactions and actions for execution to achieve a business goal and for managing failures due to non-fulfilment of obligations, cancellation of an achieved transaction, and the usual system failures.

5.3.1. TASK SPECIFICATION LANGUAGE (TALA)

Tasks are typically described in a task language (e.g. [Nodine et al., 1994], [Rusinkiewicz and Sheth, 1994], [Georgakopoulos and Hornick, 1994b]). It typically allows the specification of tasks and subtasks, alternatives and temporal constraints. What is crucial in this context, is the way failures are dealt with. Transactions are prone to many types of failures, as described in section 5.1.2. We focus here on the failures that occur when a transaction (as one subtask) is initiated but does not commit, but also when the transaction commits first, and the resulting deontic state is violated later (e.g., the airline does not deliver the ticket), or because of cancellation (e.g., the airline cancels the flight), whereby the other party undoes a previously made commitment. These features of transactions directly influence the task specification. For a CIA, it is important that tasks are not embedded in application code, but are made explicit so that the Task Manager can use them in scheduling or rescheduling the work. Rescheduling is necessary when a subtask fails or a planned subtask becomes superfluous.

Below, TaLa (the task specification language part of CoLa) is described, and examples from the business trip domain are given. In section 5.3.1.1 special attention is paid to the contingency plan, or how failures are dealt with.

A *task* can be described using a structure similar to a transaction specification. Depending on the level of abstraction, a task may consist of a number of *subtasks*, and each subtask can itself be a task (with its own task specification) consisting of other subtasks, or it can be an elementary subtask, which has to map to a particular communication *transaction* or private *action*. TaLa allows for the specification of the *Goal* (what makes the task succeed), the *Exit* (what makes the task fail - abort), (temporal) *constraints* on the execution sequence of subtasks and transactions, and possibly *deadlines* for the subtasks. Furthermore a *contingency plan* can be given, specifying compensating and contingency tasks. When specifying contingency plans it is essential that also *dependencies* between (created results of) subtasks are specified. This is needed to specify that if result A is invalidated what other results (that are dependent on A) are to be compensated. The EBNF grammar for task specification is:

```
task :          TASK task_name taskbody END-TASK ';'
task_name :     ident
taskbody :      SUBTASKS ':' {subtask ';'}
                [CONSTRAINTS ':' {constraint ';'}]
                [DEADLINE ':' {deadline ';'}]
                [DEPENDENCIES ':' {dependency ';'}]
                [CONTINGENCY ':' {contingency ';'}]
                GOAL '=' goal
                [EXIT '=' subtask [{XOR subtask}]]
subtask :       subtask_name
          |     transaction
          |     action
          |     ACCEPT '(' trans_name ')'
```

The ACCEPT() means that this transaction is not initiated by the customer agent itself. Executing the subtask ACCEPT() implies executing that part of the transaction that the customer is supposed to do, in response to the prompt of the supplier.

Below the different elements of a task specification are worked out, respectively the goal, the constraints, and the deadlines. Contingency plan and dependencies are explained in more detail in section 5.3.1.1.

In specifying the goal of a task the following relations are defined (drawing on previous work in extended transaction models [Elmagarmid, 1992]):

Definition 5.2. (Subtask relations)

Ts = (T1 AND T2 AND ...) (task/subtask relation)
Ts = (T1 XOR T2 XOR ...) (alternative set)
Ts = (T1 XOR ... XOR **skip**) (non-vital part)

XOR (or choose) prompts for seeking an *alternative* way to make the subtask run to a successful state (similar to the concept of contingency transaction in DOM [Buchmann et al., 1992]). **skip** indicates that the subtask is *non-vital*. The task can be resumed as if the subtask has succeeded.

Also combinations of these are possible. The goal of the task corresponds to the AND/XOR formula (in conjunctive normal form). A task can therefore also be seen as an AND/XOR graph of possible subtasks.

In [Dignum and van Linder, 1996] an attempt in implementing tasks is made by giving preferences to goals. This gives the opportunity to reason over preference relations on (sub)tasks and how they influence each other. This possibility simplifies the scheduling (and conflict resolution) job of the Task Manager and is included in the goal specification:

```
goal :        goalspec [{XOR goalspec}]
              [ {AND goalspec [{XOR goalspec}] } | XOR SKIP ]
goalspec :    subtaskspec ['(' pref ')']
pref :        value
```

For the temporal precedence relations between subtasks the temporal logic specification style of transactions (section 5.1.4, definition 5.1) can be used.

```
constraint :          subtaskspec_list BEFORE subtaskspec_list
                |     ALWAYS '(' subtaskspec '=>' temp_constr ')'
subtaskspec_list :    subtaskspec
                |     '(' subtaskspec {(AND|XOR) subtaskspec} ')'
temp_constr :         (NEXT|SOMETIMES) '(' constr_el ')'
constr_el :           subtaskspec [{(AND|XOR) subtaskspec}]
                |     subtaskspec_list BEFORE subtaskspec_list
                |     temp_constr
```

(for `subtaskspec` see next subsection)

The `business-trip` case is again used to illustrate the task specification. The following example specifies the business trip task and its subtasks `flight-reservation` and `hotel-reservation` and the constraint between them:

```
task business-trip
subtasks:
      flight_reservation;
      hotel_reservation;
constraints:
      flight_reservation BEFORE hotel_reservation;
Goal = flight_reservation AND hotel_reservation
Exit = T.cancel(trip)
end-task;

task hotel_reservation = T.hotel_reservation XOR skip
end-task;
/* not a vital part */

task flight_reservation = T.flight_reservation(klm) XOR
                          T.flight_reservation(quantas)
end-task;
```

The explanation of the specific parts is as follows:

- The subtasks are either tasks or transactions specified elsewhere. Elementary subtasks are the private actions the agent can perform (e.g. database retrieval).
- The goal of the task `business-trip` is "`flight_reservation` *AND* `hotel_reservation`", even though the `hotel_reservation` is a non-vital part.

The constraint specifies that first the `flight_reservation` and then the `hotel_reservation` should be pursued.

- The main task `business-trip` contains two parts, one of which the `hotel_reservation` is non-vital. By default, the other is vital. This means that if `flight_reservation` fails and no alternative is left, the main task fails as well.
- In the case of the `flight_reservation` and `hotel_reservation` subtask specifications a short-hand notation is used in which the Goal is put immediately after the task name since there are no constraints.
- The Exit line in the main task is added for illustration purposes: if the whole trip is cancelled, then no backtracking should be done.

The alternative set of `flight_reservation` contains two transactions (distinguished from task names by the prefix `T`). In the example, only two alternative suppliers are taken into account. A better approach would be to first collect a list of possible suppliers. The `flight_reservation` subtask specification can be more convenient expressed using a dependent parameter. The parameter specification and unification mechanism and the goal-seeking procedural interpretation of Prolog suggests itself as a natural choice (cf. [Kühn et al., 1992] that describes a logic language VPL for multidatabase transactions). The `flight_reservation` specification could then look as follows:

```
task flight_reservation
subtasks:
      select_flight;
      T.get(flight-schedule);
      T.flight_reservation(flight);
constraints:
      T.get(flight-schedule) BEFORE select_flight;
      select_flight BEFORE T.flight_reservation;
Goal = T.get(flight_schedule(S)) AND select_flight(S,A) AND
           T.flight_reservation(A))
end-task;

task select_flight([Head|Tail],A)
Goal = A=Head XOR select_flight(Tail,A)
end-task;

task select_flight([],A) = XOR()
end-task;
```

On the basis of a retrieved `flight-schedule` S, the subtask `select_flight` selects one possible candidate. `T.flight_reservation` is executed with the selected candidate as a parameter. If it fails, the next alternative from the `select_flight` subtask is tried.

An example including a complex temporal constraint is the task of the travel agency in after-sales service to always check with the customer if the trip went well:

```
task after-sales
subtasks:
      follow-up;
      business-trip;
      T.complaint();
constraints:
      ALWAYS (business-trip -> SOMETIMES(follow-up);
      follow-up BEFORE T.complaint(airline);
      follow-up BEFORE T.complaint(hotel);
Goal = follow-up
end-task;
```

Also for the deadline specification the deadline specification style of the transactions (section 5.1.4) can be used. Here too the general deadline, the immediate deadline, the as soon as possible deadline and periodic deadline can be specified.

```
deadline :      ['P'] [condition '<<'] goal '<<' condition
           |    goal '!'
           |    goal ASAP
condition :  proposition | subtask | time | subtask '+' time | now '+' time
```

An example of a deadline specifying something should be done before a condition holds from the business-trip case is where a hotel must be paid up front, and within 15 days after the reservation is made:

```
task hotel_reservation
subtasks:
      inquiry;
      ACCEPT(quotation);
      make_reservation;
      ACCEPT(reservation);
      payment;
constraints:
      inquiry BEFORE make_reservation;
deadline:
      payment << ACCEPT(reservation) + 15 days;
Goal = ACCEPT(reservation) AND payment
end-task;
```

The normal constraint `ACCEPT(reservation) BEFORE payment` is incorporated in the deadline specification.

A more complex example that shows combinations of different type of deadlines in the case of the travel agency is the following.

In order to apply for a special rate. a reservation should be made between the first of November and the first of December. The reservation should be accepted within a week after the request for it and the ticket should then be paid as soon as possible:

```
task special_rate_reservation
subtasks:
      T.flight_reservation(klm);
      ACCEPT(reservation);
      payment;
constraints:
      ACCEPT(reservation) BEFORE payment;
deadline:
      date(Nov.,1) << T.flight_reservation(klm) << date(Dec.,1);
      ACCEPT(reservation) << T.flight_reservation(klm) + 1 week;
      payment ASAP;
Goal = T.flight_reservation(klm) AND ACCEPT(reservation) AND payment
end-task;
```

In section 3.1 we have seen that the business logic explicitly includes the problems and objectives of the customer and the capacity of the supplier. By embedding the transactions in an agent task, the link to the objectives are now explicit, and relationships with other tasks of the customer company can be described. As far as the supplier is concerned, Goldkuhl makes no reference to objectives, only to capacity. The symmetry

between the parties the author argues for is not maintained at this point. In my point of view, the supplier has objectives as well, including producing goods and making money. These can be modelled as supplier tasks. On the other hand, the capacity and know-how do play a role, as constraining factors, but so do they at the customer side, in particular, the financial capacity or liquidity. Therefore we propose a symmetric treatment of supplier and customer *also* as far as objectives and constraints are concerned. The particular task of the supplier can be modelled as follows:

```
task sell
subtasks:
      ACCEPT(get_quotation);
      give_quote;
      ACCEPT(order_product);
      delivery;
      ACCEPT(payment);
constraints:
      give_quote BEFORE ACCEPT(order_product);
      ACCEPT(order_product) BEFORE delivery;
Goal = ACCEPT(payment)
end-task;
```

5.3.1.1. CONTINGENCY PLAN

An important part of a task specification is the contingency plan. A notorious problem with contingencies is that when a subtask is invalidated, later (dependent) subtasks may already have committed, and their result might become obsolete. Whether they have to be retried (after being compensated) or not depends on the kinds of results they produced. The designer should be given the opportunity to specify a separate contingency plan.

Different kinds of contingencies can be distinguished. First, it may be that a certain action has to be *undone* (compensation). E.g., a hotel reservation has to be cancelled because the flight schedule has changed. Since in long-lasting interoperable transactions a simple rollback is no longer possible, the action has to be compensated, in this case by cancelling the hotel reservation. Secondly, it is possible that a certain action has to be *redone*. E.g., a flight is booked and linked to a bus schedule from airport to city. When the flight schedule changes, and another airport is selected, another bus schedule must be retrieved. The Task Manager monitors the execution of the tasks. This means it also keeps records of how far a task is in its execution (which subtasks are already executed and which results are created for it, and which ones still have to be executed).

A possible contingency part for the business trip example specifies that if the `flight_reservation` is cancelled by the other party, the Task Manager has to find an alternative (as if `flight_reservation` had failed). This will involve finding another supplier. Any contractual matters that need to be resolved (e.g., collecting a fine) are not specified here, since they are handled by the contract between the agent and the airline company (section 5.2). However, dependent subtasks, e.g. `hotel_reservation`, if any, are cancelled (or if already succeeded, compensated) by the Task Manager in accordance with the contingency plan.

The CIA contingency plan consists of a set of *results* (e.g. bus schedule, ticket, reservation) that have an internal object structure, and associated methods that specify:

a) the transactions (or tasks) that can be used to create the result;

b) the transactions that can be used to close the result, meaning that after this (trans)action the result cannot be invalidated anymore;

c) the transactions that can be used to compensate the result; and

d) transactions that invalidate (and possible revalidate) the result.

This is motivated by the recognition that tasks can be of two kinds: procedural tasks that specify a set of actions to be done, such as producing different beeps, and object-oriented tasks that create or work on a certain object. Pure procedural tasks cannot be compensated, so we do not need to bother, whereas compensating or revalidating object-oriented tasks means compensating or revalidating the result object. These actions are most conveniently encapsulated in the result object structure.

The grammar of a result with its methods is:

```
contingency :       RESULT result
                    CREATED-BY subtask_name
                    CLOSED-BY action_spec
                    UPDATED-BY action_spec
                    INVALIDATED-BY action_spec [{ ',' actionspec }]
                    [REVALIDATED-BY action_spec]
                    COMPENSATED-BY action_spec
                    END-RESULT
action_spec :       transaction | action
```

If a result becomes invalidated a task can be triggered to repair the damage. This task can make use of the fact that all the essential results obtained so far (and not invalidated) are explicit. E.g., in the case where a hotel-reservation is dependent on the flight-reservation, and the flight is cancelled, the contingency plan can try to repair the damage by trying another airline. Only if that fails the hotel-reservation has to be cancelled (or compensated if it already succeeded). An example of a contingency plan for the results in the case of the business trip is the following:

```
result airline-ticket
created by flight_reservation      /* a task, see above */
closed by T.boarding
updated by T.change_flight
invalidated by T.cancel_flight
compensated by T.cancel(reserve(ticket))
end-result;

result hotel-reservation
created by hotel_reservation
closed by hotel_stay
updated by T.change_date
invalidated by T.cancel_room
compensated by T.cancel(reserve(room))
end-result;

result bus-table
created by T.get_bus-table
closed by T.boarding
updated by T.get_bus-table
compensated by skip
end-result;
```

The **created by** line of the result object specifies a task that creates the result. The task is not only called when the result has to be created for the first time, but also when it is to be revalidated. The **closed by** line indicates the task or transaction that freezes the object. This is typically done when the corresponding event has taken place, e.g., when the flight has been taken.

Since we want to allow for freedom in task specification the result objects can be used polymorphically. E.g., in the `business-trip` task, `flight_reservation` can be replaced by `airline-ticket`.

In the grammar this is expressed by the rule:

```
subtaskspec :      result | subtask
```

Executing `airline-ticket` means executing its **create** method. In general, we distinguish the following object operators: **create**, **close**, **compensate**, **update**, and **fail**. In the example of `airline-ticket`, **compensate**`(airline-ticket)` is equivalent to `T.cancel(reserve(ticket))`, and so on.

Essential in task specifications with contingency plans is the specification of dependencies. The dependencies specify other results need to be compensated if a result is invalidated. E.g., a `bus-table` is dependent on an `airline-ticket`, in the sense that a change in the latter has a bearing on the former. Various dependencies can be distinguished. At the moment we distinguish: `create`, `update` and `fail`. A `create` dependency between `A` and `B` means that if `A` is created, automatically `B` is created (by means of the task **create**`(B)`). A `fail` dependency means that if `A` fails, so does `B` and **compensate**`(B)` is triggered to achieve this. An `update` dependency means that if `A` is modified, so is `B` (by means of **update**`(B)`).

The object operators can be used in the specification of these dependencies as constraints on the level of the task (we could have specified the dependencies at the level of the result objects, but this will lower the reusability of these objects, so we prefered to specify them on task level).

```
dependency : result DEPENDS-ON result '('[CREATE] FAIL [UPDATE]')'
```

The new task specification of business-trip can then be as follows:

```
task business-trip
subtasks:
      airline-ticket;              /* = create(airline-ticket) */
      hotel-reservation;           /* = create(hotel-reservation) */
constraints:
      airline-ticket BEFORE hotel-reservation;
dependencies:
      bus-table depends-on airline-ticket (create,fail,update);
      hotel-reservation depends-on airline-ticket (fail,update);
Goal = airline-ticket AND hotel-reservation
Exit = T.cancel(trip)
end-task;
```

(Note that the dependencies defined here could be implemented straight-forward by means of database triggers (ON MODIFY etc.) in a conventional DBMS).

We can now describe what happens when an airline-ticket has been issued (created), but the flight is cancelled: The transaction `cancel_flight` (initiated by the other party) is executed and invalidates the `airline-ticket` object, as specified. As a result, the `airline-ticket` object will try to revalidate itself by choosing the next alternative in its **`create`** task, that is, in `flight_reservation`. If this succeeds, parameters may have changed, thus causing an update of the object itself and (via the `update` dependency) of the `hotel-reservation` and `bus-schedule`. If the revalidation fails, the result fails and the other fail-dependent objects are forced to fail as well, which means the execution of the compensation task, if any.

If the **`create`** task of a result ends with a **`skip`** option, the result is not vital. Such a result can never fail. So the invalidation of a result object can have various effects. If an alternative is found locally, no other result or subtask needs to be aborted. If it is not found, and the result is vital, only those results are forced to fail that are said to be dependent. This makes the contingency handling quite flexible.

It is also possible to allow the user to specify contingency plans at the object level that override the default "next alternative" search algorithm,. E.g., the user may want to specify that in the case of a cancellation, the same airline company should be requested for an alternative flight (this may be agreed upon in the contract), and not the next alternative airline company. To allow for such specifications, an optional **`revalidated by`** method can be added to the object specification. This method would overrule the default of proceeding with the **`create`** method.

We should keep in mind that specifying contingencies is difficult, since it is impossible to foresee every problem and every context. For that reason, contingency plans are not obligatory; dependencies can be given, but if they are not specified it is assumed that the other results are not affected. A contingency plan can be given, but if it is not specified, the default search proceeds.

5.3.2. TASK MANAGER

The role of the Task Manager is twofold. First, the Task Manager can be considered as the runtime module where users submit their task specification. This includes planning (and scheduling the subtasks, transactions and actions). The Task Manager processes the specification. Second, if a subtask fails, an alternative must be sought and dependent invalidated subtasks must be compensated. The user can specify a contingency plan that the Task Manager can execute. In other words, the contingency plan at the level of the tasks is what the contract is at the level of the transactions. Violation of commitments (to other parties) are being dealt with by the Contract manager, while compensation of subtasks is handled by the Task Manager.

The execution model of the Task Manager is as follows:

- when a task is called, the Task Manager collects the subtasks, puts them into a right order according to precedence relationships (such an ordering can be stored to save computing time) and puts them on the agenda The subtasks are then called one by one. This step continues recursively.
- when a subtask cannot be fulfilled, the next alternative is chosen. If no alternative is available, it means that the subtask fails. It does not mean directly that the parent is aborted, since it is possible that the subtask can still be fulfilled by choosing an alternative in the preceding subtasks. This is tried first (backtracking). Only when no alternative is left, the parent fails. The parent also fails when an exit state is reached.
- when an exception occurs (a subtask is invalidated), the contingency plan (if specified) will be executed first. Otherwise, the next alternative is taken, or, if no alternative is left, backtracking is attempted. To keep the task specification simple, we give the designer the opportunity to specify a contingency plan separately. If no contingency plan is specified, and no alternatives for backtracking are left, the task fails, but its parent may have alternatives or contingency plans of its own. Every time a task fails, the Task Manager goes up one level in the task hierarchy and the process is repeated.

The execution model enforces a "structured approach" in the task specification. It does not allow for arbitrary abort or commit dependencies between subtasks over the boundaries of the parent tasks. This modularity is enforced to keep the specification transparent and maintainable.

Below this model is refined. First the planning process is described and then the agenda knowledge base is described in more detail. In appendix B high-level algorithms for these processes are given.

5.3.2.1. PLANNING

An important job of the Task Manager is planning. Several planning strategies can be implemented, they are however not the focus of this work. Planning is difficult, since the best plans will be made when all information is available, but has the hazard they cannot be produced in time to execute them. Early planning often leads to replanning (or abandonment) if more information comes available (e.g., about the execution and failure or result of other subtasks). But as [Thomas, 1993] argues, "as long as plans can be retracted, and as long as little or no otherwise-useful time is expended in earlier-than-necessary plan refinement, and as long as the decision to retract a plan can be made quickly and appropriately when new information is received, there is no reason not to go ahead and plan".

In this thesis a simple strategy is used (see below) and the general problem of deciding when to decide is ignored. The retraction of plans is more problematic. The Task Manager should include special-purpose reasoning to detect when new information requires the agent to give up an already-defined plan and re-plan for the same goal. If execution of a

particular plan already has started then the plan cannot simply be retracted in its entirety. However, the contingency plan can be consulted for what and how already instantiated results can be compensated, and also what should happen if the goal cannot be reached anymore.

In this work a naive planning strategy is adopted, in which we start with the presupposition that there are unbounded resources, i.e. unbounded computing resources. Of course, in case of a hotel, the number of rooms is bounded. All actions can be performed in parallel (if no order or temporal constraints exist between them) and all actions take the same amount of time to execute. The agent should try to perform all actions (fulfil all obligations) before the deadlines associated with them, so it should plan the action whose deadline expires first as the first action to perform[1].

To be able to order the items on the agenda precedence relations or priorities can be specified. This is needed for some form of risk analysis, e.g. reservations for the flight and hotel can be made in parallel, however if the flight reservation does not succeed, the hotel reservation becomes useless and has to be stopped, or cancelled if already succeeded, and should therefore be delayed. It is also important for conflict resolution in case of conflicting activities (e.g. conflicting obligations). If there is an obligation $O(\alpha)$ that says α should be done, and another obligation $O(\overline{\alpha})$ that states that α should not be done (a prohibition $F(\alpha)$, α is forbidden), than it depends on where the obligation resulted from. The simple strategy here is to give obligations resulting from contracts priority over obligations resulting from tasks. This follows from the cooperating nature of the agents that gives high priority to the maintenance of business relations and adhering to the general business logic (or as said before: communicative action rather than strategic action). Furthermore, the agents are designed to be "sincere", i.e. they will try to hold their promises (commitments). As a result the goal might not be possible to reach and the task execution is aborted (it fails). An agent should not drop its goal easily, but should try to salvage it through re-planning. Of course, it is required that an agent's plans are sufficiently refined to enable the agent to achieve its goal. If an agent's goal can only be achieved if the agent begins acting now, it must at least know what its first action should be, and plan to do that action (now or at least as soon as possible).

5.3.2.1.1. Agenda

In this subsection the agenda is described in more detail, using the popular agent concepts: plans and intentions.

As with the definition of agent itself there are many interpretations of what plans and intentions are. Here my own interpretation is given, and these concepts are related to obligations and goals as introduced in this thesis. The ideas presented here can also be used as input for future research in which a formal agent theory along the lines of the BDI framework is formulated. A first attempt to do this has been described in [Dignum and van Linder, 1996], [Dignum, 1996], [Dignum et al., 1996b], although some concepts differ slightly.

[1] From Operational Research (and personal experience) we might deduce that this does not always lead to optimal (or even acceptable) plans, unfortunately.

As described in section 4.3.1, the agenda defines the normative space (or obligation and commitment space) of all things that have to be done. Based on the formal framework (chapter 6), the obligation is the primitive concept, and the other concepts are related to that. The agent in its actions is guided by three things:

- social *obligations*, or the things he has to do because he committed himself to others to do, as described in the *contracts*;
- private *goals*, or the things he wants to achieve, as described by its *tasks* (these can also be seen as commitments to itself, along the lines of [Shoham, 1993]);
- necessities, things he has to do in order to keep the database(s) consistent.

This division is guided by Habermas distinction of social, subjective and objective world, respectively. This division of the three worlds can also be used to describe the reasons of an agent **not** to perform an certain action. First there are the impossibilities in the social world, called prohibitions (as opposed to the obligations in the necessity situation) following the norms of the world the agent operates in. Secondly, the impossibilities of an agent derived from the subject world are those that are in conflict with the agent's own values (the things you don't want to happen). Thirdly, we have the impossibilities in the object world (it is just not possible, the agent cannot do it).

Following [Shoham, 1993], an agent is designed to be able to:

- request anything from anyone (including authorization to do something)
- inform anyone of a fact it beliefs or knows
- inform itself (this is useful to implement reasoning in the agent)
- declare a fact (if it has the authority or power to do so)
- commit itself to perform some action
- perform any private action (if the static/dynamic integrity constraints hold)
- perform a service on the request of another agent (including replying to messages)

However, it is possible the agent does not know how to perform a requested action (it is not a serviceof the agent, and it does not know where to obtain it).

Related to these concepts is the concept of achievability. Something is *achievable* if it has to be done (obligated, wanted, necessary) *and* is possible in all worlds. In the approach presented here the (im)possibilities and desires from the subject world are left out. In [Dignum and van Linder, 1996], [Dignum, 1996] those are specified by *wishes* of the agent: it should have both the capability and wish to establish a certain goal, before acting.

Both tasks and contracts have goals defined. In order to be able to reach a goal or fulfil an obligation plans have to be constructed. A *plan* is defined as a precise order of steps (actions or the initiating of transactions) that lead to the goal. In tasks this is specified by the subtasks, actions and transactions and the constraints (including deadlines) between them. As we have seen in section 5.1.4 a dependency graph can be set up which describes all possible paths that lead to goal and exit states. This ordering of actions etc. also has to be done for all subtasks specified. These plans can be stored for later reference and reuse

(e.g., the ordering subtask has a fixed plan that does not have to be 'calculated' each time an ordering subtask is specified, only the correct parameters should be substituted).

The goals of contracts are mainly expressed by transactions (see the examples) and therefore have a simple plan. In case a more complex goal is formulated the plan database is consulted whether it contains a plan for the task with this goal. If not a new "subtask" is formulated based on the knowledge of its private actions and results of communicative actions. This new subtask should be distinguished from the subtasks derived from the task specification of the agent, since it comes from an obligation described in a contract.

Two things the planner must do before the plan is submitted are:

- check if the agent can perform the actions specified in the plan, in other words: if it is achievable (no conflicts between things it has or wants to do, but cannot do because of lack of authorization, or incompetence). If this constraint cannot be met it means the goal cannot be reached by this plan and a failure is reported.
- check whether the goal already has been reached, e.g. by a previous (trans)action. A success can be reported and the planner can proceed with the planning of dependent actions. Instead of submitting the original plan execution now the result should be monitored to see if it is not invalidated by other (trans)actions.

After the formulation of plans (or choosing an already existing plan) for both obligations and (sub)tasks a *decision* is made which plans to place on the agenda.

As described in the previous subsection this can be a complex process. A simple strategy is adopted here: the agent should try to perform all actions (fulfil all obligations) before the deadlines associated with them, so the plan whose deadline expires first is chosen as the first action to perform and put on the agenda. The choice made can have repercussions for other plans. The tasks are not yet abandoned, since at a later stage the prohibition might not hold anymore (e.g., the agent becomes authorized to perform the action). Only if the deadline cannot be met, or the goal cannot be reached remedial actions (re-planning or looking for alternatives, with associated new plans) should be taken. In case one obligation from a contract is chosen over another this means that (although the agent tried) there is a violation (the other obligation cannot be met) and the contract manager is notified to see what should happen. Further choices between the plans are made on the basis of priorities specified and the strength (e.g. is the obligation the result of a power, authority, or charity relation).

The decisions can be informed by the ideals of the agent ([Dignum, 1996]). *Ideals* are higher order goals that provide a motivation for the choice and the ordering of the plans. Ideals are personal goals and say something about (and provide higher order strategies for) the desired behaviour of the agent. The ideals incorporated in the agents proposed here are "being communicative" (as opposed to strategic) and "being sincere" (obeying commitments, and mean what is communicated). As a result the agent tries to fulfil all obligations and commitments to others and itself.

The distinction between plans and agenda can be illustrated with the following example: suppose the travel-agent has a deadline to order a ticket after it has been paid for by a client, and it should be done before the end of the month. Although this yields an

obligation for the agent it is of no use to put this on the agenda yet, since it is impossible to know when the ticket will be paid for exactly. Instead a plan can be formulated how to order the ticket before the second deadline that can be executed as soon as the payment is in. A trigger can be placed in the database that signals this event and then the plan can be placed on the agenda for execution, which should be done before the end of the month (which again can be signalled by a trigger).

When using the general form of this obligation (deadline) `condition1 << action << condition2` as given in the TaLa grammar above (or formally $O(\phi < \alpha < \psi)$ as will be defined in section 6.3.3) the agent can be notified of `condition1` holding in three ways. First if `condition1` is a constraint on the agent's database a trigger can be defined. Secondly, if `condition1` is a time it is a function of the clock to trigger the action if the time has passed. The third way is if `condition1` cannot be controlled by the agent but is an external factor. Only when the agent receives a message (from an agent j) that `condition1` holds (formally: $[ASS_{ji}(\phi)]O_i(\alpha<\psi)$) actions can be taken. In fact what happens is that a new deadline is specified (`action << condition2`) and the plan for this can be put on the agenda (if the obligation does not conflict with others).

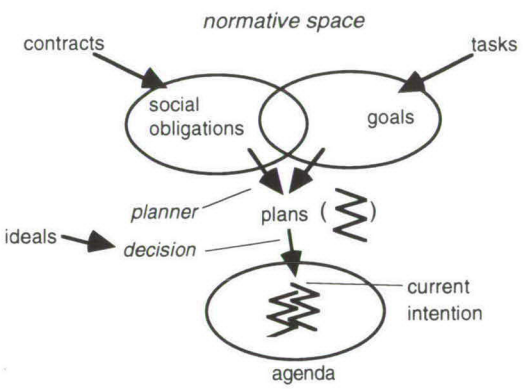


figure 5.4. Agenda, obligations and plans

The *agenda* (object) thus holds the plans it has decided to perform (and also has to and is able to perform). Here these are called the *intentions* of the agent. The items (actions) on the agenda are ordered according to their deadline and priority. We can conceive the agenda as containing interleaved plans of actions. The current intention points to the action on top of the stack (or front of the queue). Figure 5.4 gives a graphical overview in which the concepts are related.

The execution model is now a loop containing the following steps:

- execute the first agenda item (current intention);
- after every step see if the environment has changed. This includes processing of incoming messages, and responding to triggers and passing of time (clock ticks);
- check if the goals are still achievable (or, if the plans still lead to the goals);

if not: remove plans (the thread of actions) from the agenda, notify the failure and look for alternatives (re-plan), or in case of a failure to adhere to an obligation from a contract, notify the contract manager of the violation, otherwise continue;
- check if already obtained results are invalidated, and if so trigger the contingency plan (if specified);
- check if new plans can be made because of the input, trigger the planner.

5.4. COMPARISON WITH OTHER TRANSACTION MODELS

This chapter described how we distinguish between the specification of communication aspects, the scheduling aspects and the failures aspect of interoperable transactions. This not only achieves extra flexibility, but also provides a structured and formal approach to specifying interoperable systems. We introduced the concept of "contracts" for specifying and managing failures due to violations, and "tasks" for managing the failures due to cancellation and execution strategies (such as backtracking, firing of contingency plan etc.). This section compares this approach to others for specifying (interoperable) transactions and workflows and describes what the advantages of our approach are.

In [Alonso et al, 1996] a comparison is made between advanced transaction models and workflow models. It is argued that transaction models are too much centered around databases, too theoretical and that (in many respects) workflow models offer a superset of transaction models. The paper describes the FlowMark workflow model and how it can be used to emulate advanced transaction management. This implies, as is argued also in this chapter, that ACID properties, in particular consistency, should no longer be viewed as technical issue, under the responsibility of a global or local transaction manager, but as organizational issue, directly having to do with the responsibilities of human agents. In contrast to Flowmark, the model presented here is based on business process models and the LAP. As a result, the notions of obligations and authorizations are made explicit. The model abstracts from information flows and goes to the business essentials, what is actually achieved by communication.

The Flexible Transaction Model ([Elmagarmid et al., 1990]) allows specification of work flow control in the form of intra-transaction dependencies (where classical transaction model deals only with inter-transaction concurrency) and success/failure conditions, but the verification of such dependencies is not supported. Therefore, a consistent and correct specification is expected to be the input to the underlying rule-based scheduler, based on first order temporal logic. Another difference with the FTM is that the basic units in the approach presented here are communicative messages (speech acts) while theirs are (database) operations. It is for that reason that here the deontic effects of actions can be modelled, since the semantics of messages are limited while operations can do anything.

In comparison with the Interaction model ([Nodine et al., 1994]) and ConTracts model ([Wächter and Reuter, 1992]) the failure handling presented here uses backtracking as well, but is goes further in two respects. First, later (dependent) subtransactions are not aborted

immediately, but only when the corresponding result cannot be maintained. The contingency plan may also provide specific alternatives; if these succeed, no abort or backtracking is necessary. Secondly, in our model focus is on the notion of compensation in two ways: compensation of the other party as specified in the contract, and compensation or contingency handling of a result. By separating task and contract to handle different aspects of failures, we are able to treat task management as a local issue, as contrasted to being global as in Interaction. In this way, the global control is kept to a minimum, which makes the specification and implementation much more flexible.

In the Action Workflow [Medina-Mora et al., 1992], commitments are explicitly modelled by speech acts. However, it does not support reasoning over deontic effects, thus violation of commitments cannot be enforced. Neither does it support (and distinguish) the specification of tasks. Nevertheless, Action Workflow could be useful as a method for structured analysis of the transactions and contracts (see chapter 7).

Summarizing, our framework has the following advantages:

- Although the task specification is closely related to other extended and flexible transaction models (cf. [Elmagarmid, 1992]), none of them attempts to model the effect of transactions in terms of obligations and authorizations. This means that transactions can only be classified as either a success/failure in those models; it is not possible to enforce and reason over violation of obligations or to re-adjust the original goal to avoid the failure.
- Communication specification can be done using the most suitable formal framework that can be verified. E.g., transactions can be specified with propositional temporal logic, which lends itself to a well-proven verification method. Transactions are also specified as having a deontic effect, which is in accordance with the effects of messages in the transaction. This means that operational semantics of messages can be given formally. The specification makes all failures (including violation of commitments) explicit. It is easy to verify whether each failure state is handled properly, i.e. whether it is always possible to arrive in a non-failure state a-priori.
- The framework supports efficient re-design of interoperable systems as compared with for instance [Elmagarmid et al., 1990] and [Georgakopoulos et al., 1994]. E.g., when business-needs change, the only new component that needs to be redesigned could be the task execution strategies. The contract can remain stable. The reverse is also possible. The task remains the same, but the communication partners change, and hence the contract(s).

CHAPTER 6

FORMALIZING THE COMMUNICATION FRAMEWORK

This chapter describes the formal framework underlying the specification language and modelling techniques. The formal framework integrates illocutionary logic of formal communication between automated systems with the dynamic deontic logic used to specify obligations and authorizations over actions. Special emphasis is put on the dynamics of authorizations and obligations, i.e. how authorizations are created and deleted and how they influence obligations among agents. Parts of the formal framework have been described by others and myself before (see below).

The first section clarifies our choice for the basic logics used and gives an introduction in Deontic Logic. In section 2 the focus is on the basic (dynamic) action logic, describing the basic logical languages with their semantics. Section 3 describes how the dynamic deontic logic can be extended with illocutionary logic, giving the illocutionary dynamic deontic language L_{ill}.

6.1. INTRODUCTION AND MOTIVATION

This section introduces the reader to the logics that are the basis for the formalization: deontic logic, and its reduction to dynamic logic.

In this thesis a language-action perspective is taken to communication modelling and the design of CIAs. The specification of an agent includes the specification of the possible messages it can exchange with other agents. The focus here is not on information exchange but on what agents do while communicating and how they influence each others actions by using language. The basic units of communication (messages) are seen as speech acts. Illocutionary logic is used to describe these speech acts formally. Since we want to formally describe what the effects of actions and transactions (groupings of speech acts) are, dynamic logic (the logic of actions) is used. One of the effects a (speech) action can have on an agent is that obligations (to act, or to obtain a certain state) are created. Since actions of others cannot always be enforced (e.g., the other agent might decide not to perform an action that is requested) it should be possible to state that a certain action should take place without getting an inconsistency when the action does not

take place. Furthermore it is important to look at when actions (including speech acts) may or may not be performed. Deontic logic is the logic that describes the obligations and authorizations of actions, and makes it possible to reason about violations of the obligations and authorizations. The use of deontic logic makes it also possible to specify deadlines, the combination of deadlines and the detections of inconsistent deadlines (deadlines that cannot be kept jointly).

The logic can be used to model the norms that result from the communication between agents in a normative system. The combination of illocutionary and dynamic deontic logic can be used to build communication protocols or contracts, that define the subsequent steps in the communication and how these steps are related, i.e. which are the allowed reactions to a certain action. The contract between two agents describes the interactions and their effect, i.e. the factual or deontic temporal constraints that are created or deleted.

The aim is to describe the whole business communication protocol using the logical formalism, making it possible to reason about properties of the protocol. For instance, to prove that, given certain preconditions, after the order protocol has been followed one party has the obligation to deliver a product while the other party has the obligation to pay for it. The use of deontic logic within the protocol also makes it easy to describe the course of action in the cases that the communication protocol fails.

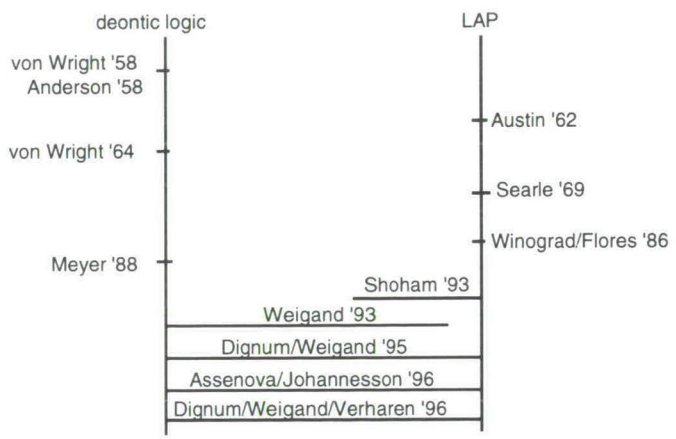


figure 6.1. Historical perspective on combining deontic logic and the Language-Action Perspective

Figure 6.1 gives an historical perspective on work that influenced the research described in this thesis. On the left side, work on deontic logic (described in more detail in the next subsection), and on the right side the Language-Action Perspective (especially speech act theory, including illocutionary logic). Shoham was the first to include speech acts and deontic concepts like commitment and obligation resulting from them in specifying the communication between agents. The application of deontic logic to communication modelling was first suggested in [Weigand, 1993]. Communication aspects have been described in [Dignum and Weigand, 1995a], [Dignum and Weigand, 1995b], and

[Weigand et al., 1995], and semantics and multimodal logics for communication have been described in [Wieringa et al., 1989], [Weigand et al., 1995], and [Dignum et al., 1996a]. The use of linguistics for ISs was described in [Weigand et al., 1996].

A similar approach to the one presented here is described recently in [Assenova and Johannesson, 1996]. They give a first order action logic (FAL) that includes deontic and illocutionary constructs for the modelling of communication between agents. Their approach differs from ours in that they explicitly represent time (although in [Dignum and Kuiper, 1996] the approach presented here is extended to also include explicit references to time). The FAL approach can be given a simple logic programming semantics and has a relatively simple implementation. On the other hand, they lose the possibility to reason over the obligations and authorizations, since they do no describe axioms.

6.1.1. DEONTIC LOGIC

For the semantics of communication models, we use Dynamic Deontic Logic ([Meyer, 1988], [Wieringa et al., 1989]). Here only a short overview of this logic is given. The interested reader is referred to [Meyer, 1988] for a comprehensive description.

Deontic logic is a branch of modal logic that is concerned with (reasoning about) norms and normative versus non-normative behaviour. Deontic means 'as it should be'. Typically deontic logic has operators for deontic/normative modalities such as permission obligation, and prohibition. Originally used in philosophy to formalize (reasoning about) notions in ethics and philosophy of law, recently it has become apparent that deontic logic can also be useful in certain areas of computer science and AI. It provides an adequate tool to describe integrity constraints of ISs (e.g. [Wieringa et al., 1989]), particularly those dealing with obligations and prohibitions, but also in other applications as diverse as office systems (e.g. [Kimbrough et al., 1984], [Lee, 1988a,b]), fault-tolerant systems (e.g. [Khosla and Maibaum, 1987]), security policies (e.g., [Minsky and Lockman, 1985], [Glasgow et al., 1989]), and of course, in legal expert systems (e.g., [Sergot, 1990], [Jones et al., 1979], [Biagioli et al., 1987]). [Wieringa and Meyer, 1993] describes the application of deontic logic to organizations, which can be divided into two groups:

- *policy specification*. Deontic logic is used to prescribe behaviour of organization components and the ISs that support it. The policies are guidelines for behaviour that are meant to be followed up by whomever they are directed at. Deontic logic can be used to make the policies unambiguous and to explore the consequences of different specifications. In many cases, policy designers are interested in what would happen if their policies are not followed up. The system is not to break down in the situation something occurs that does not follow regulations. Instead we want to be able to reason about it and take appropriate action.
- *normative organization behaviour*, which prescribes the behaviour of the organization in its environment using deontic logic. The analysis of contracting given in section 5.2 is an example of this. Because the behaviour of an organization should comply with the law, this can be viewed as a special kind of legal application of deontic logic.

Also [Jones and Sergot, 1993] indicates the value of adopting the normative systems perspective and illustrates the role of deontic logic (and also that of a logic of action) in the formulation of models of computer systems.

There have been many attempts to try and capture deontic notions in a formal manner and describe axioms and rules for deontic logic. Historically, the first was by Ernst Mally ([Mally, 1926]), followed by the system of von Wright ([von Wright, 1951]) (called Old System). The first axiomatized deontic logic system that has been widely accepted is the Standard Deontic Logic of von Wright.

In standard deontic logic the notion of deontic primitive 'obligation' is captured by a monadic modal operator 'O'.

'Op' is read that p is obliged. The others (P 'permission', and F 'prohibition') are defined below. The following axioms and rules (called the standard deontic logic system, or KD, because of the prominent place of the K- and D- axioms) are taken from [Meyer and Wieringa, 1993] (rules and derived rules are in the form $\varphi_1, \varphi_2 / \psi$):

(KD0)	All (or enough) tautologies of Propositional Calculus	
(KD1)	$O(p \supset q) \supset (Op \supset Oq)$	the K-axiom
		"Obligation is closed under logical implication"
(KD2)	$Op \supset Pp$	the D-axiom
		"Obligation implies permitted"
(KD3)	$Pp \equiv \neg O \neg p$	"Permission is the dual of obligation"
(KD4)	$Fp \equiv \neg Pp$	"Forbidden is not permitted"
(KD5)	Modus Ponens: $p, p \supset q / q$	
(KD6)	O-necessitation: p / Op	

(KD6 must not be confused with the (invalid) assertion $p \supset Op$. It states that if p is established as a theorem, then we may also derive Op).

As with other modal logics, the semantics of the standard system is based on the notion of possible worlds. Given a Kripke-model $M = (S, \pi, R)$ and a world $s \in S$ we give the following semantics to the modal operators:

$(M, s) \models Op$ iff $\forall t\ [R(s,t) \Rightarrow (M, t) \models p]$
$(M, s) \models Pp$ iff $\exists t\ [R(s,t)\ \&\ (M, t) \models p]$
$(M, s) \models Fp$ iff $\forall t\ [R(s,t) \Rightarrow (M, t) \not\models p]$

The operator O is treated as the basic modal operator: for Op being true in world s we have to check whether p holds in all alternatives of s, as given by the relation R. This reflects the idea that something is obligated if it holds in all (ideal) worlds (relative to the world where you are). The operator P is the so-called dual of O: Pp is true if there is some alternative world where p holds. The F-operator is very similar to the O-operator: something is forbidden (in a world) if it does not hold in any alternative (to that world). Validity is as usual in modal logic: formula p is valid with respect to a class C of Kripke-models, denoted $C \models p$, if p is true in every world of every Kripke-model in C, that is, $(M,s) \models p$ for every $M = (S, \pi, R) \in C$ and every $s \in S$. One can prove that the system KD is sound and complete.

The theorems derived from KD contain a number that are considered paradoxes, since they are counter-intuitive to a greater or lesser degree (see [Castaneda, 1981] and [Meyer and Wieringa, 1993] for a list). Many researchers proposed systems to overcome these difficulties. Von Wright himself proposed the New System ([von Wright, 1964]) to capture conditional obligations (which later in [von Wright, 1965] was amended by himself). In the mean time Anderson [Anderson, 1958] proposed to reduce deontic logic to alethic modal logic, i.e. without deontic content, except for a special propositional atom V, indicating a liability to some sanction or punishment. Others have proposed to incorporate aspects of time to overcome some of the difficulties with 'traditional' deontic logic. [Thomason, 1981] even argues that deontic logic requires a foundation in temporal logic, reducing the obligation of p to a temporal statement that p holds in all future courses that would be morally acceptable. [van Eck, 1982] enriches deontic logic with temporal operators, and [Fiadeiro and Maibaum, 1991] propose a (semantical) reduction to temporal specifications.

6.1.1.1. REDUCTION TO DYNAMIC LOGIC

As von Wright already states in [von Wright, 1963, 1968] the sense of the deontic operator relies on what is meant by an *action*, or *doing* something. What is under a deontic predication is a statement of an action. Oφ should be read that a certain agent is obliged to *do* φ. [Castaneda, 1981] also points at the fact that many of the paradoxes of deontic logic could be solved by distinguishing propositions (assertions) from actions (practitions).

Inspired by the observations of obligations on actions and Anderson's unsatisfactory reduction to alethic modal logic, Meyer ([Meyer, 1988]) proposed a reduction of deontic logic to propositional dynamic logic, using Anderson's violation atom V, indicating that in the state of concern a violation of the deontic constraints has been committed. In Meyer's approach the concern is with, what is called in deontic literature, "*Tunsollen*" rather than "*Seinsollen*" (cf. [Von Wright, 1980]). [Meyer and Wieringa, 1993] gives a good introduction of the reduction of deontic logic to dynamic logic.

Propositional dynamic logic ([Harel, 1984]) consists of the normal propositional logic extended with modal operator $[\alpha]$ for every action α in the language. An expression $[\alpha]\varphi$ is read as "the performance (execution) of action α leads necessarily to state (possible world) in which φ holds". The formal semantics is given by means of a Kripke structure where there are accessibility relations R_α associated with each action α. In this approach, α is forbidden (F), α is permitted (P) and α is obligated (O) are reduced to dynamic expressions as follows:

$F\alpha \equiv [\alpha]V$

action α is forbidden iff the performance of α yields a state where V holds;

$P\alpha \equiv \neg F\alpha \ (\equiv [\alpha] \neg V)$

action α is permitted iff α is not forbidden (iff there is some way to perform α that leads to a state where V does not hold);

$O\alpha \equiv F(\text{—}\alpha) \ (\equiv [\text{—}\alpha]V)$

α is obligatory iff not-doing α is forbidden (iff not-doing α leads to a state of violation (V holds)).

(although the negation of the action α, denoted —α, expressing the non-performance of this action, may be intuitively clear, it is not entirely trivial to define this notion in a formal manner, see [Meyer, 1988], [Dignum and Meyer, 1990] for this).

Other dynamic logic approaches were taken by [Khosla and Maibaum, 1987], [Khosla, 1988] and [Fiadeiro and Maibaum, 1991]. Meyer's approach is similar to [Khosla and Maibaum, 1987] who define an extension of modal action logic called Deontic Action Logic (DAL). The similarity is that every state of the world is labelled as either forbidden or permitted. One difference is that in Meyer's logic, every action that leads to a forbidden state of the world is forbidden, and every action that leads to a permitted state of the world is permitted. Another difference is that the reason for violation can be represented in the violation predicate, which provides information to specify the appropriate corrective action and to give informative error messages. Furthermore, action negation is heavily used to define the relation between the three modal operators. Other differences concern the use of propositional negation (to enforce deterministic processes) and the kind of semantic structure for specification ([Meyer and Wieringa, 1993]).

Recently some new approach to combine dynamic logic and action logic have been undertaken (see e.g. [Herrestad and Krogh, 1995], [Santos and Carmo, 1996] and [Tan and Thoen, 1996]) but these will not be discussed here.

6.2. LOGICAL FOUNDATION FOR MODELLING COMMUNICATION

In order to model the communication between agents in a distributed system the illocutionary language L_{ill} is developed. The language is presented bottom-up. First a static language L_{stat} is given, that describes the predicates that are part of all message and action specifications, after which the language of actions L_{act} (a variant of dynamic logic ([Harel, 1979]) and transactions language L_{tract} is described. Then the dynamic deontic language L_{dd} is introduced. In section 3 this language is extended with speech acts, and temporal operators in order to model deadlines, to form the illocutionary logic L_{ill}.

An important note is that when we speak about an "action" we do not mean the elementary action as used in dynamic logic. Instead a more generalised action, or transaction as it is called here, is meant. Typically, where we speak about actions we also mean transactions, e.g., in the case of a directive speech act DIR(α) (which itself is an action), α can be a transaction. The action logic therefore is a logic of transactions. One reason why the elementary action concept does not suffice lies in the specification of deadlines, which means the incorporation of temporal aspects in the logic. In [Dignum et al., 1996a] we have done this by extending the dynamic logic with transactions. The incorporation of time in dynamic logic is a complex matter that we leave for future work. A first attempt at specifying temporal aspects of actions is given in [Dignum and Kuiper, 1996]. In my view changing the names as used in traditional dynamic and deontic logic will lead to more confusion than just taking into account that whenever we speak about actions, the generalised form is meant.

This standpoint, however, has some consequences for the specification of deontic modalities. As described in section 1.1.2.3 the fundamental reason for the use of deontic concepts in communication modelling is that coordination of behaviour always requires some form of mutual commitment. Committing to something is specified as having the obligation that that something is the case in the present state. However the standard Anderson reduction cannot be used to specify the meaning of this, because of the many paradoxes it contains. In Meyer's reduction to dynamic logic the deontic operators reach over actions only. Because the action should be executed in the future, it cannot be guaranteed, so the interpretation "it will happen in *all* future courses of events" is too strong, but the interpretation "it will happen in *some* course of events" is too weak. Therefore the formula "α is obligatory" is interpreted as: "not doing α leads to violation". However now not only actions should be taken into account but also transactions, and it should be possible to specify that something should have been done or happened before a certain deadline. By expressing that a violation is created if something is obligated and not the case in the present state we get a more precise meaning of what it is that something is on an agent's agenda. Violations do not cause logical inconsistency, but can be the trigger for sanctions or repair actions.

6.2.1. THE STATIC LANGUAGE L_{stat}

To describe the semantics of communication processes, first the meaning of the messages is specified.

Definition 6.1. (message)

A *message* has the following structure. First, it consists of an *illocution* and a *proposition*. A proposition, in turn, consists of an *action* involving one or more *objects*. Illocution type, action type and object type draw on a given set of *predicates*.

L_{stat} ([Wieringa et al., 1991]) is a simple first-order language with variables, constants, function symbols and predicate symbols. Two special predicates are the unary predicate E (existence) and the binary predicate = (equality). Terms and formulas are built in the usual way using $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\forall$, $\exists$, and punctuation symbols '(', ')', '[' and ']'. We use infix notation for '='. The existence predicate E is used by convention to single out the set of existing objects among the set of possible objects. The following abbreviations are used:

$\forall^E x[\phi(x)] == \forall x\ [E(x) \Rightarrow \phi(x)]$ and
$\exists^E x[\phi(x)] == \exists x\ [E(x) \wedge \phi(x)]$.

We presuppose the usual model concept from first-order predicate logic.

The language can be given a Herbrand-Kripke possible world semantics which is omitted here. The interested reader is referred to [Weigand et al., 1995].

If knowledge is expressed as closed statements about objects of a certain type, then we must be able to talk about types. We follow Sowa ([Sowa, 1984]; see also [Guarino, 1992]) in using an explicit *type* predicate to declare the type of a term.

Definition 6.2. (Typed Logic)

Let T be a finite set of constants not occurring in L_{stat}. The elements of T are called *type names* and τ is used as meta-variable over T. L_{stat} is extended to the typed language TL_{stat} as follows:

1. TL_{stat} contains a special binary predicate *type* and the set T of type names. The only well-formed atomic formulas that can be built with *type* are of the form *type(t, τ)* for a term t and a type name τ, and the only place where τ can occur is as the second argument of *type*. *type(x, τ)* is called a *declaration* of x.
2. We introduce the abbreviations

   $\forall x : \tau\ (\phi(x)) == \forall x(\text{type}(x, \tau) \rightarrow \phi(x))$ and

   $\exists x : \tau\ (\phi(x)) == \exists x(\text{type}(x, \tau) \wedge \phi(x))$.
3. The language TL_{stat} is the set of all *closed* statements that can be built this way and which have all their variables typed. The inference relation $\vdash$ is defined as usual for first-order logic. We only consider formulas in prenex normal form, i.e. $Q_1x_1 \ldots Q_nx_n(\phi(x_1, \ldots, x_n))$, where $x_1, \ldots, x_n$ are all the free variables in ϕ and Q_i are quantifiers. Because all variables are typed, we can write this as $Q_1x_1 : \tau_1 \ldots Q_nx_n : \tau_n\ (\phi'(x_1, \ldots, x_n))$ with $\tau_i \in T$.

The type and role semantics are omitted here, see [Wieringa et al., 1991] and [Weigand et al., 1995] for this. In giving a semantics to the type names, there is a choice of keeping the extension of a type name constant in each world, or varying it. This choice has an intuitive meaning, e.g. compare the types *Person* and *Employee*. Some objects can become employees or cease to be employees without coming into existence or ceasing to be. There is life before being hired by a company, as well as after terminating a contract. On the other hand, there is no kind of object that can become a person without coming into existence, or that can cease to be a person without ceasing to exist. Apparently, being a person is an essential property of objects in the way that being an employee is not. [Wieringa et al., 1991] calls types like *Person natural kinds* and types like *Employee roles*.

6.2.2. THE DYNAMIC LANGUAGE L_{act}

Definition 6.3. (Action Logic)

The language L_{act} of parameterized actions ([Wieringa et al., 1989]) is given by the following BNF:

$\alpha :: = \underline{a} \mid \alpha_1 + \alpha_2 \mid \alpha_1\ \&\ \alpha_2 \mid \overline{\alpha} \mid$ **any** $\mid$ **fail**

$\underline{a}$ stands for the atomic action expression, like "order(i,j,p)", which states that agent i orders p from agent j. The first parameter indicates the subject of the action. The meaning

of $\alpha_1 + \alpha_2$ is a choice between α_1 and α_2. $\alpha_1 \,\&\, \alpha_2$ stands for the parallel execution of α_1 and α_2. The expression $\overline{\alpha}$ stands for the non-performance of the action α. The **any** action is a universal or "don't care which" action. The **fail** action is the action that always fails (deadlock). This action does not lead to a next state.

The semantics of actions can be given in a possible world semantics, cf. [Meyer, 1988].

6.2.3. THE TRANSACTION LANGUAGE L_{tract}

In this section the transaction logic language together with the semantics of (trans)action expressions is given.

Definition 6.4. (Transaction Logic)

The language L_{tract} of transaction expressions is given by the following BNF:

$\beta :: = \underline{a} \mid \beta_1 + \beta_2 \mid \beta_1 \,\&\, \beta_2 \mid \overline{\beta} \mid \textbf{any} \mid \textbf{fail} \mid \beta_1;\beta_2$

Note that the definition of L_{act} is almost the same as for L_{tract} except that we do not allow for sequence of actions.

6.2.3.1. TRANSACTION EXPRESSION SEMANTICS

The semantics of (trans)action expressions is given in two stages. First an algebra of uninterpreted actions (called a *uniform* semantics [de Bakker et al., 1986]) is defined, which allows for the interpretation of equalities between action expressions without taking their effect into account. Next, a state-transition semantics of action expressions is given in which the effect of steps on the state of the world are defined.

6.2.3.1.1. Algebraic action semantics

With every atomic action expression $\underline{a} \in Act$ we associate an event a in a given class $\boldsymbol{A}$ of events, with typical elements $a,b,c....$ Different atomic action expressions are associated with different events in $\boldsymbol{A}$. Events are the semantical entities on which we shall base our interpretation of action expressions. We further assume a special event δ, which is not an element of $\boldsymbol{A}$, called failure (comparable to deadlock in process algebra [Baeten and Weijland, 1990]). The relation between an action expression $\underline{a} \in Act$ and the associated event $a \in \boldsymbol{A}$ is more involved than just interpreting $\underline{a}$ as a. We shall interpret atomic action expressions $\underline{a} \in Act$ in a more sophisticated way, which we call "open": the meaning of an atomic action expression $\underline{a} \in Act$ will be the event $a \in \boldsymbol{A}$ corresponding with it, in combination with any other subset of the events in $\boldsymbol{A}$. Thus $\underline{a}$ expresses that a occurs, but it leaves open which other events occur simultaneously (in the same step) with a. The intuitive motivation for this is that if we say that an event a occurs, we do not mean that nothing else occurs in the world.

Definition 6.5. (Steps)

1. The set $\{\delta\}$ is a step.
2. Every non-empty finite subset of $\mathcal{A}$ is a step. The powerset of non-empty finite subsets of $\mathcal{A}$ will be denoted by $\wp^+(\mathcal{A})$.

Notation: In concrete cases we write the sets indicating steps with square brackets, in order to distinguish them easily from other sets that we will use. So, the step consisting of δ is written as $[\delta]$ and the step consisting of the events a and b is written as

$$\begin{bmatrix} a \\ b \end{bmatrix}.$$

The above definition prevents the simultaneous execution of the special event δ with other events, because it is not in $\mathcal{A}$. This is necessary, because it is not possible to perform an event and at the same time have a deadlock.

To denote the subsequent execution of actions we make use of sequences of steps. These sequences can be finite or infinite. We will call these sequences of steps *traces* conform the terminology used in the semantics of concurrent programming [Broy, 1986]. The definition of a trace is given as follows:

Definition 6.6. (Traces)

A trace is a finite or infinite sequence $S_1S_2...S_n$ ... of steps.
ε stands for the empty trace.
Only the last step of a trace may be $[\delta]$.
The number of steps in a trace t is called the *length* or *duration* of t, denoted by *dur(t)*.
$dur(\varepsilon) = 0$

Notation: We use $t,t_1,t_2,...,t',...$ to denote traces. We use A* to denote the set of all traces that can be formed from $\mathcal{A}$.

Definition 6.7. (Domain)

The domain $\mathcal{D}$ for our model of transaction expressions from *Tract* is the collection of sets of traces. An element of $\mathcal{D}$ is called a choice set and is denoted with $T,T_1,...$.

The use of choice sets as elements of the domain indicates the inherent non-determinism of the performance of the actions. Only when the semantics of a transaction expression consists of a choice set with one element will the transaction be deterministic (see also [Dignum et al., 1994]). Just as for traces we can define the length of a choice set (which will indicate the length of the transaction expression):

Definition 6.8. (Duration of choice set)

The *length* or *duration* of a choice set T, denoted by $dur(T)$ is defined as:
$dur(T) = max\{dur(t) \mid t \in T\}$

Below, we interpret transaction expressions in terms of choice sets. In order to do this we define the semantical counterparts of the syntactic operators in *Tract* (+, &, ¯ and ;). Before we give the definitions of these operators, some helpful functions are defined.

We start with the definition of prefixes of traces.

Definition 6.9. (Prefixes of traces)

$pref(t) = \{t' \mid t' \circ t'' = t\}$

Note that ε is an element of the *pref* of any trace. The '∘' operator denotes concatenation of traces and is defined formally in definition 6.12 below.

The next function defines the longest common prefix of two traces.

Definition 6.10. (Longest common prefix)

Let $t_1 = S_1...S_n$ ... and $t_2 = S_1'...S_m'...$
Then $maxpref(t_1,t_2)$ is the longest trace t such that $t \in pref(t_1)$ and $t \in pref(t_2)$.

Note that if $S_1 \neq S_1'$., $maxpref(t_1,t_2) = \varepsilon$.

Finally we define an operator (T^{δ}) on choice sets, which removes traces ending in [δ]. These traces are only removed if there is another trace that is the same but with [δ] replaced by another trace.

Definition 6.11. (Trace ending removing)

Let T be a choice set then
$T^{\delta} = T \setminus \{t \mid t = t'[\delta] \wedge \exists t'' \in T : t'' \neq t \wedge t' \in pref(t'')\}$

The operator T^{δ} is closely related to what is called "failure removal" in [de Bakker et al., 1986]. The idea is that failure is avoided when possible, i.e. when there is a non-failing alternative. In [Broy, 1986], this is called *angelic* nondeterminism.

We will now define the semantical counterparts of each of the syntactic operators. We start with the simplest, the ";". The semantical counterpart of this operator is the concatenation of choice sets (representing the semantics of the transactions that are performed in sequence).

Definition 6.12. (Concatenation operator)

1. Let $t = S_1...S_n$ and $t' = S_1 \Rightarrow ...S_m \Rightarrow$ be two traces (possibly infinite) then

$$t \circ t' = \begin{cases} S_1...S_n & \text{if } S_n = [\delta] \\ S_1...S_n S_1'...S_m' & \text{if } S_n \neq [\delta] \end{cases}$$

   If t is an infinite trace, then $t \circ t' = t$ for any trace t'.
   $t \circ \varepsilon = \varepsilon \circ t = t$.
2. Let T and T' be choice sets, then $T \circ T'$ is defined as the choice set $\{t \circ t' \mid t \in T, t' \in T'\}$.

Note: $T \circ \{\varepsilon\} = \{\varepsilon\} \circ T = T$ and $\{[\delta]\} \circ T = \{[\delta]\}$ and $T \circ \{[\delta]\} = \{t \circ [\delta] \mid t \in T\}$.

For the parallel operator "&" we use a set-intersection $\odot$, which is almost the same as the normal set-intersection, except that a trace can appear in the intersection not only if it appears in both sets, but also if it appears in one set and the other set contains a prefix of it. The definition assures that if two transactions are compatible, then the length of the transaction that results from performing them simultaneously is equal to the length of the longest transaction.

Definition 6.13. (Parallel operator)

1. Let t and t' be traces:

$$t \odot t' = \begin{cases} t & \text{if } maxpref(t,t') = t' \\ t' & \text{if } maxpref(t,t') = t \\ maxpref(t,t') \circ [\delta] & \text{otherwise} \end{cases}$$

2. Let $T,T' \in \mathfrak{D}$:

$$T \odot T' = (\cup\{t \odot t' \mid t \in T,\ t' \in T'\})^{\delta}$$

The semantical counterpart of the choice operator is defined as follows:

Definition 6.14. (Semantic counterpart of choice operator)

For $T,T' \in \mathfrak{D}$:

$$T \oplus T' = ((T \cup T') \setminus (\cup\{t \odot t' \mid t \in T,\ t' \in T' \wedge t \neq t'\})^{\delta}$$

The above definition states that the choice between two choice sets is the union of those two choice sets minus some type of intersection. However, it is not the actual intersection of the two choice sets that is subtracted, but those traces that only appear in one set and have a prefix in the other set.

This complicated definition is needed to secure equalities like: $\alpha_1 + (\alpha_1;\alpha_2) = \alpha_1$

Finally, we define the semantic counterpart of the negation operator.

Definition 6.15. (Negation operator)

The definition of "~" is given as follows:

1. For a step S,

$$S^{\sim} = \wp^{+}(\mathcal{A}) \setminus \{S\}$$

2. For a non-empty trace $t = S_1 \ldots S_m \ldots$

$$t^{\sim} = \bigcup_{n \leq dur(t)} S_1 \circ \ldots \circ S_n^{\sim}$$

3. For a non-empty set $T \in \mathfrak{D}$

$$T^{\sim} = \odot_{S \in T} S^{\sim}$$

That is, for a step the negation just yields the set-theoretic complement of $\{S\}$ with respect to $\wp^{+}(\mathcal{A})$. The negation of a trace consists of all the traces that start with a prefix and end with the negation of the step following that prefix. The negation of a choice set T is the "special" intersection of the sets(!) of the negations of all the traces contained in T.

We can now give the algebraic semantics of action expressions:

Definition 6.16. (Algebraic semantics)

The semantic function $[\ ] \in Act \rightarrow \mathcal{D}$ is given by:

$[\underline{a}] = \{S \in \wp^+(\mathcal{A}) \mid a \in S\}$

$[\alpha_1 + \alpha_2] = [\alpha_1] \circledcirc [\alpha_2]$

$[\alpha_1 \,\&\, \alpha_2] = [\alpha_1] \circledcirc [\alpha_2]$

$[\alpha_1 ; \alpha_2] = [\alpha_1] \circ [\alpha_2]$

$[\overline{\alpha}] = [\alpha]^{\sim}$

[**fail**] = $\{[\delta]\}$

[**any**] = $\wp^+(\mathcal{A})$

The first clause of the above definition expresses that the meaning of the action expression $\underline{a}$ is exactly as we have described informally before: it is the set of steps that contain the event a, representing a choice between all (simultaneous) performances of sets of events which at least contain the event a, so that the performance of a is guaranteed but also other events may happen simultaneously. The meaning of the action expression **fail** is comparable to a deadlock. The only event that can be performed is δ. The action expression any is the complement of **fail**. It stands for a choice of any possible combination of events.

Finally we define duration, equality and implication between action expressions.

Definition 6.17. (Duration, equality, implication of action expressions)

The **duration** of α is defined as $dur(\alpha) = dur([\alpha])$.

Action expressions α_1 and α_2 are equal, written $\alpha_1 =_{\mathcal{D}} \alpha_2$, iff $[\alpha_1] = [\alpha_2]$.

α_1 *involves* or *implies* α_2, written $\alpha_1 > \alpha_2$ iff $[\alpha_1] \subseteq [\alpha_2]$.

6.2.3.1.2. State-transition action semantics

To get a state-transition semantics, we postulate what effects events have in terms of state transformations (we do this relative to a set Σ of states).

We assume that there is a function $eff : \mathcal{A} \rightarrow (\Sigma \rightarrow \Sigma)$, such that $eff(a)$ is a function from states to states. (For simplicity, we assume events to be deterministic. Elsewhere, it is shown how nondeterministic events can be incorporated [Meyer, 1988]) Two events are called *compatible* if their joint effect is independent of the order in which they occur.

Definition 6.18. (Accessibility relation)

Let $S = [a_1, \ldots, a_n] \subseteq \mathcal{A}$ be a step consisting of pairwise compatible events. The accessibility relation $R_S \subseteq \Sigma \times \Sigma$ is defined as follows:

$R_S(\sigma, \sigma') \Leftrightarrow_{def} eff(a_1) \circ \ldots eff(a_n)(\sigma) = \sigma'$

On the basis of the accessibility relation R_S we also define an accessibility relation based on traces.

Definition 6.19. (Accessibility relation on traces)

Let t and t' be traces then:
$R_{t \circ t'}(\sigma, \sigma') \Leftrightarrow_{def} \exists \sigma''\ R_t(\sigma, \sigma'') \wedge R_{t'}(\sigma'', \sigma')$

6.2.4. THE DYNAMIC DEONTIC LANGUAGE L_{dd}

Now that the basic languages of actions and transactions are defined we can introduce the dynamic deontic language. This includes the dynamic operator [ ], and the deontic operator O for obligation. The intuitive meaning of $[\alpha]\Phi$ is that after the execution of α, Φ necessarily holds. The formulas defined by $O(\phi)$ define the "classical" deontic formulas as introduced by von Wright ([von Wright, 1951], [Åqvist, 1984]). The informal meaning of $O(\phi)$ is that ϕ should be the case in the present state. If both the obligation for ϕ and $\neg\phi$ hold in the same state we call this a violation, expressed by the special predicate *Violation*. The other deontic operators can be introduced using the usual abbreviations:

$F(\phi) \equiv O(\neg\phi)$
$P(\phi) \equiv \neg F(\phi)$

The language L_{dd} is defined with the following assumptions:

1. We introduce a special class Ag of agents. The agents in the distributed system can be humans (as in Workflow Management) or database applications (as in EDI) or the CIAs defined before that have been delegated certain tasks;
2. actions are parameterized; the first parameter represents the agent of the action. The formal introduction of parameterized actions can be found in [Dignum and Meyer, 1990];
3. the operators deontic operators O, F and P are indexed with the Superordinator and Subordinator agents. Thus $O_{ij}(\phi)$ should be read as: "agent i is obliged to agent j that ϕ holds". The violation corresponding to this obligation can be indexed accordingly.

Definition 6.20. (Dynamic Deontic Logic)

The language L_{dd} of dynamic deontic logic is given by the following BNF:

$\Phi :: = \phi \mid \Phi \vee \Psi \mid \Phi \wedge \Psi \mid \neg\Phi \mid [\alpha]\Phi \mid O(\phi) \mid B(i,\phi) \mid I(i,\phi) \mid I(i,\alpha)$

Where ϕ is a first order logic formula from L_{stat}, and α an element of L_{tract}, i is an element of Ag. The language L_{stat} is supposed to contain a special predicate *Violation*.

The formal semantics of $\vee$, $\wedge$, $\neg$ are standard and omitted here. The formal semantics of the other formulas are based on the semantics of action expressions given in the previous section and special accessibility relations:

Definition 6.21. (Dynamic Deontic Logic semantics)

$\sigma \models [\alpha]\Phi : \forall t \in [\alpha]\ \forall \sigma' \in \Sigma\ R_t(\sigma, \sigma') \Rightarrow \sigma' \models \Phi$
$\sigma \models O(\phi) : \forall \sigma' \in \Sigma\ R_O(\sigma, \sigma') \Rightarrow \sigma' \models \phi$
$\sigma \models B(i,\phi) : \forall \sigma' \in \Sigma\ R_B{}^i(\sigma, \sigma') \Rightarrow \sigma' \models \phi$
$\sigma \models I(i,\phi) : \forall \sigma' \in \Sigma\ R_I{}^i(\sigma, \sigma') \Rightarrow \sigma' \models \phi$
$\sigma \models I(i,\alpha) : \forall \sigma' \in \Sigma\ R_I{}^i(\sigma, \sigma') \Rightarrow \sigma' \models \alpha$

Instead of giving properties of the accessibility relations R_t, R_O, $R_B{}^i$, $R_I{}^i$ some useful axioms are given. In section 6.3.3 a more complete semantics is given.

We use the following axioms concerning the use of the action construction:

Axiom 1. (Action 1)

1. $[\alpha](\phi_1 \rightarrow \phi_2) \rightarrow ([\alpha]\phi_1 \rightarrow [\alpha]\phi_2)$
2. $[\alpha_1 + \alpha_2]\phi \leftrightarrow [\alpha_1]\phi \wedge [\alpha_2]\phi$
3. $\left[\overline{\alpha_1 + \alpha_2}\right]\phi \leftrightarrow \left[\overline{\alpha_1}\ \&\ \overline{\alpha_2}\right]\phi$
4. $\left[\overline{\alpha_1\ \&\ \alpha_2}\right]\phi \leftrightarrow \left[\overline{\alpha_1} + \overline{\alpha_2}\right]\phi$
5. $[\mathbf{fail}]\phi \leftrightarrow \mathbf{true}$

The meaning of $\alpha_1 \rightarrow \alpha_2$ is that the performance of α_1 involves the performance of α_2. For instance, "drinking coffee" involves "drinking" and "washing and singing" involves "singing". It is a kind of implication between actions. Of course this relation is reflexive and transitive. We have the following axioms:

Axiom 2. (Action involvement)

1. $\alpha \rightarrow \alpha$
2. $(\alpha_1 \rightarrow \alpha_2) \wedge (\alpha_2 \rightarrow \alpha_3) \rightarrow (\alpha_1 \rightarrow \alpha_3)$
3. $(\alpha_1 \rightarrow \alpha_2) \rightarrow ([\alpha_1]\phi \rightarrow [\alpha_2]\phi)$

We also use the following abbreviation:

Definition 6.22. (Equality)

$\alpha_1 = \alpha_2$ iff $\alpha_1 \rightarrow \alpha_2 \wedge \alpha_2 \rightarrow \alpha_1$

For the deontic operators the system KD (section 6.1.1) holds:

Axiom 3. (Obligations)

1. tautologies of Propositional Calculus
2. Modus Ponens
3. O-necessitation
4. K: $O(\varphi \rightarrow \psi) \rightarrow (O(\varphi) \rightarrow O(\psi))$
5. D: $\neg O(\varphi \wedge \neg\varphi)$

From above axiom system the theorems

$O(\varphi \wedge \psi) \rightarrow (O(\varphi) \wedge O(\psi))$ and
$O(\varphi \vee \psi) \rightarrow (O(\varphi) \vee O(\psi))$ can be derived.

Besides the obligation over propositions also obligations over actions $O(\alpha)$ can be specificied. The intuitive meaning of the deontic operations over actions is as follows: the action α is obliged if not doing α leads to a violation state, α is forbidden if doing α leads to a violation state, and α is permitted if it is not forbidden to do α.

To show we can still talk about violation when a certain action is not performed we jump ahead to section 6.3.3 and borrow the temporal operator PREV (the intuitive meaning of PREV(α) is "the present state is actually reached by performing α") and apply proposition 6.1. $O(\alpha)$ can then be defined as:

$O(\alpha) \equiv [\textbf{any}]O(PREV(\alpha))$

or, after whatever action you do, the obligation that the previous action performed was α should hold. We can now also look at what it means to *not* do α:

- by definition we have: $[\overline{\alpha}]PREV(\overline{\alpha}) \leftrightarrow [\overline{\alpha}]\neg PREV(\alpha)$
- from the obligation we have: $O(\alpha) \equiv [\textbf{any}]O(PREV(\alpha) \rightarrow [\overline{\alpha}]O(PREV(\alpha))$
- which means that we have: $[\overline{\alpha}]\neg PREV(\alpha) \wedge O(PREV(\alpha))$

or, after the non-performance of α the obligation that the previous action is α *and* the previous action was not α will hold, which is a violation.

The Belief and Intention operators do not play a very important role, as for instance in many other agent theories, but they are used in section 6.3.2.2 for the representation of the intended effects and sincerity conditions of speech acts.

The meaning of $B(i,\phi)$ is that agent i believes ϕ. We use the "standard" axioms for believes:

Axiom 4. (Belief)

1. $B(i,(\phi_1 \rightarrow \phi_2)) \rightarrow (B(i,\phi_1) \rightarrow B(i,\phi_2))$
2. $\neg(B(i,\phi) \wedge B(i,\neg\phi))$
3. $B(i,\phi) \rightarrow B(i,B(i,\phi))$
4. $\neg B(i,\phi) \rightarrow B(i,\neg B(i,\phi))$

$I(i,\alpha)$ means that agent i intends to perform α and $I(i,\phi)$ means that agent i intends to bring ϕ about. These are very weak notions for which only the following axioms hold:

Axiom 5. (Intention)

1. $(\alpha_1 \rightarrow \alpha_2) \rightarrow (I(i,\alpha_1) \rightarrow I(i,\alpha_2))$
1'. $(\phi_1 \rightarrow \phi_2) \rightarrow (I(i,\phi_1) \rightarrow I(i,\phi_2))$
2. $\neg(I(i,\alpha) \wedge I(i,\overline{\alpha}))$
2'. $\neg(I(i,\phi) \wedge I(i,\neg\phi))$

6.3. SPEECH ACTS AND DYNAMIC DEONTIC LOGIC

In this section the logical formalism that incorporates the speech acts into dynamic deontic logic is described. Before this can be done we revisit the illocutionary logic, as defined by [Searle and Vanderveken, 1985] and introduced in section 2.1.1.3, and look how this applies to communication between formal (computer) systems.

Again it is stressed that communications in multi-agent systems are considered to be speech acts and the aim is to provide a semantics for such communicative actions. We do not focus on the natural language aspects of the problem. In other words, we do not provide a theory of what a given natural language utterance may be interpreted as. The research described here on communication and traditional work in natural language processing are complementary in that the former is concerned with the content and structure that different communications must have, while the latter is concerned with the form they must take to accurately correspond to that content.

6.3.1. ILLOCUTIONARY LOGIC FOR FORMAL COMMUNICATION

It is important to make the distinction between informal (human) communication and formal (computer) communication, because many aspects that are important for informal communication do not occur in formal communication or have no impact on the effect of the communication. For instance, the concept of 'sincerity', which indicates whether the speaker actually means what he is saying or not, has no influence on the communication between formal systems. Here the elements of illocutionary logic that are necessary to describe formal communication are described.

The illocutionary act with its three components is taken as starting point. So, an act of communication between formal systems will also be described by:

- propositional contents
- illocutionary context
- illocutionary force

The *propositional contents* of the illocutionary act expresses what the speech act is about. It describes objects and actions of which the meaning is described in the lexicon.

The *illocutionary context* indicates the relevant knowledge about the situation in which the speech act is made. From section 2.1.1.3.1 we learned that this can be factual knowledge about the place where the speech act is performed, but also epistemic knowledge about the intentions and believes of the participants in the speech act. It also includes the speaker and addressee of the speech act themselves.

From the five elements of illocutionary context only the speaker and addressee are used. The speaker and addressee are part of the speech act itself and are incorporated in the formal representation of the speech act. These two elements seem to be closer related to the communication than the other elements of the context. The time and circumstances

are not incorporated in the illocutionary act itself but are indeed seen as the (independent) context in which the act is performed. Time is relevant, however the interest is not with the specification of the moment in time the speech act is performed, but with the specification of deadlines (see below). In formal network-based communication, location seems to be less relevant. A communication between a computer in Australia and a computer in Holland is not easily located at one point in space. Although the location might be of importance in communication between two persons that are at the same place, it is not of any importance in communication between formal systems.

The *illocutionary force* determines for a large part the reasons and goal of the communication. It contains seven elements. Besides the illocutionary point, the other elements of the illocutionary force are all dependent on the illocutionary point. They either indicate the strength of it or the effect of it in some way. From the seven components of illocutionary force only five will be used in formal communication:

- illocutionary point
- degree of strength of the illocutionary point
- mode of achievement
- propositional content conditions
- preparatory conditions

The central element of the illocutionary force is the *illocutionary point*. Of course, this is also used in formal communication. The illocutionary point indicates the type of effect for which the act is performed. Five different illocutionary points are distinguished. The basic illocutionary types that are supported in the framework are *assertive, directive, commissive* and *declarative* ([Austin, 1962], [Searle, 1969], [Searle and Vanderveken, 1985], [Lehtinen and Lyytinen, 1986]). The fifth illocutionary type Searle distinguished is the *expressive*, whose point is to express the speaker's psychological state, feelings and attitude towards the state of affairs. The expressive is left out of the framework since the attitude of one of the agents towards some state of affairs has no influence on the communication in formal systems. An example of a message with a illocutionary force is: the assertion that a certain flight has been reserved, or the request that a certain flight be reserved (from agent A to agent B). In these two cases, the proposition is the same, but because of the different illocutions, the meaning of the two messages is quite different. Note that an action occurrence referred to in an assertive message means that the action has taken place, or is taking place, depending on the time of action, and that an action occurrence referred to in a directive message means that the action must be executed.

The *degree of strength* of the illocutionary force in a formal context is not so much used as a degree of intention of the speaker like in the original analysis of Searle. In that context, for instance an 'order' expresses a stronger desire to have the propositional contents of the speech act be realised by the addressee than a 'request'. In a formal communication context the degree of strength has an influence on the effect of the speech act and thus on the possible responses to the speech act. It might be possible that it is not obligatory to answer an order if the speaker has no authority over the addressee, while a request is always answered.

The *mode of achievement* indicates that some conditions must hold for the illocutionary act to be performed in that way. E.g., a directive can be given through a command or an order. A command makes use of a position of authority of the speaker while an order does not. Below distinctions in normative grounding of illocutionary points are described, such as power, authority and charity. They can be viewed as different modes of achievement.

In many cases the illocutionary point forces some *conditions on the propositional content* of the speech act. E.g., the propositional content of a promise must be that the speaker will cause some condition to hold in the future. One cannot promise to have done something in the past or that someone else will do something. The propositional content conditions are not modelled in our illocutionary logic at this moment. They can be modelled through a refinement of the language of speech acts which renders only those speech acts syntactically correct that comply to the propositional content conditions. In an IS environment, the propositional content conditions are contained in the data model.

There are basically two types of *preparatory conditions*. First, those that are dependent of the illocutionary point. E.g., if the speaker promises something it is presupposed that the thing he promises is beneficial for the addressee and also that the speaker can in some way fulfil the promise. There are also preparatory conditions that depend on the propositional content of the speech act. E.g., if I order someone to open the window I presuppose that the window is closed. Both types of presuppositions are included in the preparatory conditions of the illocutionary force.

The two elements left out of the illocutionary force in a formal communication context are the sincerity conditions and the degree of strength of these sincerity conditions. In speech act analysis it is important to be able to determine whether the intentions that are conveyed in the propositional contents of the speech act coincide with the intentions of the speaker. That is, one would like to be able to indicate that a certain statement is a lie. However, in formal communication this is not important. The intentions of the speaker do not play a role in formal communication except in as far as they are expressed in the speech acts. E.g., if someone orders a product without having the intention of ordering the product, all the consequences of the speech act will be exactly the same as when he would have that intention. This is formalized by making the intention of the speaker a direct effect of the speech act.

In the framework one component to the illocutionary acts is added, the *intended effect*. The intended effect of an illocutionary act is only effectuated if all the conditions of the act are fulfilled. For instance, the intended effect of ordering a product is the obligation of the addressee to deliver the product. However, there may be all kinds of circumstances that prevent this obligation to arise. For instance, when someone can only order a product if one paid all previous deliveries (a preparatory condition) and the speaker did not comply to this rule. In this case there is still an effect of the speech act, e.g. the addressee now knows that the speaker wants to order a product again. But this effect is not equal to the intended effect. The definition of a successful illocutionary act is an illocutionary act for which the intended effects are also actual effects of the speech act.

6.3.2. L_{dd} EXTENDED WITH SPEECH ACTS

In order to model the communication between agents in a normative system the language L_{dd} has to be extended to incorporate speech acts, as described in illocutionary logic. However, first we take a closer (formal) look at the authorization relations between agents. After this the speech acts are formally introduced.

6.3.2.1. AUTHORIZATION RELATIONS

Coordination of behaviour is relatively easy when there exists a hierarchical ordering between the agents. When agent A is superior to agent B, then a request of A will always lead to an obligation for B. However, especially when these systems belong to different organizational units, or in CSCW applications, where the agents are humans standing in a peer relationship, the coordination requires more effort. Let us suppose that A does send a request to B. In general, this can lead to an obligation on the part of B for three fundamentally different reasons ([Dignum and Weigand, 1995a]):

1. Charity
2. Power
3. Authorization

Power, authorization and charity can be seen as different validity claims, as were introduced in section 3.1.2 and 5.2.2.

1. *Charity* means that B answers A's request without being forced to do so.

   We take it for granted that systems that include humans, or are closely intertwined with human affairs, can never be formalized completely. Such systems should leave open the possibility of open requests. The request itself does not create an obligation; an obligation arises only when B replies with a positive commitment.

   The charity relation does not have a special notation.

2. *Power* means that B answers A's request because of some dominance relationship between A and B external to the communication network.

   This is the case in the hierarchical system mentioned above. By "external" we mean that the dominance relationship is not rooted in the communication process, but by mutual consent. Regarding power, [Auramäki and Lyytinen, 1996] state: "It seems to be useful to know if we are commanding or promising. Use of power is a part of everyday action, and we do not want to exclude communication based on the use of power from the analysis of communication". A power relationship can be restricted to a certain domain or to specific roles of the agents. There exists a power relation between the agent i and the agent j with respect to action α, if i has the power to order j to perform the action α. For instance, the boss can order his secretary to reserve a flight for him. Note that he might not have the power to order his secretary to make coffee for him! We assume that the power relation is persistent, i.e. it is not changed or finished by the fulfilment or non-fulfilment of the command; and is only changed in special occasions,

like when a manager is appointed. The power relation defines a partial ordering on the class of agents for every action α. This ordering is reflexive (self-power) and transitive but not necessarily total.

The most important property of the power relation is that it provides a basis to create obligations from one agent to another. The power relation can also be defined with respect to a proposition. This means so much as that i has the power to convince j of the truth of ϕ. E.g., a student will (usually) consider the statements (assertions) of a teacher to be true. This power relation has no legal connotation, because it will not be connected to obligations but to believes of another agent.

Notation: if i has power over j with respect to α we write: $j <_{\alpha} i$.

If i has power over j with respect to the truth of ϕ we write: $j <_{\phi} i$.

The above properties are made formal in the following definition and axiom:

Definition 6.23. (Power relations)

We use $PC_1(\alpha)...PC_n(\alpha)$ to indicate the actions that can change the power relation between two agents with respect to α. $PC_1(\phi)...PC_n(\phi)$ indicate the actions that can change the power relation between two agents with respect to ϕ.

The following axioms hold for the power relation:

Axiom 6. (Power relation)

1. $i <_{\alpha} i$
1'. $i <_{\phi} i$
2. $i <_{\alpha} j \wedge j <_{\alpha} k \rightarrow i <_{\alpha} k$
2'. $i <_{\phi} j \wedge j <_{\phi} k \rightarrow i <_{\phi} k$
3. $i <_{\alpha} j \rightarrow \left[\overline{PC_1(\alpha) \cup ... \cup PC_n(\alpha)}\right] i <_{\alpha} j$
3'. $i <_{\phi} j \rightarrow \left[\overline{PC_1(\phi) \cup ... \cup PC_n(\phi)}\right] i <_{\phi} j$
4. $(\alpha_1 \rightarrow \alpha_2 \wedge i <_{\alpha 1} j) \rightarrow i <_{\alpha 2} j$
4'. $(\phi_1 \rightarrow \phi_2 \wedge i <_{\phi 1} j) \rightarrow i <_{\phi 2} j$

Note. Rule 2 and 2' are introduced to be able to model relationships between contracts. Although they apply to most real-life cases (e.g., the CEO-manager-employee relation) there are some cases where the relations do not hold. For instance, in the Dutch legal system the district attorney can order an agent to arrest someone, and the agent has the power to make the arrest, but the district attorney can not make the arrest himself.

3. The third relation between agents is the *authorization* relation. This relation can be established for a certain time with mutual agreement (under certain restrictions).

   So, when B has committed itself to a certain service, a request of A leads to an obligation when the conditions are met. E.g., I can agree that a company can order me to pay a certain amount of money after they delivered a product. This relation ends after I pay the money. A refusal would lead to a violation of the agreement, which makes this case different from both the first one and the second.

See for instance [Dietz and Widdershoven, 1992] for the distinction between power and authorization claims in CSCW tools such as the Coordinator ([Flores et al., 1988]).

Notation: The authorization relation is modelled using a special predicate.

If i is authorized to do α we write: auth(i,α).

The semantics of an authorized request to do α is that O(i,α) holds (as a postcondition) provided that auth(i,α) holds (as a precondition). Although it seems that the authorization does not establish a relation between two agents, it does so whenever the action α involves another agent (which is always the case for speech acts as we will see below). E.g., auth(i,order(i,j)) means that i is authorized to order something from j. See also the next subsection for a discussion on the use and dynamics of authorization.

6.3.2.2. SPEECH ACTS

A speech act is formalised as an *illocutionary point* (indicating the goal of the speech act) with three parameters: the Speaker, the Addressee, and the content.

The following basic speech acts are distinguished:

Definition 6.24. (Basic speech acts)

DIR(i,j,α)	i does a request to j for α
COM(i,j,α)	i commits himself to j to do α
ASS(i,j,ϕ)	i asserts to j proposition ϕ
DECL(i,j,ϕ)	i declares and informs j that ϕ holds from now on

From these basic speech acts we can construct other basic speech acts by e.g. using the logical negation of actions.

Definition 6.25. (Speech acts)

FOR(i,j,α) = DIR(i,j, $\overline{\alpha}$)	i forbids j to do α
PER(i,j,α) = DECL(i,j,$P_{ji}(\alpha(j))$	i permits j to do α

The basic speech acts correspond to those given in section 2.1.1.3.1. There might be some dispute over the question whether the declarative DECL has an Addressee parameter, since if it succeeds, the effect will be a change of the world and not of the knowledge of the Addressee only. Depending on the preparatory conditions, it is not necessary that there is an Addressee at all. However, in general it makes little sense to perform a declarative speech act and not inform anybody. Hence, the Addressee should be understood here as the agent (or set of agents) that is informed.

As explained in section 2.1.1.3.1, declaratives can only be used for specific institutionalized speech acts, so the propositional content is usually rather restricted. In practice, a limited number of specific declaratives will be distinguished, such as the "authorization" action described above.

The language of all acts is now defined in two steps. First the set of all speech acts L_{Sact} is defined.

Definition 6.26. (L_{Sact})

1. All basic speech acts are elements of L_{Sact}.
2. If $\alpha \in L_{Sact}$ then also IP(i,j,α) $\in L_{Sact}$ and IP(i,j, $\overline{\alpha}$) $\in L_{Sact}$ where IP $\in$ {DIR, COM}

Note that this is a recursive definition. So, we can have speech acts about speech acts, etc. The following axioms hold for speech acts (for the inference relation on actions used here, see [Wieringa et al., 1991]):

Axiom 7. (L_{Sact})

1. for IP $\in$ {DIR, COM}:
   IP(i,j,α_1) & IP(i,j,α_2) = IP(i,j,α_1 & α_2)
   IP(i,j,α_1) $\cup$ IP(i,j,α_2) $\rightarrow$IP(i,j,$\alpha_1 \cup \alpha_2$)
2. for IP $\in$ {DECL, ASS}:
   IP(i,j,ϕ_1) & IP(i,j,ϕ_2) = IP(i,j,$\phi_1 \wedge \phi_2$)
   IP(i,j,ϕ_1) $\cup$ IP(i,j,ϕ_2) $\rightarrow$IP(i,j,$\phi_1 \vee \phi_2$)

The language L_{act} of actions is extended to include speech acts and can now be defined as:

Definition 6.27. (L_{ACT})

$L_{ACT} = L_{act} \cup L_{Sact}$

Again it should be stressed that although the speech acts are actions themselves the content of the speech act might be a (logical) transaction ($\alpha \in L_{tract}$).

The *propositional content conditions* are not modelled at this moment. They can be modelled through a refinement of the language L_{Sact} which renders only those speech acts syntactically correct that comply to the propositional content conditions. In an Information System environment, the propositional content conditions are contained in the data model.

The *preparatory conditions* (ϕ) and the intended effects (ψ) of a speech act (α) can be modelled through the following schema:

$\phi \rightarrow [\alpha]\psi$

which means that if ϕ is true then ψ will hold after α has been performed.

When deontic constraints are specified, this means the constraint *can* be violated. In such a case, we usually also want to express what the consequences are in terms of sanctions or remedial actions. These can be specified in our logic taking the Violation predicate as precondition.

The intended effects of the speech acts are described by means of deontic and epistemic operators, while the preparatory conditions refer to either the authorization relation or the power relation. For instance, a directive (DIR) can be made on the basis of a power relation (a command) or authorization (in which cases it is an order). For each basic speech act three variants are distinguished, indicated by a subscript c, p or a, for charity, power and authority respectively. So, DIR_a stands for an authorized request, whereas DIR_p stands for an order based on power. Similarly for assertives and declaratives. For commissives, the distinction seems to be not very relevant and is ignored here. Likewise, when the distinction of the powerbase of a speech act is not important, we will ignore the subscript. We have the following general preparatory conditions and intended effects for the basic speech acts. Of course, for speech acts mentioning specific actions there might be more conditions and effects.

Axiom 8. (Intended effects)

1. $([DIR_p(i,j,\alpha)]\ O_{ji}(\alpha)) \leftarrow j <_\alpha i$	an order based on power creates an obligation
2. $([DIR_a(i,j,\alpha)]\ O_{ji}(\alpha)) \leftarrow auth(i,DIR(i,j,\alpha))$	an authorized request creates an obligation
3. $[COM(i,j,\alpha)]\ O_{ij}(\alpha)$	a commitment (promise) creates an obligation
4. $[DECL_a(i,j,\phi)]\ \phi \leftrightarrow auth(i,DECL(i,j,\phi))$	an authorized declaration creates a fact
5. $[DECL_p(i,j,\phi)]\ \phi \leftrightarrow j <_\phi i$	a declaration based on power creates a fact
6. $([ASS_a(i,j,\phi)]\ B(j,\phi)) \leftrightarrow auth(i,ASS(i,j,\phi))$	an authorized assertion creates belief of the fact in the agent
7. $([ASS_p(i,j,\phi)]\ B(j,\phi)) \leftrightarrow j <_\phi i$	an assertion based on power creates belief of the fact in the agent

6 and 7 express the fact that an agent can be authorized to assert some facts. If this agent asserts such a fact the effect is that the Addressee(s) will believe that fact (which is not the same as making the fact true, as with a declaration!). That an assertion expresses a belief can also be described by saying that the effect of the assertion is that the hearer assumes that the speaker believes such and such. This effect is independent from the sincerity of the speaker. This is especially useful to create a set of common believes between several parties, which in the end may trigger some common action of the agents. For instance, if a bank and a company both believe that it is profitable to invest money in a new venture, this may result in the actual investment being financed by the bank.

3 describes formally the commitment of agents mentioned so often. As the result of i performing a $COM(i,j,\alpha)$ (commitment) action $O_{ij}(\alpha)$ becomes true (cf. [Dignum and Weigand, 1995a]), i.e. by committing itself to an action, an agent i obliges itself towards j to perform action α. The commitment is a private one if j is the same as i. Although the obligation does not ensure the actual performance of the action by the agent, it does have a practical consequence. If an agent commits itself to an action and afterwards does not perform the action a violation condition is registered, i.e. the state is not ideal (anymore).

The axioms describe the effects of power and authorization speech acts, but not of those based on charity. A request based on charity (DIR_c) does not create an obligation directly, but may urge the Addressee to commit himself. This is correct, although we might add some politeness rules that say that a message is always replied to. Such rules can be built on sincerity conditions. E.g., a request based on charity would be replied by either a commissive or an assertion of the effect that the agent does not commit himself:

$[DIR_c(i,j,\alpha)]\ O_{ji}(COM(j,i,\alpha) \cup ASS(j,i,\neg O_{ji}\alpha))$

Axiom 8 also illustrates how obligations can arise for an agent:

- by means of a command (DIR_p) and an existing power frame. In that case, the obligation arises independent of the commitment of the agent (1)
- by means of an authorized speech act (2)
- by means of a commitment of the agent (3)
- by means of a declaration the obligation exists and an existing power or authorization frame:
  $i <_\alpha j \rightarrow [DECL_p(j,i,O_{ij}(\alpha))]\ O_{ij}(\alpha)$
  $auth(i,DECL(j,i,O_{ij}(\alpha))) \rightarrow [DECL_a(j,i,O_{ij}(\alpha))]\ O_{ij}(\alpha)$
- by means of a request based on charity, followed by a commitment from the agent (no authorization is needed, the agent commits itself by free choice):
  $[DIR_c(j,i,\alpha)][COM(i,j,\alpha)]O_{ij}(\alpha)$

As argued in section 6.3.1 the sincerity conditions are left out of the illocutionary force in a formal communication context. It is assumed that an agent is always sincere, and its intentions are a direct effect of the speech act used. We can therefore describe the effects of performing speech acts on the mental states of the agents as follows:

Axiom 9. (Sincerity effects)

$[DIR(i,j,\alpha)]\ I(i,\alpha)$	any DIR speech acts expresses that i intends α to happen
$[DECL(i,j,\phi)]\ I(i,\phi)$	any DECL speech acts expresses that i intends to bring about ϕ (by the speech act)
$[ASS(i,j,\phi)]\ B(i,\phi)$	any ASS speech act expresses that i believes ϕ

So the effect of a DIR_c is at least that the receiving agent knows about the speaker's intention, and this can trigger him to commit himself, or a refusal message.

6.3.2.3. THE DYNAMICS OF AUTHORIZATION

This subsection discusses some aspects of authorization, the granting and retracting, and in general the dynamics of giving authorizations.

If a subject is not authorized, it cannot issue a DIR_a speech act successfully. In that case, it can try to attain an authorization first. This can be done by means of $DIR_c(i,j,DECL(j,i,auth(i,DIR(..))))$, i.e., a request for authorization of the other party. If the other party complies to the request and grants the authorization, the subject gets authorized from that time on. This example shows that a dynamic normative system

should not only formalize authorized behaviour itself, but also the creation of authorizations, and, for that matter, the deletion. The basic assumption underlying our formalization is that authorizations can only be made and retracted by an act of the other party. In this way, the autonomy of the agents is ensured. Because the establishment of authorizations is an important and frequently occurring speech act the following notation is introduced:

$$AUT(i,j,\alpha) == DECL_a(i,j,auth(j,\alpha))$$

So, $AUT(i,j,\alpha)$ means that i gives authorization to j to do α. Of course, this speech act is only successful if i is authorized to give this authorization. For that reason, we have to presuppose the following axiom:

Axiom 10. (Authorization)

1. $auth(i,AUT(i,j,DIR_a(j,i,\alpha(i))))$
2. $auth(i,AUT(i,j,ASS_a(j,i,p)))$

This says that each agent is authorized to authorize other parties as far as actions and beliefs of the agent himself are concerned. This is irrespective of whether the granting of authorizations is forbidden by for example a higher power. If that would be the case, the authorization would still be successful, although the agent might be punished for it.

Authorizations may refer to any action: material actions, communicative actions, and also to speech acts. An example of such an "indirect" authorization is the following:

$$auth(i,DIR_a(i,j,AUT(j,i,\alpha)))$$

This says that i is authorized to direct j to authorize him action α. So i might be not authorized yet, but he has the possibility to attain an authorization if he wants. From this example it is clear that quite precise agreements can be made. Such agreements may also concern the retracting of authorizations.

The ability to retract an authorization should be left to the subject of the authorization. If i has granted j an authorization, it is only j who can retract the authorization. For this purpose, we introduce a new declarative RTR:

$$RTR(i,j,\alpha) == DECL_a(i,j,\neg auth(i,\alpha))$$

The preparatory condition of RTR is that the authorization does exist. By axiom 10, every agent is authorized to retract authorizations given to him. If an agent has first granted an authorization, and then wants to retract it, he must ask the other party to do so. Of course, the agents may have made appointments. E.g., the agent who grants the authorization may ensure himself of the authorization to request the retracting. The effect is that he can have the authorization retracted whenever he wants.

$$auth(j,\alpha) \wedge auth(i,DIR_a(i,j,RTR(j,i,\alpha)))$$

An important question with respect to authorization is whether an authorization can be passed on. In the axiomatization above, this is possible but not dynamically. New

authorizations can be created by means of the AUT action only, and this action can only be performed by the object of the authorization (the one who becomes obliged). What is possible, dynamically, is that agent i gives agent j the authorization to request from him to give authorizations to some agent k. In this way, j can pass the authorization on, but only via i. That is, j issues the following directive to i:

$$DIR_a(j,i,AUT(i,k,DIR_a(k,i,\alpha)))$$

thus creating for i an obligation to do AUT(i,k,..).

Before that, j must be authorized by i in the following way:

$$AUT(i,j,DIR_a(j,i,AUT(i,k,DIR_a(k,i,\alpha))))$$

This speech act succeeds because every agent is, by axiom 10, authorized to grant authorizations concerning its own behaviour. If j had issued a DIR_c instead of a DIR_a, he would not have needed the latter authorization, but then it would depend on i's charity whether he would commit himself or not.

In the specification language it is possible to stipulate more general authorizations. E.g., j is authorized to grant authorizations about i's behaviour independently. However, the question is how such a specification becomes valid in the normative system. In terms of justification, this can only be done by means of power. An adequate treatment of the question will lead us to a formal definition of inheritance and delegation. Although no formal definition is given here I do want to make some remarks about it.

In normative systems, it is usual to organize authorizations and obligations in the form of roles. In an organization, roles can be secretary, manager, bookkeeper, etc. In the agent perspective, a role is nothing more than a set of capabilities and deontic rules. The rules specify the authorizations of the agent as well as the authorizations other agents have with respect to the agent. Furthermore, they specify obligations, permissions, and prohibitions. As in database authorization models (e.g., [Bertino and Weigand, 1994]) roles can be organized in an inheritance hierarchy. So, if secretary "isa" employee, and employees enjoy a certain authorization, then so does the secretary. For a discussion of problems related to the inheritance of permissions and obligations, see [Wieringa et al., 1991].

Inheritance should be distinguished from delegation. *Delegation* means that an agent assigns tasks to a subordinated agent, and this can only be done along the power dimension. From the axioms given above it follows that if $i <_\alpha j$, then j can issue directives, prohibitions and permissions, and these cause obligations etc. by virtue of the power relation. The power relation is independent of the inheritance relation between roles. What needs to be worked out is how delegation interacts with authorization. There are two cases to consider, one in which the 'lower' agent is the subject of the authorization and one in which he is the object of the authorization, owned by some third agent k.

When agent j has some authorization, he might want to delegate this to a subordinate i. E.g., j is the boss, who is authorized to give bank orders and he wants to delegate this to a clerk. In the normative systems proposed here, this can be done by asking the bank to authorize the clerk, as described above. So the boss is dependent on the cooperation of the bank. To retract the authorization, he can order the clerk to do so. We might release the

preparatory condition of retracting a bit by allowing a superordinate to retract authorizations from subordinates, but since he can achieve the same effect by a power directive, this complication of the RTR would be redundant.

When agent j wants to give an authorization to k to the effect that k can direct the subordinate i (for some action), he must order i to grant this authorization to k (use his power). He might reserve for himself the right to retract the authorization (more precisely, the right to request the retracting of the authorization).

The final point to consider with respect to delegation is the upward inheritance of obligations. Again, there are two cases: (i) there exists an obligation of agent k to agent i; is this an obligation to superordinate j as well ? (ii) there exists an obligation of agent i to agent k; is this an obligation of superordinate j as well ? A special case of the latter is when agent i has committed himself.

In contracting we assume the agent that handles the negotiation and commits itself to perform actions is authorized to do so (as these tasks have been delegated to it). As stated in section 5.2, we are only interested in the lowest level of contracts, but just the issues of the inheritance of obligations to superordinates or the organization itself is the focus of attention when grounding contracts. See also section 5.2.3.

6.3.2.4. DEONTIC LOGIC AXIOMS AND SPEECH ACTS

We will now take a look at some traditional deontic axioms and discuss how they are interpreted in the context of speech acts.

Axiom 11. $P_{ij}(\alpha) == \neg O_{ij}(\overline{\alpha})$

This axiom (sometimes also seen as an abbreviation) stipulates that everything which is not forbidden, is permitted. If permitted is interpreted as "authorized", this axiom is too strong, since authorizations only make sense for actions that affect the behaviour of others, such as DIR speech acts. In a network of autonomous agents, most of the agent's capabilities will be private actions, actions that affect the inner state of the agent only. For those actions, authorizations make no sense. In the weak sense of P as given in Dynamic Deontic Logic, the axiom is valid by definition..

Axiom 12. $O_{ij}(\alpha) \Rightarrow P_{ij}(\alpha)$

This axiom from the system of Von Wright is not valid in deontic dynamic logic. As argued in [Weigand, 1993] it *is* acceptable when it is given a speech act interpretation: a rational agent cannot request an action that is not permitted by himself. In the illocutionary logic, this constraint has different interpretations. One interpretation is that requesting something implies giving permission. Formally,

$DIR(\alpha) \supset PER(\alpha)$ or, equivalently, $[DIR(i.j,\alpha)]P_{ij}(\alpha) \leftarrow auth(j,i,\alpha)$

Note that in the illocutionary/deontic logic authorizations are needed for making authorized requests, but not for *all* actions, as mentioned above. For those actions, it is somewhat overdone to mark them as authorized.

Another way to formulate the above axiom in deontic logic is :

$\neg(O(\alpha) \wedge F(\alpha))$

which can be interpreted in the context of illocutionary acts as the fact that one cannot request an action and forbid it at the same time. Using the axioms for illocutionary acts we have that:

$DIR(\alpha)$ & $FOR(\alpha) = DIR(\alpha \,\&\, \overline{\alpha}) = DIR(\textbf{fail})$

If the speaker is authorized to let the hearer do both α and $\overline{\alpha}$ it holds that:

$[DIR(i,j,\textbf{fail})]O_{ji}(\textbf{fail}) = [DIR(i,j,\textbf{fail})][\textbf{any}]$ *Violation*

This means that the speech act can be successful, but that in the resulting state all actions lead to "Violation", i.e. are not deontically acceptable. In [Weigand, 1993], the speech act itself was not legitimate, i.e., DIR(i,j,**fail**) is made equivalent to **fail**. This is reasonable when prescribing rational communication, but too strong in a descriptive system. Requesting and afterwards forbidding makes sense if we take the second action as overruling the first one. In that case, the inconsistency would be solved by an appropriate formulation of the frame axioms, that is, the axioms that define the persistence of obligations (and other knowledge) from one state to another (see section 6.3.3.4).

6.3.2.5. EXAMPLES

Having given the formal definition of the logical language including speech acts we can come back to the statement that this would make it possible to model the business logic model from chapter 3. The communicative business logic can be made precise by formalising the communicative acts (messages) in the presented logic.

The following gives some examples of a generic business relation written in dynamic deontic logic:

$[DIR_C(i,j,\text{give-quotation}(j,i,g,p))]\ O_{ji}(\text{give-quotation}(j,i,g,p) \cup \text{refuse}(j))$

After a request for a quotation (i.e. a directive based pm charity) the company is obliged to give the quotation or send a refusal. This follows from the generic business rule that a request for a service offered is always answered.

$[\text{give-quotation}(j,i,g,p)]\ \text{auth}(i,DIR_a(i,j,\text{deliver}(j,i,g,p)))$

If a company gives a quotation for a certain price (p) the client is authorized to order the product (g) for that price. (i.e. a meaning definition for give-quotation).

$\text{auth}(i,DIR_a(i,j,\text{deliver}(j,i,g,p))) \rightarrow [DIR_a(i,j,\text{deliver}(j,i,g,p))](O_{ji}(\text{deliver}(j,i,g,p)) \wedge [\text{deliver}(j,i,g,p)]\ \text{auth}(j,DIR_a(j,i,\text{pay}(i,j,p))))$

If a customer is authorized to order a product for a certain price (i.e. a quotation has been given for that price) then the company is obliged to deliver the product after the customer has ordered it. (this follows directly from axioms 8) After delivery of the product, the company is authorized to order the customer to pay for it.

$$\text{auth}(j,\text{DIR}_a(j,i,\text{pay}(i,j,p))) \rightarrow [\text{DIR}_a(j,i,\text{pay}(i,j,p))]O_{ij}(\text{pay}(i,j,p))$$

If an order has been delivered (and authority acquired to request payment) a request for payment induces an obligation for the customer to pay. (This follows directly from the axiom 8)

$$O_{ij}(\text{pay}(i,j,p)) \rightarrow [\text{pay}(i,j,p)]\,\neg\text{auth}(j,\text{DIR}_a(j,i,\text{pay}(i,j,p)))$$

Finally, after the customer has paid, the company cannot request another payment again.

Note that the obligation to pay is made conditional: it becomes effective only when the supplier requests for it. This is of course only one way of doing it; the obligation can also be instantiated directly when the order is given, and even precede the delivery. These different ways of working can be distinguished in the logic described here.

The logic can also be used to reason about violations. This part is not explicitly included in the business logic framework, but is part of the business contract as well.

$$O_{ij}(i,j,\text{ship}(i,j,\text{goods})) \rightarrow [\neg\text{ship}(i,j,\text{goods})]\,O_{ij}(i,j,\text{pay}(100)) \wedge \text{auth}(i,j,\text{DIR}_a(i,j,\text{ship}(i,j,\text{other\_goods})))$$

if i is obliged to ship the goods and he does not do it, he is obliged to pay a fine and the other party j is authorized to request (other) goods

$$O_{ij}(i,j,\text{ship}(i,j,\text{goods})) \rightarrow [\text{cancel}(i,j,\text{ship}(i,j,\text{goods}))]\,\neg O_{ij}(i,j,\text{ship}(i,j,\text{goods}))$$
$$\text{date}() == t \rightarrow [\text{cancel}(i,j,\text{ship}(i,j,\text{goods}))]\,O_{ij}(i,j,\text{pay}(50))$$

if i is obliged to ship the goods and cancels the shipment after a certain date, he is obliged to pay a fine. (After a cancellation the original obligation is removed)

Finally, the logic must be precise about the duration of certain authorizations and obligations. That is, certain frame axioms must be derivable:

$$O_{ji}(\text{deliver}(j,i,g,p)) \rightarrow [\text{deliver}(j,i,g,p)]\,\neg\text{auth}(i,\text{DIR}_a(i,j,\text{deliver}(j,i,g,p)))$$

If a company has to deliver a product and actually does it, the customer is no longer authorized to request delivery of the product. (One might omit this formula or replace it with a formula that limits the validity of the quotation to a period of time).

Besides O and auth, we include one more primitive operator in the deontic specification language. This is acc, for accomplishment. acc(α) means that action α has been executed. As with O and auth, it takes typically two messages, one of both parties, to establish such a fact.

In section 3.1.2 the importance of the satisfaction stage was emphasised and the *mutual* satisfaction that must be the goal. In the dynamic deontic logic, two levels of satisfaction can be distinguished. The basic level of satisfaction is reached as soon as the obligations of both partners have been fulfilled, that is, acc(α) is true for all actions α in the contract. However, the contract may also describe authorizations for both partners to make claims in the case of dissatisfaction. The second level of satisfaction, and the real end of the interaction, is when these authorizations have expired as well.

The borderlines between the other stages can be expressed in deontic logic as well. The negotiation stage ends with the establishing of certain authorizations. The contractual stage ends with the establishing of certain (usually, mutual) obligations, making up the contract. In the case that the authorizations have been negotiated beforehand the contractual stage can be entered right away.

Although the above formulas describe an exchange between the customer and the company exact and complete, they lack structure. A contract can be formalized in two ways. We can consider it as a set of propositions in the dynamic deontic logic language (in particular, propositions of the form auth(i,α) and O(α)) or we can consider it as a set of speech acts. For instance a (very) simple contract in logic:

O_{ij}(pay(i,j,$10)) ∧ auth(i, DIR(i,j,ASS(j,i,p)))

"e.g., customer must pay $10,- and can ask the travel agency to assert some proposition about discount flights".

The same contract in speech acts:

COM(i,j,pay(i,j,$10)) & AUT(j,i,DIR(i,j,ASS(j,i,p)))

Since what can be done dynamically in the normative system is more restricted than the language itself, the latter option would imply a limitation on what is possible. This can be considered an advantage. A nice feature of this option is also that the contract is then a continuous extension of the atomic speech acts. In fact, it is nothing more than a parallel execution of atomic speech acts (authorizations, directives, ...).

6.3.3. THE ILLOCUTIONARY LANGUAGE L_{ill}

We can now bring all the previous together to define the illocutionary logic language L_{ill}. The language has the same form as L_{dd} except that the transactions β now also include the speech acts and is based on L_{ACT} instead of L_{act}. In order to be able to specify deadlines (e.g., a time before which an action should be performed or a certain state should be obtained) a temporal operator is introduced.

Definition 6.28. (L_{ill})

L_{ill} the language of logical formulas with typical elements φ and ψ is given by the following BNF:

φ :: — p | φ∧ψ | –φ | [β]φ | ⟨⟨β⟩⟩φ | PAST(α,i) | O(φ) | *B*(i,φ) | *I*(i,φ) | *I*(i,α)| PREFER(α,α')

with p a proposition; φ a first order logic formula from L_{stat},; α, α' actions, elements of L_{ACT}; and β a transaction, element of L_{tract} (as defined above).

Note: other propositional connectives such as ∨ and → are assumed to be introduced as the usual abbreviations. Also the special proposition *false* is introduced as the abbreviation of p∧–p for some p ∈ Prop (the set of atomic propositions).

The informal meaning of $[\beta]\phi$ is "doing β necessarily leads to a state where ϕ holds". The formulas defined by $\langle\langle\beta\rangle\rangle\phi$ involve a variant on the standard dynamic logic definition of the consequence of (trans)actions. $\langle\langle\beta\rangle\rangle\phi$ means that after performing the transaction denoted by β the formula ϕ holds and it does not hold before β is completely performed. The formula defined by PAST involves a type of temporal operations on actions. To keep the logic as simple as possible the temporal operators only reach over action expressions and not over transaction expressions. The meaning of PAST(α,i) is that α has actually been performed i steps ago. The meaning of PAST(α,0) in that case is "the present state is actually reached by performing α".

Note that we make a difference between the possible ways that the state can be reached and the way it actually *is* reached. Although more approaches exist that combine temporal and deontic logics (e.g., [Thomason, 1981], [Fiadeiro and Maibaum, 1991], [Maibaum, 1993], [Horty, 1996]) these approaches tend to express the deontic concepts in terms of the temporal operators. Here a different approach is taken. The temporal operators are actually "added" to the deontic logic that is used as the basis.

The last type of formulas introduce a preference relation between actions, and indicate that a certain action α is preferred to be performed over an other action α'.

6.3.3.1. SEMANTICS OF FORMULAS

The semantics of formulas in L_{IJI}, based on the semantics of transaction expressions, can be given by means of the Kripke structure $\mathcal{M} = (\mathcal{A}, \Sigma, \pi, R_{\mathcal{A}}, \leq, R_O, R_B{}^i, R_I{}^i)$.

$\mathcal{A}$ is a finite set of events.

Σ is a set of states (worlds).

π is a truth assignment function to the atomic propositions relative to a state: π is a function $\Sigma \rightarrow (\text{Prop} \rightarrow \{tt, ff\})$, where *tt* and *ff* denote truth and falsehood, respectively. Thus, for $p \in \text{Prop}$, $\pi(\sigma)(p) = tt$ means that the atomic proposition p is true in state σ.

The accessibility relation $R_{\mathcal{A}}$ specifies how transactions can change states. The relation $R_{\mathcal{A}}$ is defined as follows: $R_{\mathcal{A}} = \{R_t \mid t \text{ a trace}\}$, reflecting that R_t is the relevant entity.

$\leq$ is a function $(\Sigma \times A^*) \times (\Sigma \times A^*) \rightarrow \{tt, ff\}$. The function indicates for two state/history pairs which of the two is preferred. Here only the preference relation is used to indicate a preference relation between actions. No logic for the preference relation itself is given. However, one might intuitively think that an action α is preferred over an action β if it leads to states in which less constraints are violated or the violations are considered less harmful (i.e. which are more ideal in a deontic sense). For a thorough treatment of this type of logic see [Boutilier, 1994]. Here the preference relation is taken to be primitive.

R_O is the deontic relation that with respect to a state σ reached by trace γ indicates the ideal situation consisting of state σ' and trace γ'. R_O resembles the classical deontic relation in modal interpretations, except that not only states are considered, but pairs of states and traces. We assume the relation to be serial, i.e. for every world σ and trace γ there exists at least one pair (σ',γ') such that $R_O((\sigma,\gamma)(\sigma',\gamma'))$ holds.

$R_B{}^i$ and $R_I{}^i$ are relations that indicate belief and intention, respectively. They express that in the state σ' that can be reached from state σ for agent i the belief (respectively intention) of the proposition (or action) hold. As with R_O the relations are assumed to be serial.

The interpretation of formulas in L_{ill} in Kripke structures is as follows: The formulas are interpreted with respect to a structure $\mathcal{M}$ and a pair $(\sigma,\gamma) \in \text{Comp}(\mathcal{M})$

Definition 6.29. (Semantics of L_{ill})

Given $\mathcal{M} = (\mathcal{A}, \Sigma, \pi, R_{\mathcal{A}}, \leq, R_O, R_B{}^i, R_I{}^i)$ as above and $(\sigma,\gamma) \in \text{Comp}(\mathcal{M})$, we define:

1. $(\mathcal{M}, (\sigma,\gamma)) \models p \Leftrightarrow \pi(\sigma)(p) = tt$ (for $p \in \text{Prop}$)
2. $(\mathcal{M}, (\sigma,\gamma)) \models \phi_1 \wedge \phi_2 \Leftrightarrow (\mathcal{M}, (\sigma,\gamma)) \models \phi_1$ and $(\mathcal{M}, (\sigma,\gamma)) \models \phi_2$
3. $(\mathcal{M}, (\sigma,\gamma)) \models \neg\phi \Leftrightarrow$ not $(\mathcal{M}, (\sigma,\gamma)) \models \phi$
4. $(\mathcal{M}, (\sigma,\gamma)) \models [\alpha]\phi \Leftrightarrow \forall t \in [\alpha]\ \forall\sigma' \in \Sigma\ [R_t(\sigma,\sigma') \Rightarrow (\mathcal{M}, (\sigma',\gamma \circ t)) \models \phi]$
5. $(\mathcal{M}, (\sigma,\gamma)) \models \langle\langle\alpha\rangle\rangle\phi \Leftrightarrow \exists(\sigma',\gamma') \in \text{Comp}(\mathcal{M}) : \gamma' = \gamma \circ t \wedge t \in [\alpha] \wedge (\mathcal{M}, (\sigma',\gamma')) \models \phi$ $\wedge \neg\exists\alpha_1\ \exists\alpha_2 : (\alpha_1;\alpha_2 =_{\mathcal{D}} \alpha) \wedge \exists(\sigma',\gamma') \in \text{Comp}(\mathcal{M}) : \gamma' = \gamma \circ t \wedge t \in [\alpha_1] \wedge (\mathcal{M},(\sigma',\gamma')) \models \phi$
6. $(\mathcal{M}, (\sigma,\gamma)) \models O(\phi) \Leftrightarrow \forall(\sigma',\gamma') \in \text{Comp}(\mathcal{M})\ [R_O((\sigma,\gamma),(\sigma',\gamma')) \Rightarrow (\mathcal{M},(\sigma',\gamma')) \models \phi]$
7. $(\mathcal{M}, (\sigma,\gamma)) \models \text{PAST}(\alpha,0) \Leftrightarrow \exists t \in [\alpha],\ \gamma'\ [\gamma = \gamma' \circ t]$
8. $(\mathcal{M}, (\sigma,\gamma)) \models \text{PAST}(\alpha,i) \Leftrightarrow \exists(\sigma',\gamma') \in \text{Comp}(\mathcal{M})\ \exists S \subset \mathcal{A}\ [\gamma = \gamma' \circ S \wedge (\mathcal{M},(\sigma',\gamma')) \models \text{PAST}(\alpha,i\text{-}1)]$
9. $(\mathcal{M}, (\sigma,\gamma)) \models \text{PREFER}(\alpha_1,\alpha_2) \Leftrightarrow \forall t \in [\alpha_2], R_t(\sigma,\sigma_2) \rightarrow (\exists t' \in [\alpha_1]\ (R_{t'}(\sigma,\sigma_1) \wedge (\sigma_1,\gamma \circ t') \leq (\sigma_2,\gamma \circ t))$
10. $(\mathcal{M}, (\sigma,\gamma)) \models B(i,\phi) \Leftrightarrow \forall\sigma' \in \Sigma\ [R_B{}^i(\sigma,\sigma') \Rightarrow (\mathcal{M},(\sigma',\gamma')) \models \phi]$
11. $(\mathcal{M}, (\sigma,\gamma)) \models I(i,\phi) \Leftrightarrow \forall\sigma' \in \Sigma\ [R_I{}^i(\sigma,\sigma') \Rightarrow (\mathcal{M},(\sigma',\gamma')) \models \phi]$
12. $(\mathcal{M}, (\sigma,\gamma)) \models I(i,\alpha) \Leftrightarrow \forall\sigma' \in \Sigma\ [R_I{}^i(\sigma,\sigma') \Rightarrow (\mathcal{M},(\sigma',\gamma')) \models \alpha]$
13. ϕ is *valid* w.r.t. model $\mathcal{M} = (\mathcal{A}, \Sigma, \pi, R_{\mathcal{A}}, \leq, R_O)$, notation $\mathcal{M} \models \phi$, if $(\mathcal{M},(\sigma,\gamma)) \models \phi$ for all $\sigma \in \Sigma$ and γ.
14. ϕ is *valid*, notation $\models \phi$, if ϕ is valid w.r.t. all models $\mathcal{M}$ of the form considered above.

The first four definitions are quite standard and will not be explained any further here. In (5) the fact that the condition ϕ becomes true for the first time after performing the (complete) transaction denoted by α is defined. The definition of the static obligation (6) involves both the state and the trace (and *not* just the state). In this way, we can express that the circumstances described by PAST(α,0) are obligatory. E.g., it might be obligatory to have just done the action indicated by α. This means that the history (i.e. the trace) of an ideal world might differ from the history of the present world. This feature is used to define obligations on actions with deadlines below. It should be noted that using the semantic definition of [**any**]ϕ makes it possible to express the usual temporal operators over static formulas as given in e.g. [Emerson, 1989]. Points (7) and (8) define extra temporal operators reaching over action expressions. Point (9) defines what it means that one action is preferred over another.

6.3.3.2. OBLIGATIONS AND DEADLINES

Before a definition of the deontic operators (over transactions) for deadlines is given, first a helpful operator is introduced. This operator indicates that a formula ϕ is true as soon as a formula ψ becomes true. It is defined formally as follows:

Definition 6.30. (When operator)

ϕ **when** $\psi \equiv \langle\langle\gamma\rangle\rangle\psi \rightarrow [\gamma](\psi \rightarrow \phi)$

Using the definitions from L_{III} one general type of obligations can be introduced: the obligation with deadlines. Using this general type, some special types of obligations are defined that are often used to describe all types of deadlines.

Definition 6.31. (General obligation with deadline)

$O(\phi < \alpha < \psi) \equiv O(\alpha < \psi)$ **when** ϕ
with $O(\alpha < \psi) \equiv \langle\langle\gamma\rangle\rangle\psi \wedge dur(\gamma) = n \rightarrow [\gamma](\psi \rightarrow O(\exists i : 0 \leq i < n : PAST(\alpha, i)))$

The most general form $O(\phi < \alpha < \psi)$ stands for the fact that α should be performed after ϕ has become true and before ψ has become true. Intuitively $O(\alpha < \psi)$ stands for the fact that α should be performed before ψ holds true. I.e. if ψ becomes true (sometimes) for the first time after performing the transaction denoted by γ then it is obliged that α has been performed in the course of γ (in the last n steps).

The first specialization of the general obligation is made with respect to the begin and end conditions of the period in which the action should be performed. The definition in the general case is somewhat complicated because whether these conditions hold true might depend on the type of (trans)action that is performed.

In the case we take the conditions to be purely temporal they do not depend on the transaction performed anymore. We distinguish between relative and absolute time conditions. For the absolute time conditions we can introduce a special variable *time*. The following axiom should hold for the values of this variable:

Axiom 13. (Time variable)

$\models time = k \rightarrow [\mathbf{any}]time = k+1$

That is, all actions are assumed to take equal time and the length of an action defines the basic unit of time.

Using this axiom the general obligation with pure absolute temporal deadline can be defined as follows:

Definition 6.32. (General obligation with absolute temporal deadline)

$O(temp_1 < \alpha < temp_2) \equiv [\mathbf{any}^n]temp_1 \rightarrow [\mathbf{any}^n]O(\alpha < temp_2)$
with $O(\alpha < temp_2) \equiv [\mathbf{any}^m]temp_2 \rightarrow [\mathbf{any}^m]O(\exists\, 0 \leq i < m : PAST(\alpha, i))$

For relative deadlines the definitions are as follows:

Definition 6.33. (General obligation with relative temporal deadline)

$O(now+temp_1 < \alpha < now+temp_1+temp_2) \equiv (time=now \wedge [\mathbf{any}^n]time=now+temp_1) \wedge [\mathbf{any}^n][\mathbf{any}^m]time=now+temp_1+temp_2) \rightarrow [\mathbf{any}^{n+m}]O(\exists\, 0 \leq i < m : PAST(\alpha, i))$
and
$O(\alpha < now+temp_2) \equiv (time=now \wedge [\mathbf{any}^m]time=now+temp_2) \rightarrow [\mathbf{any}^m]O(\exists\, 0 \leq i < m : PAST(\alpha, i))$

The next specialization of the general case is in fact the type of dynamic obligation as it is used in most dynamic deontic logics. It is the "immediate" obligation, which means that the action should be performed as the next action. The immediate obligation is defined as follows:

Definition 6.34. (Immediate obligation)

$O!(\alpha) \equiv O(\alpha < now+1)$

For the following the abbreviation $PREV(\alpha)$ for $PAST(\alpha,0)$ is used. From the definitions the following equivalence can easily be proven:

Proposition 6.1. (Immediate obligation)

$O!(\alpha) \equiv [\mathbf{any}]\, O(PREV(\alpha))$

So, α is obligated if, whatever one does now, it will be true immediately afterwards that one was just previously obligated to do α. I.e. if one does $\overline{\alpha}$, a state is reached where a violation occurs, indicated by the fact that both $O(PREV(\alpha))$ and $PREV(\overline{\alpha})$ hold. Note that by definition the following formula is valid for all actions $\alpha \in Act$:

$[\alpha]PREV(\alpha)$

Therefore

$O!(\alpha) \rightarrow [\overline{\alpha}]\, (\neg PREV(\alpha) \wedge O(PREV(\alpha)))$

With the general type of obligation with deadlines it is also possible to describe an obligation that has to be fulfilled as soon as possible. This obligation is interpreted as meaning that the action should be performed as soon as no other actions with a higher "preference" are performed. The definition is as follows:

Definition 6.35. (ASAP obligation)

$O?(\alpha) \equiv O(true < \alpha < PREV(\beta) \wedge PREFER(\alpha, \beta))$

This obligation can be used when no strict deadline is given, but we want the action to be performed at some time. It resembles the "liveness" property as described in [Fiadeiro and Maibaum, 1991], except that the obligated action cannot be postponed indefinite. It has to be performed before an action with lesser importance is performed.

The last type of obligation that is described is the periodic obligation. This obligation returns every time a certain condition holds true and should be fulfilled before another condition holds true. E.g. an order should be placed after the stock of computers has fallen below 15 and before the level dropped below 5. Although this seems the same as the general obligation described above it is a bit different. The condition that the stock falls below a certain level will be true periodically (one hopes) and every time this happens an order for replenishment should be made. The periodic obligation is described as follows:

Definition 6.36. (Periodic obligation)

$PO(\phi < \alpha < \psi) \equiv \forall\ n : dur(\gamma) = n \rightarrow [\gamma](O(\alpha < \psi) \vee justdone(\alpha))$
with $justdone(\alpha) \equiv (\exists\ 0 \leq i < n\text{-}k : PAST(\alpha, i)) \wedge \gamma = \beta_1;\beta_2 \wedge dur(\beta_1) = k \wedge [\beta_1]\phi$
$\wedge\ (\forall \beta' : \beta' = \beta_1;\beta_3 \wedge dur(\beta') < n \rightarrow \neg[\beta']\phi))$

justdone(α) states that α has been done after the last time that ϕ became true. The definition of PO(ϕ<α <ψ) states that (from now on) it is always obligated to do α before ψ holds true except when α has been "justdone".

6.3.3.3. MODELLING DEADLINE EXAMPLES

The following examples of deadlines can be described in the logical formalism in a natural and concise form.

> When a secretary is hired in the order department, he/she has to pass the exam in "blind typing" within the first year.

This example is modelled as follows:

$\forall$ p : PREV(*Hired*(p,order_dep) $\rightarrow$ O(*Pass*(p,blind_typing) < *now*+*year*)

I.e., if *Hired*(p,order_dep) has just been done then there is an obligation to perform the action *Pass*(p,blind_typing) between *now* and a year time. We assume that *year* stands for an integer that indicates how many times an action should be performed to advance the absolute time with one year.

The second example shows some combinations of different types of deadlines.

> After the stock of business trip expenses forms (btef) has fallen below 10 an order should be made before the stock is less than 6. If an order has been made the delivery should follow within 5 days. If the delivery is not made in time a reminder should be sent. After the receipt of the goods payment should be effectuated within 30 days.

This example is modelled by the following formulas:

(1) O(PREV(*fall-stock*(btef < 10)) < *Order* < *stock*(btef) < 6)
(2) PREV(*Order*) $\rightarrow$ O(*Delivery* < *now*+5**day*)
(3) (PREV(*Order*) $\wedge$ [**any**^{5*day}](¬$\exists_{0\leq i<5*day}$ PAST(*Delivery*,i))) $\rightarrow$
[**any**^{5*day}]O!(*Send*(*reminder*))
(4) PREV(*Receipt*) $\rightarrow$ O(*Pay* < *now*+30**day*)

The third formula is a typical example of how the violation of an obligation triggers another obligation. This is very natural, because the violation of an obligation should lead to some rectifying action, which is usually an obligation as well.

The next example illustrates an obligation that should be fulfilled "as soon as possible".

> After the secretary has phoned to register a failing central heating system (during the winter) a mechanic should try to repair it as soon as possible, but at least within 24 hours. (From a contract between a service company and the organization).

This is an example of having an obligation to perform an action as soon as possible. In this case it might be that the service company is very busy and got several calls at the same time. In that case it is not possible to go to all clients at the same time. However, if the mechanic goes to all the clients one after the other we would say he fulfilled the obligation of the service company. The above example can be modelled as follows:

$$\text{PREV}(\textit{Report}(\text{ch})) \rightarrow (\text{O?}(\textit{Try-repair}(\text{mechanic,ch})) \wedge \text{O}(\textit{Try-repair}(\text{mechanic, ch}) < \textit{now}+24*\textit{hour}))$$

The last example illustrates the use of periodic obligations.

> The employees of the company have to be paid their salaries between the 25th and 30th day of each month.

This example can be modelled very simple as follows:

$$\text{PO}(\textit{monthday}(\textit{time}) = 25 < \textit{Pay}(\text{salary,emp}) < \textit{monthday}(\textit{time}) = 30)$$

where *monthday* is a function that returns the day of the month given an absolute point in time.

6.3.3.4. FRAME AXIOMS

Instead of definition 6.31 above, it is also possible to provide frame axioms that specify whether the obligation is carried over to the next world or not. Here the two approaches are compared.

An obligation persists until it is satisfied or violated. It is satisfied when the action is performed, and it is violated when at the time of evaluation (the deadline) it is not performed. If no deadline is given, the evaluation is assumed to be at the end of history (Judgement Day).

The frame axioms will be stated as propositions since they can be derived from the definitions given above. If α should be performed before ψ holds true then this obligation still holds if α is not performed and we have not reached the deadline ψ.

Proposition 6.2. (Frame axioms)

1. $\models (\text{O}(\alpha < \psi) \wedge \neg\psi) \rightarrow [\overline{\alpha}]\text{O}(\alpha < \psi)$
2. $\models \text{O}(\alpha < \psi) \rightarrow (\psi \rightarrow \text{O}(\text{PREV}(\alpha)))$

Proof:

According to definition 6.31 $O(\alpha < \psi)$ means that (traces in) transactions γ leading to ψ also lead to $O(PAST(\alpha,i))$. Now let $O(\alpha < \psi)$ be true at state s_1, and let s_2 be a state reached from s_1 via some action but not α. Then it is evident that all (traces in) transactions γ_1 from s_2 leading to ψ also lead to $O(PAST(\alpha,i))$, since each (trace in) γ_1 is a subtrace of some (trace in) γ starting from s_1. The only restrictions are (1) that α should be somewhere in γ_1, so it should not be done already before s_2; and (2) that γ was not empty already, i.e., ψ did already hold in s_1. These two restrictions are met by the conditional part of proposition 1.

The second proposition also follows from definition 6.31. If ψ holds, it means that the γ in the definition is empty. The definition can be reduced then to

$O((\exists i : 0 \leq i < n : PAST(\alpha ,i)))$ for n=0, which is equivalent to $PREV(\alpha)$.

Note that according to the proposition the obligation disappears when α is performed or when the deadline is reached. This does not mean that $O(\alpha < \psi)$ is false afterwards, but rather it may hold or not. This is specified simply by saying nothing about this case. The most we can say is that the obligation will not always hold:

$$\models (O(\alpha < \psi) \wedge \neg\psi) \rightarrow \neg([\alpha]O(\alpha < \psi))$$

Under a Closed World Assumption, this will be interpreted as a negation.
The two propositions can also be take together as follows:

$$\models (O(\alpha < \psi) \wedge \neg\psi) \rightarrow [\,\overline{\alpha}\,] (O(\alpha < \psi) \wedge (\psi \rightarrow O(PREV(\alpha))))$$

6.4. CONCLUSIONS

The combination of dynamic deontic logic and illocutionary logic gives a formal framework and integrated semantics that provides for the precise description of the concepts used. Although deontic logic has been applied in the field of ISs before, the dynamics of normative systems have received very little attention. Here we explored the way deontic statements are created and adapted in communication processes, and the role they play in the regulation of communication itself. It is shown how authorizations can be requested, granted and also retracted, thereby creating a dynamic environment for the establishment and derogation of authorized norms.

It is a characteristic of messages that they seldom stand on their own, e.g., a request is typically followed by an acknowledgement or commitment. Therefore messages can be organized in transactions on basis of the effect of one message and the preconditions of the next message. In this way protocols can be built for sequences of messages that appear often. However, the communication is not limited to protocols that are predefined. With the logic formulas the exact effects of each message in a protocol can be inferred. Both in the case when a certain message is expected (and thus usually authorized) as well when it is unexpected. It is also possible to calculate the precise effects of the complete communication protocol that is used. This makes it easier to react to breakdowns in the communication.

Agents have an agenda that contains the actions they are supposed to be performed at due time, specified by the obligations for those actions. We have shown how these obligations come about as the result of the performance of speech acts (and the relationships between the communicating agents). The formal meaning of the agenda is a set of deontic temporal constraints. In the context of the proposed agent architecture, this set functions as a program.

The present work opens some areas for further research. In particular the temporal aspects of the language are rather primitive. We assume that all actions take the same amount of time, which, of course, is not very realistic. A second area for further research is the influence of the deadlines on the (planning of) actions of the system. As described in section 5.3.2.1 here the naive strategy of planning the action whose deadline expires first as the first action to perform is adopted. However, more intelligent planning algorithms are called for.

CHAPTER 7

TOWARDS A DESIGN METHODOLOGY

As stated in chapter 1, the way of working of people and organizations changed drastically the last decade. Most IS development methods do not correspond well to the new ways of thinking (paradigms) needed for the development of modern ISs (e.g. CIS) supporting the new way of working. Traditional IS development methods emphasize the modelling of the object system (that part of reality that represents the data of the domain and processes that work on it) almost exclusively. Furthermore, they are either based on too strict a data view, or too strict a process view of the object system. Although 'new' modelling methods based on the object-oriented paradigm overcome this last problem, they still suffer from the first one: the strict focus on the object system.

In this thesis the focus is on the use of ISs in and between organizations to support the coordination of activities. Therefore we are not only interested in the object system but also in modelling the organization (including users and (existing) ISs), called the subject system. From the language-action perspective we learned that this means that both information and communication, and their (coordinating) role in the functioning of the dynamic organization are to be modelled.

This chapter proposes a modelling methodology describing how to improve the design of CISs supporting organizational communication. It does not give completely new cookbook methods with definite steps on how to find all objects and build the system. The aim is to produce an abstract model of organizational communication as a basis for formulating requirements of a supporting software system (the envisioned CIA). For modelling purposes the specification language is complemented with graphical models. Besides the CoLa models existing LAP-based modelling techniques can be used for constructing the different models, sometimes adapted to provide a closer match to the CoLa concepts introduced in chapter 5.

In the first section an overview of the methodology is given, describing the different models and their relationships. In section 2 we take a closer look at the separation of Environment of Discourse and Universe of Discourse, which is the basis for the modelling approach presented here. Section 3 describes the different phases in the methodology, the modelling processes, in more detail. In section 4 the different diagram techniques for constructing the models are given and section 5 contains a comparitive analysis with related work.

7.1. METHODOLOGY OVERVIEW

This section presents an overview of the proposed methodology for CIA design. The methodology is strongly communication-driven, and it combines models of organizational communication with a model of information content of speech acts.

Although the goal is to describe a methodology that can be used to develop automated IS that support the communication and coordination of business activities in an organization, the first steps can also be valuable if no automated system is implemented, in that it clearly describes the authorization and communication relations that can be used to improve on the way business is conducted. In this respect the methodology can be used for business process (re)design and is similar to DEMO ([Dietz, 1994a,b]), BAT ([Goldkuhl, 1995]) and Action-Workflow ([Medina-Mora et al., 1992]) (section 7.5 contains a comparative analysis with these methods).

The methodology is strongly influenced by the distinction between EoD and UoD (see next section) and drives the construction of the different models as is shown below. The different phases in analyzing organizational communication and developing supporting systems for it are represented in figure 7.1.

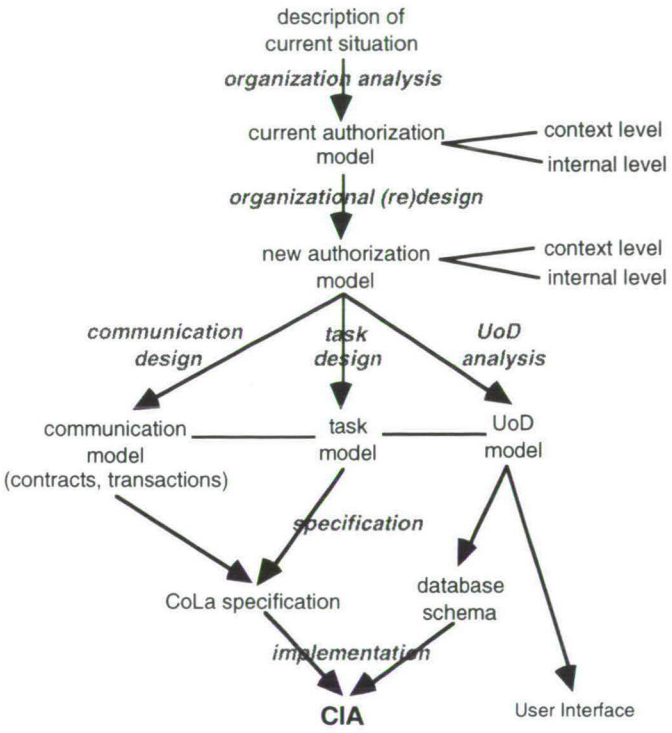


figure 7.1. Methodology for CIA design.

From a description of the current situation we start with an organizational analysis phase that describes the communication lines in and between organizations. The goal is to obtain an overview of present authorizations and obligations of the actors in the environment. In organizational (re)design we can propose improvements or changes to the organizational authorization model. This (new) model is the input for the communication design, task design and UoD Analysis phases, in which we describe in more detail the communication between actors (the contracts and transactions), the tasks of the actors and the information they work on from the domain, respectively. The results of these phases will serve as input for the (formal) specification phase in which the CoLa specification (chapter 5) and also the database schema are generated. Finally, using this specification, the system (CIA) can be implemented, preferably based on the provided agent-architecture (chapter 4).

The phases are represented as being sequential or parallel. Feedback between successive phases is not represented in the figure. In fact, the process is of a more iterative nature than a waterfall-like nature; in working out a model one often goes back to a previous step to refine the models there and then forward changes.

This also concerns the question to what detail concepts should be worked out. If the techniques are used to obtain models for the goal of analyzing the current situation and see how to improve on that, not all aspects and details have to be worked out. As an example in communication modelling we can first look at the success line of communication, and later add the necessary communication links and communicative actions (with related authorizations and obligations) in case of failure. In case the models are used as input for the specification of an automated IS to support the communication it is required that all relevant details are modelled, i.e., all relevant messages that can be exchanged should be modelled in detail.

The goal is to obtain models describing the situation understandable for all parties involved. The order in which the models are obtained is not prescribed, but the process outlined below is the most logical way of obtaining all the models.

The different steps are described in more detail in subsections 3.1–3.8. The proposed modelling techniques used to construct the models are explained in section 4. First, however, the relationships between the different models is explained.

7.1.1. RELATIONSHIP BETWEEN MODELS

The Authorization Model is the first model to construct. Figure 7.1 illustrates this can be described on a context and an internal level. Initially a high level Authorization Model will be constructed, specifying only the communication lines between the organization and others in the environment (context level) and between organizational units and individual actors (internal level). The communication lines are described by Business Contracts following the business logic from chapter 3. They can be described by a refined Authorization Model specifying the actors and authorization and obligation transactions between them. Here changes can be made in the communication lines (with associated authorizations and obligations) according to the (re)design phase.

From this we can construct the Communication Model, consisting of the Contract Model and the Transaction Model. The Contract Model specifies each contract in detail, i.e., the deontic states (authorizations and obligations) and the transactions that create (or modify or destroy) them (section 5.2.4). The transactions in the Authorization Model and Contract Model can be decomposed in elementary communicative actions, also called speech acts (section 5.1.4). These are modelled in detail in the Transaction model.

The tasks of the agents are modelled in the Task Model. The elementary subtasks are either initiations of transaction, which again are described in the Transaction Model, or actions, modelled in the UoD behaviour part.

The information content of speech acts (core predicates) is modelled in the UoD. Speech acts make reference to actions, described in the behaviour part of the UoD Model, and objects, described in the Object Model part of the UoD Model.

For ease of modelling the models have associated graphical diagrams. The authorization model can be represented by an Authorization Diagram (AD), the communication model consists of a contract model represented by a Contract Diagram (CD) and a transaction model represented by a Transaction Diagram (TrD), the task model can be represented by a Task Diagram (TD) and finally the UoD model can be represented by the UoD Diagram (UoD).

The relationships between the diagrams are illustrated in figure 7.2.

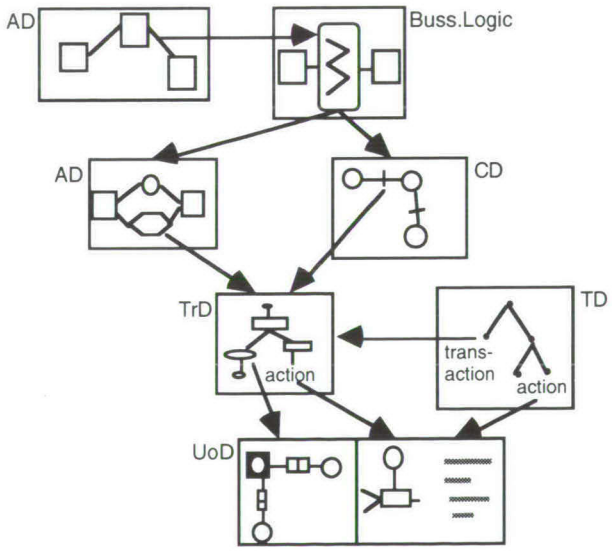


figure 7.2. Relationships between diagrams

Figure 7.3 illustrates the relationship between communication (speech acts) in the communication diagram and information in the UoD diagram. On the left side the general relationship between speech acts and object types is given. Remember that every speech act has an information content (section 2.1.1 and section 6.3.1). The core predicate of the

information content is represented by an object type in the UoD Model. If there is a relationship between two speech acts, there also is a relationship (represented by a fact type) in the UoD Model between the object types. The right hand side gives an example business communication of sending an invoice (request for payment that is followed by either the assertion that it is paid or a complaint about the invoice, which is followed by the assertion that it is paid). The associated object types are invoice, payment and complaint, respectively.

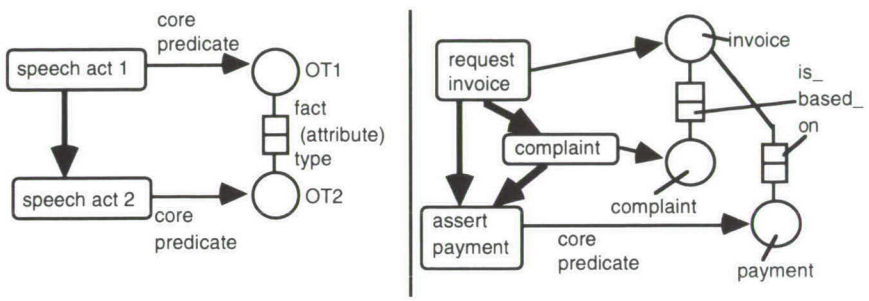


figure 7.3. relationship between speech acts and UoD objects
(after [Holm, 1996, fig.III-19, p.155])

Below this separation between modelling communication and modelling what is communicated about is described.

7.2. UoD/EoD SEPARATION

When analyzing a domain and building a conceptual model of it for purpose of IS design, it is necessary to make a distinction between two kinds of relationships between the IS and the domain. On the one hand the IS holds information (data, rules) *about* the domain (object world). On the other hand, the IS will be situated *in* some organizational environment where it supports the communication and coordination (subject world). Since the conceptual model should describe all relevant aspects, rules, etc., of the specific domain (environment of the organization, or a part of it) two aspects or "projections" of the conceptual model can now be distinguished: the communication structures, and the content of the communication.

The former is represented in the so-called *Environment of Discourse (EoD)*, and the latter in the *Universe of Discourse (UoD)*. The EoD describes the discourse (linguistic agents (human or computerized), message types, and rules that prescribe and describe the communication) itself as a process without going into the contents. What is said, and more in particular the meaning of these terms, is described in the UoD. Whereas the IS itself is usually not found in the UoD —the IS takes an objective stance— it is the central object in the EoD. The EoD/UoD distinction is important: it separates clearly the activities or operations of an organization from the things operated upon. Of course, the two domains are closely intertwined, because the discourse going on in the EoD is about the UoD.

In this thesis an organization is considered to be a system of acting and communicating actors. A description of the organization consists of the communication links between the actors and between them and (the actors in) the environment of the system, represented by the EoD. The part of the world to which the acting and communicating of the actors is related is represented by the UoD.

The UoD/EoD distinction can be illustrated by modelling practice. E.g., in advanced EDI (Electronic Data Interchange) specification methods a distinction is made between *data model* (to describe the meaning of the content of messages) and *message protocol* (to describe the external behaviour of two organizations per transaction) [ISO, 1989].

Traditional analysis methods (cf. [ISO, 1982]) and also today's popular OO analysis methods do not make the EoD explicit. It is peculiar that the EoD is not worked out in any of the approaches described in the ISO report. In recent proposals, this is repaired. E.g., in the field of knowledge acquisition, [Mizoguchi, 1993] describes ontology research for the sharing and reuse of knowledge bases. He separates content ontology (divided in task and domain ontology), communication ontology and indexing ontology. Also DEMO ([Dietz, 1994a,b]) takes the separation of object (UoD) and subject world (EoD) into account.

The communication modelled in the EoD takes the form of speech acts between interacting subjects, and the relationships between these speech acts. For this reason, most traditional methods for dynamic modelling are not appropriate. E.g., State Transition Diagrams are not very useful since they focus on one object only and hide the interactions. Data Flow Diagrams (DFDs) make a distinction between processes and data. However, this distinction breaks down in the case of speech acts since these are actions and data flows at the same time. We therefore need other modelling techniques.

Existing traditional techniques for UoD modelling, like ER ([Chen, 1976]) or NIAM ([Nijssen and Halpin, 1989]), suffer from the fact that they do not support the modelling of behavioural aspects or interactions. They only describe the static aspects of the UoD: objects, attributes, relationships and static integrity constraints. To model the behaviour of the UoD, different non-integrated techniques have to be used.

Generalization, aggregation and encapsulation of static and behavioural aspects of objects are techniques that are commonly adopted in conceptual modelling, and are often combined in the so-called Object-Oriented approach. 'Pure' OO models allow to describe the behaviour of the UoD as an integral part of object modelling, but suffer from the fact that the interactions between objects are hidden in the object definitions (its 'methods'). Interactions between objects are described from a local point of view, each object (type) describes its own interactions with other objects. As a result it is hard to obtain a global view of all interactions between objects. This problem is known as the *ravioli problem* where there are a lot of tiny well-structured objects that are easy to understand in isolation, but whose interactions are nearly impossible to decipher ([Taylor 1990]). Distributing the interactions among the object types also reduces the reusability of these objects in other domains because of the domain-specific interactions incorporated in the objects. For these reasons an OO model in which it is possible to make relationships and interactions between object types explicitly visible is more useful for UoD modelling. An example of such a model is NORM ([De Troyer, 1991]), described in section 7.4.3.1.

7.2.1. ENVIRONMENT OF DISCOURSE EXAMPLE

This paragraph illustrates the Environment of Discourse concept. Again let us consider the business trip case. The EoD model of the business trip case describes:

- the *agents*, individual like the travel agency, or generic, like "customer";
- the *message types*, such as "request for hotel reservation", "assert credit limit", and "authorize payment";
- *obligations and authorizations* of the EoD agents, such as the obligation of the airline company to reserve a seat on the plane after a ticket is issued for it;
- obligations and authorizations of the IS itself as one of the agents of the EoD, in this case the travel agent CIA, e.g.: "If the airline company requests payment for a ticket I ordered, then I have the obligation to pay";

In our example the travel agency is the central agent in the EoD; others are the hotel (with whom reservation requests, etc. are exchanged), the airline company, the customer and a bank. The airport is not mentioned at all in the description, but since the airline company has certain responsibilities for making sure the airline can reach the destination, there must be an agent with which to settle this. E.g., the "permission to land" is supposedly declared by the airport. Also the insurance company is not mentioned, but one can imagine that both the customer and travel agency itself are insured against mishappenings causing the cancellation of the trip. It depends on the tasks and transactions to model whether these agents should be taken into account. A guideline might be to identify all possible agents in the high level authorization model. After the authorizations and obligations are described it can be decided to not model the communication with them further.

The UoD specification, in this case, looks at the contents of the messages, and finds terms like "ticket", "hotel room", "flight schedule". It has to explicate whether these terms represent object types, or attributes, or relationships, and assign them a meaningfull definition as agreed upon by the agents in the EoD. E.g., the "payment due" is the sum of the costs of making the reservation, taxes, costs of insurance, etc.

7.3. METHODOLOGY STEPS

In this section the phases that can be distinguished in the methodology are described in more detail. It follows figure 7.1 from top to bottom and left to right. The diagram techniques mentioned are described in more detail in section 7.4.

7.3.1. ORGANIZATION ANALYSIS

The purpose of this step is to discover the *essential* communication structures of the organization. By essential we mean that we abstract both from technological issues and reproduction of data, leaving us with 'creative' communications only. Since we are only

interested in authorized communication behaviour, the first step is to model the authorizations (permissions, prohibitions) and also the obligations of the communicating actors. This model is usually the most difficult to get, since authorizations and obligations are often left implicit, but also the most crucial for the success of the business processes and the IS supporting them in the organization. With every obligation the strength of the obligation should be recorded, i.e. whether it is the result of communicative behaviour based on power, authorized, or other (charity) communication. This also holds for the authorizations.

Both this and the next model can be described at two levels. First the 'environmental or context level' in which the organization is described as a whole, communicating with and committing to external actors in the environment. Secondly, the 'internal level' where the authorization model is decomposed according to distinguishable units in the domain (e.g. company departments, organization units).

Subtasks of this step are:

- analysis of the organizational environment:

  Collecting a *description of the current situation*, involving the identification of external parties and the characterization of the kind of authorization links with the focal subject organization, i.e., an overview of all recurrent communications. This can be done using the authorization modelling technique.

- analysis of the internal organization:

  The focal subject organization is decomposed into units, and ultimately, human or automated actors. The authorization links from the previous subtask must be decomposed as well. For this also the authorization modelling technique can be used. For every actor-pair the communication links between them are modelled in more detail using the communication modelling technique. This describes the authorizations, obligations and accomplishments of the communicating partners and the transactions leading to them.

- analysis of objectives:

  This involves the identification of current problems and the setting of goals for the near future.

Outputs of this step are the *current authorization model* (context level and internal level), *initial communication model* and the objective statement. The current authorization model describes the current obligations and authorizations of the different actors, also called subjects (human as well as computerized), in the domain.

7.3.2. ORGANIZATIONAL (RE)DESIGN

In this step changes to the communication structures are introduced. This can be done by introducing new communication lines, or shifting existing authorizations and obligations to other actors to improve the efficiency or effectiveness of the business processes, or the introduction of new actors (in particular an automated system to support the business process). These three situations are illustrated below.

Organizational redesign must be an integral part of system development. Just automating existing communication lines is often a bad choice, and often much more can be gained by changing the responsibilities of users and systems. The analyst should not only envisage and design the new reality, but also has to indicate how the current reality is transformed to this new reality. The new reality does not fall from heaven. As [van de Weg, 1995] states: "In IS engineering we have to deal with the transition from an existing (whether or not automated) IS to a future-to-be automated IS. This transition may have an impact on the way of working and/or the way of thinking of users of both ISs. Modelling of a to-be automated IS means designing a new reality. The IS is to be implemented accordingly". E.g., if the new situation involves new obligations (to certain agents), these have to be assigned. In the case of system development, it is necessary to *embed* the new system in the current reality. This is achieved by performing a limited number of operations on the existing communication structures. These operations are not only performed at "design time", but, in a rapidly changing environment, are the subject of continuous negotiation and adaptation by the partners. The partners do not only communicate according to the business processes (sending orders, paying etc) but also change the process itself, e.g., by providing the other party with new authorizations.

7.3.2.1. CHANGING THE COMMUNICATION STRUCTURES

An example of the introduction of new communication lines is the introduction of a Just-In-Time (JIT) strategy in the organization. Instead of reporting the stock supplies to the distributor (only) who then notifies the supplier, a goods-selling organization (e.g. a supermarket) now directly can notify the supplier, cutting out the redirection through the distributor. This introduces a new communication line between the sales-organization and the supplier, however this concerns information flow only, no change in the authorization structure are made. Figure 7.4 gives a graphical representation of the example.

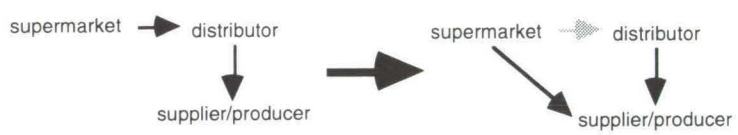


figure 7.4. New communication line example

Figure 7.5 shows an example where a change in communication lines also includes a shift in authorizations. In the original situation the bank clerk can handle all money withdrawal requests of customers up to $1000,- (he/she has the authority to process such requests at all times). However, for requests over $1000,- the bank clerk must ask permission to the bank manager, who can grant him permission to proceed. If the request is for an amount over $10.000,- the bank manager himself has to get permission from the head-office. The head-office can give the bank manager the authority to process the order, who in his turn can give the bank clerk the permission to proceed (the bank manager thereby *delegates* the authority of the head-office to the bank clerk).

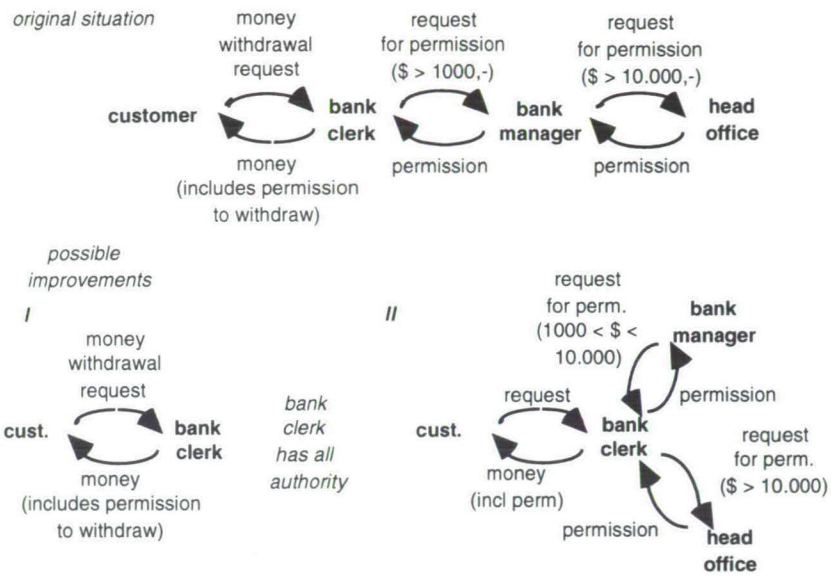


figure 7.5. Shift in authorizations example

One alternative to improve on this situation (where six communications are necessary) is that the bank might decide the bank clerk is granted permission (has from now on the authority) to handle all requests, and justification and evaluation follow afterwards, including the correction of possible wrong-doings. Of course this introduces a greater risk for the bank but it takes this for granted, because it expects a greater gain from lowering the communication costs and freeing the bank manager and the head-office from handling withdrawal requests and let them concentrate on other business. Another (less risky) solution is where the bank clerk can direct withdrawal request over $10.000,- directly to the head office, who then can give the bank clerk the authority directly. (Of course, other solutions are possible too, but here we want to show how communication and authorization lines are changed in different situations).

Starting with the current authorization model, we hypothesize that changes in the communication (authorization) structure are typically of the following types:

- *mediation*: one communication link is split up in two by putting one new (assistant) communication subject in between.
- *delegation*: a communication (and/or authorization) link is moved from one subject to another (assistant) subject.

The subject that is introduced can be either human or automated.

To support mediation and delegation, we need operations of *introducing* (and its counterpart, removing) a subject, *authorizing* a subjet (and *retracting* an authorization) and *assigning/retracting* an obligation (for some task). Note that these operations are not technical, but organizational in nature.

The introduction of an assistant subject can be necessary if in the current situation the role of one of the actors is overloaded and one wants to separate the different roles the actor has (with the corresponding authorizations and obligations). This is a standard operation when decomposing the organization after having modelled the communication links between the organization and its environment.

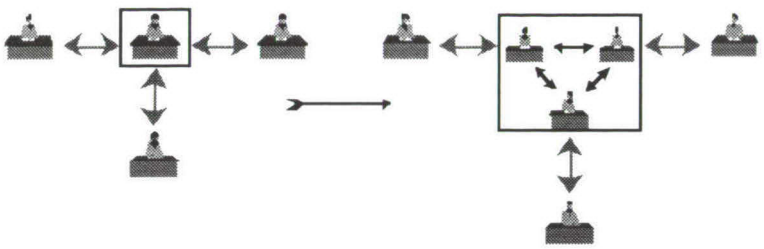
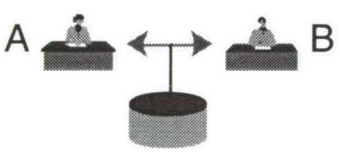


figure 7.6. Decomposition of communication roles.

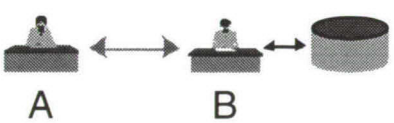
When we apply this to the introduction of automated systems in the business process to support the communication or tasks of the actors, four situations can be distinguished:

i) Electronic communication support:



Here the communication between two actors is supported by an automated system, e.g. an email system.

ii) Electronic task support:

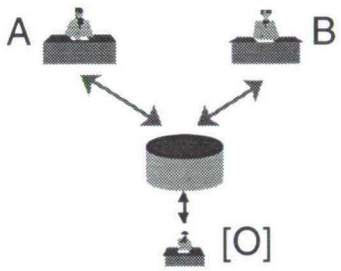


Here the task of one of the actors is supported by an automated system, e.g. in the case where the secretary uses an IS to record appointments for her boss.

In these two cases no shifts in authorization or obligation are introduced. In the second case there is a new communication line (that between B and the supporting IS), however this has no influence on the existing communication between the actors. Of course, for modelling these situations and designing such systems all communication has to be specified in detail.

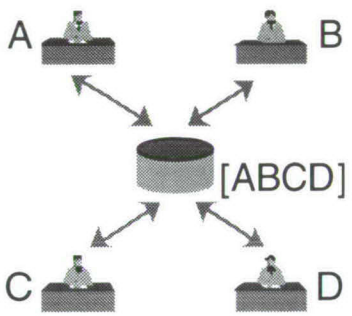
The two other cases do have consequences for the authorizations and obligations in the discourse. These are:

iii) Mediation through automated systems:



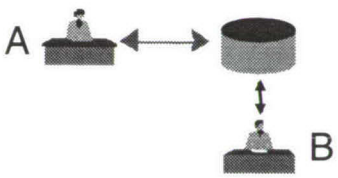
Here the communication between two actors is changed into two new communication lines between the actors and a new system. Often the system falls under the responsibility of a third party. This case describes the situation where two actors communicate through a central database in an organization, or, of a broker in electronic commerce. New authorizations are introduced, e.g. if actor A stores something in the system (if he has the authority to do so) then actor B must have the authority to access that information.

A special case is where a system is introduced to support the work of and communication within a group and the responsibility for the system lies with the group and not with one person.



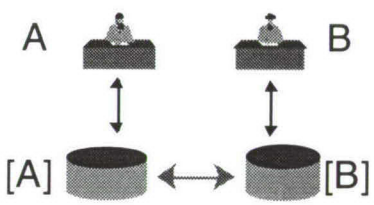
Such solution is only useful if the tasks of the actors shift and now become the tasks of the group. The authorizations now are delegated to the group too. An example of such a system is a co-authoring tool. This is a highly complex situation and we will not go into this here any further.

iv) Delegation:



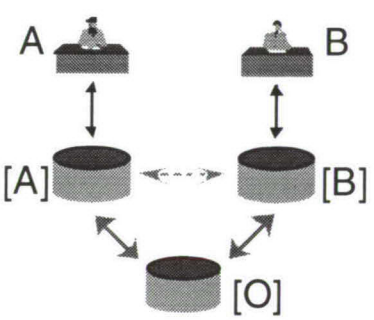
Here one of the actors is 'replaced' by an automated system. The new communication line is between actor A and the system. Actor B has delegated (some of) its tasks to the system. The system falls under the responsibility of actor B but it manages the authorizations and obligations of that actor. E.g., in the case of an automated booking system, where the travel agency can book reservations directly with the system of the airline company.

A special case of this is in (advanced) EDI or electronic commerce where both actors are supported by automated systems. The communication between the actors is replaced by communication between the systems.



The basic types can be combined to account for more complex structures. For example, the mediation type can be levelled, with one mediator (server database) for each department, and one global mediator (server database) for the company.

Also, a combination of the delegation and mediation type is possible:



The actors do have responsibility for some data, but there is also a common portion that is assigned to a third party. Two variants can be distinguished, one where the clients can communicate directly with each other and the other where they cannot. Note that when there is no server, we come back to the delegation type again.

Both the mediation and standard delegation type only contain one system. In these cases, the interfaces between system and human actor should be designed carefully, with proper attention to the authorizations. In the group mediation and the special delegation (EDI) type, and also in the combination of the mediation and delegation type there are several systems. In these cases, the interfaces between the systems are of particular importance. The interfaces with the human actors, the system owners, are less relevant because they are responsible for the system themselves.

Although system components are now introduced to the communication, the model is still abstract in the sense that a 'system' can still be implemented in different ways. E.g., it can be one DBMS, or one Distributed DBMS, with or without replication etc. These can be varied to achieve the best possible performance. This is the concern of the technical design and implementation phase. It is abstracted from here where we only concentrate on the conceptual level, describing changes in communication and authorization.

7.3.2.2. MODELLING THE NEW SITUATION

Due to the introduction of the (new) IS(s) and other organizational changes the obligations and authorizations change. To model the impact of these changes and the impact of the introduction of the IS(s), we can look in the organizational (re)design phase at the communication between the environment and the IS(s), and changes in the communication between subjects in the environment.

The new (desired) authorizations are modelled in the *new authorization model*. As in the organization analysis, we first consider the organizational environment, after which the focal subject can be decomposed into smaller units.

Subtasks of this step are:

- design of the organizational environment:

  Changes to the focal subject organization are introduced, thereby possibly changing the authorization model, leading to new or changed recurrent communications. For this the authorization modelling technique can be used.

- design of the internal organization:

  The changes introduced on the organizational environment may have different implications for the authorizations of and communication between the subjects at the internal level. However, changes can be limited to the internal level only. For this the communication modelling technique can be used.

Output of this step is the *new authorization model* (environmental or context level and internal level) and a new communication model. These model gives the new authorization links between the agents and obligations that follow from them, again the strength (power, authority, charity) should be modelled.

7.3.3. COMMUNICATION DESIGN

Every subject (human or system) can be involved in several communication situations at any one time, e.g. the travel agency has to deal with the customer, the airline company, the hotel and the bank. Every communication situation describes the set of speech acts performed by the communicating subjects. The set of speech acts that is treated as a logical whole is called a *discourse*.

Obligations and permissions belonging together logically (agreements belonging together as a unity, e.g., order and payment) are grouped in *contracts*. The authorization model can be described by a set of authorization contracts. These contracts can be developed for the organization as a whole, but also bottom-up and per occasion, for every discourse. The authorization model contains high level specifications, it only describe the obligations and authorizations of the subjects in the EoD with respect to communicative actions. Each obligation or authorization may require several communicative acts (speech acts) between the involved subjects. The communicative actions are described by *transactions*, specifying the messages and their synchronization (dependencies). The communication model contains the contracts and transactions for every discourse.

This step aims at describing all communications in detail. Subtasks of this step are:

- completion of the communication model:

  All relevant authorizations and obligations and accomplishments are modelled for every communication link between two actors.
- realization of the communication model:

  The communication model is realized (i.e., described by means of transactions).
- construction of the transaction model:

  For every transaction the speech acts that are performed and their synchronization are modelled.

Output of this step is the *communication model*. consisting of contracts and transactions for every discourse.

The authorization model together with the communication model(s) give a formal description of the behaviour and the communication interactions in the EoD. We call this the *EoD model*.

Building the EoD model is not a simple task, since we must be clear about the permissions and obligations of the organizational agents. However, this is considered an *advantage*, because it is exactly one of the most important functions of conceptual modelling and because agreement between the partners on these structures is critical for the success of the subsequent phases of design and implementation.

7.3.4. TASK DESIGN

If a subject's tasks are to be delegated to an automated system (the IS) the tasks of the agent have to be modelled carefully. This consists of identifying both the private actions the agent can perform and the communicative actions (the initiation of transactions). This

step can be repeated for every organizational role that is to be supported by the agent. Tasks can be described at different levels and consist of a number of subtasks. Subtasks of this step are:

- hierarchical task analysis:

  High level tasks that are delegated to the agents are described and (possibly) decomposed into subtasks. For each task its goal is specified and also the basic (temporal) relations between the subtasks are specified. This step is performed recursively for each subtask.

- detailed task analysis:

  For each goal alternatives can be specified. An important aspect in task specification is the modelling of deadlines (the task has to be performed before a certain time or event). Furthermore, (result) dependencies between subtasks and a compensation and contingency plan can be given.

Output of this step is the *task model*.

7.3.5. UoD ANALYSIS

After the EoD modelling or in parallel, the UoD is modelled.

The first purpose of the *UoD modelling* is to derive a (formal) description of the content of the communication in the EoD, being the domain information. The second purpose is to give operational descriptions for the elementary subtasks of the IS (or private actions). The description is called the *UoD model* and is given in terms of objects (both structural **and** behavioural properties) and explicit relationships. The UoD model is more than a static information model, and instead should be viewed as the information **and** action model. Behaviour of objects is modelled by means of methods and triggers. E.g. in the business-trip case the UoD model contains both the object flight-schedule, as well as the get(flight-schedule) and put(flight-schedule) operations, which are private actions of the airline company IS. Also in the UoD model the bill (from the hotel) is described, together with operations to record the result of certain update events, such as the message from the hotel that the bill is $200,-.

Input for the UoD modelling comes from the authorization models as well as from the description of the current/new situation. There is a close relationship with the task model in that the elementary tasks are actions that are to be represented in the UoD model.

Subtasks of this step are:

- conceptual object modelling.

  For each IS and for each of its discourses, the object type model and the behaviour of these objects of the UoD are determined.

- integration of the different object models.

  After the first subtask, the models of the different discourses of one IS are merged into one.

Output of this step is the *UoD model*.

Although it is not yet integrated in the approach presented here, the Lexicon could play an active role in UoD analysis. On the one hand (section 4.3.7) the lexicon contains the terminology of a certain domain, it defines the conceptual definition of the domain concepts. The UoD model can be used as input for this. The information model describes a concept in terms of attributes and relationships with other concepts. This can be used to give a (linguistically expressed) definition of the concept. In circumscribing the terminology for a particular application domain, the designer might draw on available terminology for the generic domain, i.e. use general lexicons and dictionaries.

On the other hand the lexicon can be used in UoD modelling, in that only concepts can be used in domain modelling that have a definition described in the lexicon. See [Burg and van de Riet, 1995] for a method that applies this idea. However, it means that a lexicon should be present (and filled) before UoD modelling starts. Using the lexicon this way does not have to be restricted to UoD modelling, since the lexicon contains *all* concepts in the domain including the message types etc., it could also be used in EoD (authorization and communication) modelling and task modelling.

The three models (communication, task, and UoD model) are closely interrelated: the communication model describes transactions the tasks in the task model draw from, the contents of the messages from the communication model is specified in the UoD model, and also the elementary actions from the task model are described in the 'action-part 'of the UoD model. All models draw from the restricted terminology from the lexicon.

7.3.6. SPECIFICATION AND DATABASE DESIGN

The communication, task and UoD model are input to the specification phase.

One goal is the formal specification according to CoLa of transactions, contracts and tasks. In the implementation phase this is used to fill the knowledge bases of the CIA. The second goal is the specification of the system behaviour, the database schema (the specification of the objects in the database) and the operations that work on it. As said above, in order for the IS to function properly its elementary tasks (private actions) should be specified. Furthermore, the CIA should keep track of what part of the information model is updated by what speech act, and also how the knowledge bases and agenda should be updated as the result of speech acts, e.g. the creation of obligations because of a commitment made. These operations are part of the UoD model and can be specified in a formal language as is shown in section 7.4.3.1.

Subtasks of this step are:

- CoLa specification:

  The transactions, contracts and task models are translated into the formal specification language CoLa (defined in chapter 5). At the moment there is no tool available that automagically maps the models to the CoLa specification language. However, the models contain all concepts from the specification language. The transaction model can be mapped to Trans, the contract model to

coLa , and the task model to TaLa. In the mapping process special attention should be paid to addition and complete description of constraints and deadlines.

- database schema and operations specification:

  For each object in the UoD model, it is determined whether it should be persistent or not. This can be expressed in the form of epistemic constraints: what the CIA should know. This includes the acceptability of unknown values. From the UoD model, the database (information objects) schema is constructed and operations working on it. Private actions of the agent are specified as software functions.

  The UoD model and the database schema may be very similar, however, in the database schema, design aspects will be taken into consideration, e.g. some objects from the UoD model will not be persistent. The database schema can be described with the same technique used for UoD modelling, namely NORM (see section 7.4.3.1). In fact the database schema is a NORM schema with more detail and design choices. Also the constraints should be specified in more detail, and the methods should be described completely.

The outputs of this step are the *CoLa specification* and the *database schema* (including operations specification).

7.3.7. IMPLEMENTATION

This step describes the implementation of the CIA(s) and user-interface(s).

The CoLa specification and the database schema (including the operations specification) are the starting point for the actual implementation of the IS into the target hard- and software. As described in section 1.1.3 we feel an agent implementation fits this design best. If the agent architecture from chapter 4 is used the specifications are used to fill the knowledge bases of the CIA. In appendix C a prototype CIA is described.

Although not yet done the implementation of the Lexicon deserves some special attention. It is possible to implement the whole Domain lexicon as part of the agent, however the tasks of the agent usually concern a restricted set of discourses (e.g. the hotel agent does not have to know about tickets, that are vital for both the travel agency and the airline company agent) and therefore we can decide to only implement that part of the lexicon that concerns the discourses the agent is involved in.

The EoD and UoD models are also input for the implementation of the user-interface of the CIA. Although not mentioned before, the user-interface is an important part of the CIA. In the past, interface development has been seen as something that had to be done after the software was developed, but with the advent of user-centred design approaches the interface has been given a lot more attention. A Language-Action Perspective on IS development can contribute to interface development since besides task modelling (as usual in interface development) we also have the opportunity to describe the communicative behaviour (both on authorization and obligation level, and transaction and message level) between user and system.

Research is being undertaken to extend the UoD modelling technique to provide an information model of a basic user interface, along the lines of [van den Boogert, 1996] that describes research on extending NIAM for interface modelling, but more research has to be done to fully integrate this with the approach presented here.

Subtasks of the user-interface implementation step are:

- determination of I/O devices:

  The I/O devices with which the electronic subject communicates have to be determined. They may be paper, light pen, etc.

- specification of the interface objects:

  Interface objects are defined as subclasses of interface object types defined in some library. An interface object typically corresponds with one communication (speech act).

- specification of the interaction syntax:

  For each interaction between the subject and other agents and interfaces, the syntactic form has to be determined, including keystrokes and other events from the I/O devices.

The outputs of this step is the *CIA* and the *User-Interface*.

7.4. MODELLING TECHNIQUES

In this section the diagram techniques used to construct models of the domain are described. The first subsection focuses on techniques for modelling the EoD, consisting of authorization and communication models. The second subsection focuses on techniques for modelling the tasks and the third subsection describes the UoD modelling technique. With every technique examples are given, mainly from the business trip domain. Again it should be stressed that no new techniques have to be invented, but existing techniques can be used. At some points, however, these are adapted to get a closer match with the formal specification concepts from CoLa.

7.4.1. EOD MODELLING

The overview of the methodology showed that EoD modelling starts with high level models of the domain concerning the essential communication (authorizations and obligations). The authorization diagram is described in the first subsection. Following this, a more detailed communication model is constructed, giving the communication contracts and also the communication transactions.

One should not confuse this approach to communication modelling with conversation modelling as used in natural language understanding and natural language interface design. In conversation modelling attention is given to factors such as feedback, turn-taking, things the communication model tries to abstract from. In the CoLa methodology presented here, communication is modelled at three levels: first, we have the authorization model, then, by refinement, the communication model, and then the CIA specifications.

A general modelling principle is the use of a classification hierarchy. The subjects in the EoD can be organised in a specialization hierarchy. E.g., a travel agency clerk IS-A employee IS-A person. In this hierarchy more specific subjects inherit the authorizations from their more general parent(s). Subjects can also be decomposed (aggregation structure). In this way, it is possible to specify authorizations top-down, starting at the company level and then going down to for instance department level down to the individual subjects.

Note: The "actors or subjects" here can play what in conceptual modelling is usually called "roles". It is possible that one human or electronic subject acts as many actors. Some of these subjects are also object types in the UoD, e.g. a travel agency clerk can be both a subject (interacts with the IS in case of booking a trip) and a UoD object (a representation of the clerk in an IS for salary administration). Independently, every subject will also have to occur as an object in the database, because this is needed at least for user authorization checking. When we refer to a subject in the EoD we therefore usually describe only a "role" of that subject.

Contracts are also organized in specialization and aggregation hierarchies. This has the following advantages:

- Contracts can be modelled as complex objects. This allows both for complex contracts to be split up in smaller parts, and for reference in a contract to another contract.
- Contracts can be specialized and generalized. A contract is a specialization of another if it adds new authorizations and/or strengthens the pre- or postconditions. General contracts can be reused from one application to another.

However, as we have seen in section 5.2.3 specifying relationships between contracts is a complex matter both from a conceptual and formal point of view and is left for further research. Since this is not worked out in this thesis no modelling techniques for specializing or aggregating contracts are given.

7.4.1.1. AUTHORIZATION DIAGRAM

The authorization model of an organization is the specification of the authorized communicative behaviour of the actors. It specifies communication links between actors inside and outside the organization and whether or not obligations are created based on the authorization of the initiating actor. Because this model describes the authorizations and obligations we can also call this the *normative systems model.*

An authorization diagram can be given for both the context level and internal level. When describing the context level the internal structure and working of the organization is abstracted from. The first step is to identify the actors that play a role in the communication. For this a very simple diagram can be constructed where actors are represented by boxes and communication links between them as simple lines with arrows at both ends. For example, in the case of the travel agency:

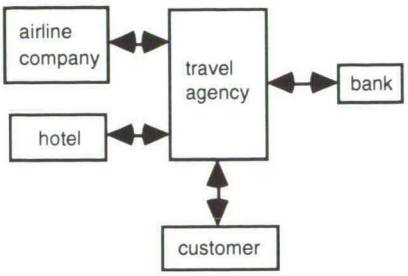


figure 7.7. Context level Authorization Diagram for travel agency.

The next step is to decompose the travel agency in organizational units or actors that play a role within the organization:

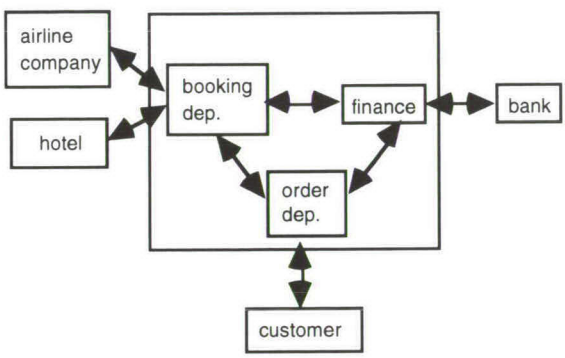


figure 7.8. Internal level Authorization Diagram for travel agency.

Each communication line is now worked out in more detail, describing the authorizations and obligations of the actors.

Several methods can be applied to refine the communication lines, such as DEMO or Action-Workflow (see section 7.5 for a discussion on these techniques in EoD modelling). However, we feel that the business logic framework as described in chapter 3 gives the best possibilities for identifying the transactions that create the authorizations and obligations between two communicating actors, especially those between actors from different organizations. In the business logic framework each communication line is seen as communication between a supplier and a customer of products (goods and services), consisting of a number of logical steps. The service provided by the supplier (and desired by the customer) can be as simple as providing a piece of information (e.g., a flight-schedule). Each communication line between the hotel and the booking department in the above example can therefore be described as a contract consisting of the four phases:

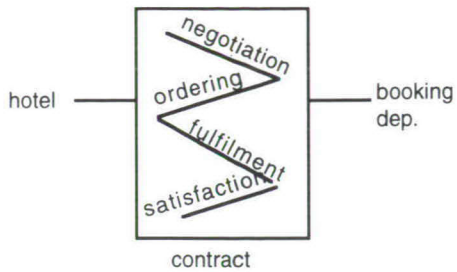


figure 7.9. Business logic contract between booking department and hotel

For each contract a detailed authorization diagram can be constructed specifying the authorizations and obligations creating transactions.

An example AD for the hotel-booking department is given in figure 7.10. The legend of the authorization diagram (AD) is given in figure 7.11.

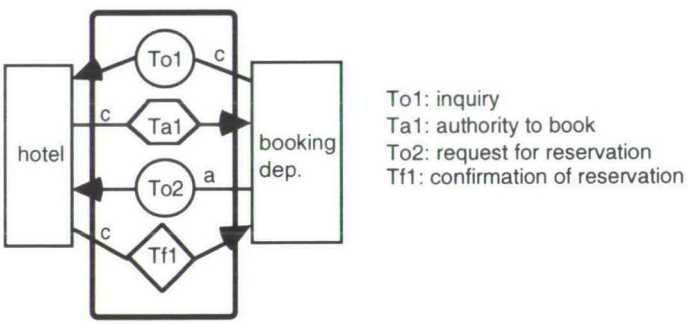


figure 7.10. Authorization Diagram of hotel reservation

The example shows the interaction between the booking department actor and the hotel actor. It starts with an initial request (based on charity) for booking information (To1), after which the hotel can give an offer (Ta1, authorizing the booking department to make a reservation under certain conditions). The booking department can then make an authorized reservation (To2), which is confirmed by the hotel (Tf1). Every interaction link from the initiating actor is labelled with the authorization claim (be it c (charity), a (authorization), p (power)).

In section 6.3.2.1 the authorization claims were described. Since they play such an important role in the modelling of authorizations and obligations the effects of these claims on the creation of obligations are repeated here:

- charity: the performance of a speech act (a message from actor A to B with an illocutionary point and some propositional content) is successful and an obligation for B is created with respect to the content, only if B accepts the point raised. E.g., if A requests the delivery of goods based on charity, only if B accepts this an

obligation for B to deliver the goods is created. Alternatively, B can refuse and the speech act fails. This also implies that a speech acts is successful **iff** the speaker is authorized to perform it according to the authorization model [Weigand, 1991b].

- power: if actor A has power over actor B (e.g. boss - secretary) then every speech act from A to B creates an obligation for B (independent of the approval of B). E.g., if the boss orders (requests on basis of power) the secretary to book a trip, an obligation for the secretary is created to do this.
- authority: in the performance of a speech act an obligation for B is created because B has agreed to it that A can make such a claim, i.e., B has granted A the authority to perform a speech act with a certain contents, creating an obligation for B to do so. E.g., if a hotel has granted the travel agency the authorization to direct the hotel to reserve a room, a request of the travel agency for a reservation creates an obligation for the hotel to do so. Refusal or non-performance of B results in a violation of the agreements and remedial actions should (and can) be taken.

In figure 7.10 only the communication success line between booking department and hotel is modelled. The next steps are to model the contract in more detail (e.g., the specification of what should happen if the communication fails) and to work out the transactions. Both are described in the next subsection.

The AD is inspired by Dietz' Communication diagram ([Dietz, 1992a]) and CSDs (Communication Structure Diagrams, from an earlier incarnation of DEMO [Dietz, 1990a]) for modelling the EoD. As in DEMO an organization is conceptualized as a system of mutual influencing actors. Interaction between two actors is said to take place if one of the actors is the sender (or initiator as Dietz calls it) and the other actor is the receiver of the transaction type (as defined in section 5.1) creating an obligation or a fact (an update of the receiver's belief) for the receiver, depending on the type, if the sender has the proper authority (including power over the receiver).

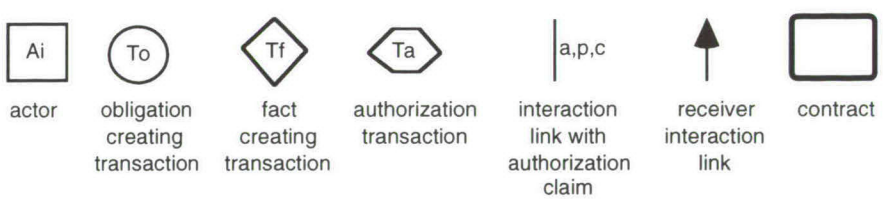


figure 7.11. Legend of the Authorization Diagram

In the AD an actor is represented by a box. The transaction types distinguished are: "obligation creating," represented by a circle; "fact creating", represented by a diamond; and "authorization", represented by a hexagon. Complex transactions can be combined by drawing the symbols behind each other. The actor initiating a transaction is connected to the transaction symbol by a sender link, represented by a plain line tagged with the authority claim of the sender. The receiving actor is connected to the transaction symbol by a receiver link, represented by a line with an arrow head pointing to the actor box.

7.4.1.2. COMMUNICATION DIAGRAMS

The communication model can be considered a refinement of the authorization model. Here the transactions are modelled in more detail, specifying all message types (speech acts) that make up the transactions, and it describes how these messages are exchanged between the actors in carrying through the transaction. It furthermore describes the communication contracts, specifying the obligations that are created in carrying through the transactions and (trans)actions of what should happen in case an actor violates one of the obligations that is created for it. The communication model corresponds most closely with Dietz' idea of essential communication. It describes the obligations and authorizations of the subjects with respect to the speech acts, and the conditions attached to them, such as they are agreed upon by the organizational subjects involved.

7.4.1.2.1. Contract diagram

As we learned from the business logic framework (section 3.1.2) some speech acts in the discourse belong logically together. E.g., the inquiry for a specific service, the ordering of it, and the delivery and payment. For this the concept of 'contract' was introduced in section 4.1. A *contract* is a set of related authorizations and obligations together with conditions on the relationship between the acts and rules governing the violation of permissions, prohibitions and obligations.

In modelling the communication transactions between two actors at this level we are only interested in the deontic effects of the transactions i.e. whether obligations are created or removed, whether or not some results are accomplished thereby fulfilling an obligation, and whether authorizations are granted or revoked. A contract therefore is specified as a set of deontic clauses. A deontic clause describes the status of an interaction of the two partners in terms of obligations, authorizations and accomplishments. The contract model (CM) is graphically represented by a Contract Diagram (CD) which is a Petri Net. The legend is given in figure 7.13.

Figure 7.12 shows an example in which the contract of figure 7.10 is represented in a CD (Petri Net) (as above, successline only):

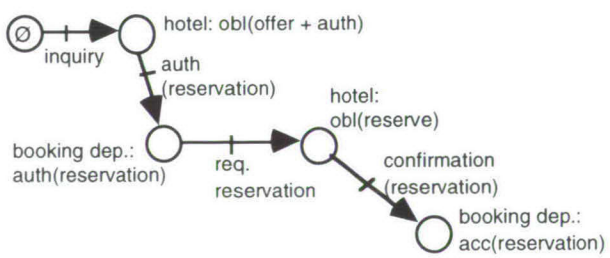


figure 7.12. booking contract diagram (success line only)

The example shows that after an inquiry an obligation for the hotel is created to make an offer and grant authorization (permission) to the requester to take on the offer. After the authorization transaction this permission is created for the booking department. After the request for reservation (using the authorization) the obligation to make the reservation is created for the hotel. After the confirmation of this the booking department has accomplished the reservation.

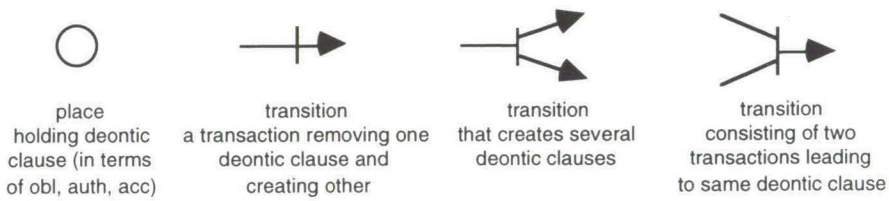


figure 7.13. Legend of the Contract Diagram

The CD describes the deontic clauses and their dynamics, i.e. a clause is created by one or more transactions, and is removed by other transactions. The status of the interaction can be represented by a *set* of clauses. A Petri Net consists of a set of places, a set of transitions, an input and an output function from the set of transactions to the set of places. In this case, states are identified with one deontic clause and the transitions are interoperable transactions that create or remove the deontic clause.

Petri Nets also allow for the specification of more complex situations, such as where one message creates several obligations (figure 7.14). E.g., by ordering an obligation for the supplier to deliver the service is created, and also the obligation for the customer to pay is created, depending on the agreements between the actors. We can also represent that only authorized transactions create obligations (figure 7.15). Only when both Obl.1 and auth. are present the Transaction is carried through successfully and Obl.2 is created.

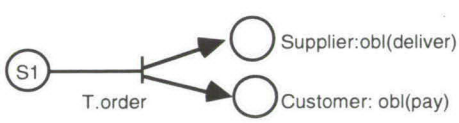


figure 7.14. Multiple obligation creation

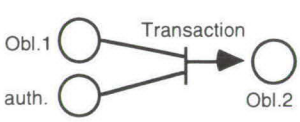


figure 7.15. Authorized obligation creation

Figure 7.16 shows a contract between the travel agency and the airline company corresponding to the flight-reservation contract specification example in section 5.2.4.

Here ø denotes the starting state. S1 denotes the obligation of the airline to fly raised by the flight reservation. S2 denotes the obligation of the travel agency to pay, also raised by the flight reservation. S3 is the goal state for the travel agency and S4 the goal state for the airline company. If both are reached (if both travel agency and airline are satisfied) the contract ends in state ‡. In case the success line is left (e.g. by a cancel transaction) it can be specified how to return to it (in the example by the obligation of the airline to pay a fine, and the obligation to pay back the ticket if the travel agency already paid).

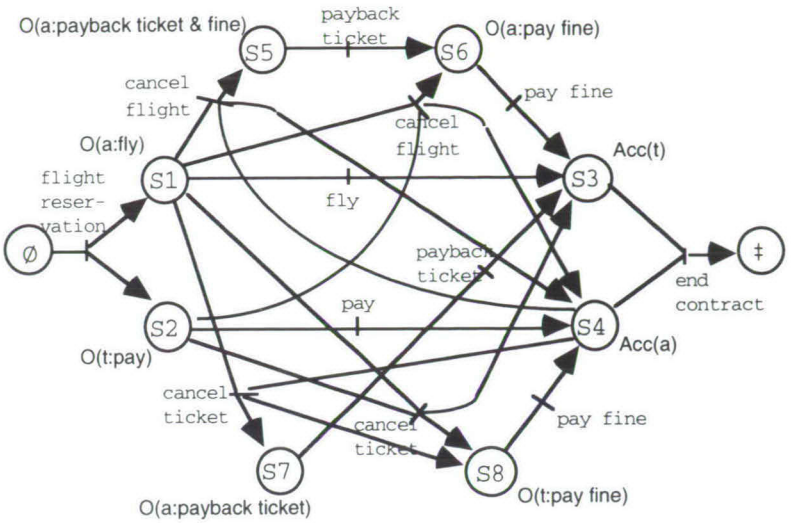


figure 7.16. Contract dynamics of a flight-reservation

Although the coLa specification of the contract as given in section 5.2.4 does not look complicated, it can be concluded from the diagram that specifying satisfaction of both parties and all possible exceptions and their deontic effects is a complicated task.

Figure 7.16 does not include alternative flight offers by the airline company after a flight is cancelled. The inclusion of this would complicate the diagram even further, since it involves the making of choices. For instance, whether the ticket should be paid back and a new obligation for the travel agency to pay is raised after an alternative flight is offered depends on whether the travel agency has already paid or not. In any case the transition to state S4 (acc(airline)) from the cancel-flight transaction no longer holds and also the payment of a fine (pay fine transaction from S6) alone does not lead to an acceptance state of the travel agency (S3).

The modelling of choices is a complex matter in standard Petri Nets. A better option would be to use an extended form of Petri Nets, e.g. Coloured Petri Nets ([Jensen, 1987]).

7.4.1.2.2. Transaction diagram

The transaction model describes communicative behaviour in the EoD in terms of speech acts exchanged between interacting subjects and relationships between these speech acts.

There are several possibilities for modelling transactions. One alternative is the construction of a dependency graph as described in [Ngu et al., 1994], as mentioned in section 5.1.4. The advantage is that hereby also the (temporal) constraints specification can be verified. In [Verharen et al., 1994] we described how extended CSDs ([Dietz, 1990a]) can be used for modelling transactions. A third possibility is to use action diagrams (remember that speech acts are actions), like SIMM diagrams of BAT ([Goldkuhl, 1996]), or action diagrams from DEMO ([van Reijswoud, 1996]). However, these methods lack

concepts, such as the temporal constraints, and deadline specification, that are part of the transaction specification as defined in section 5.1.4. Therefore, a technique is chosen that follows this specification closely. Of course, again one can decide not to model all details in the initial stages.

The Transaction Model is graphically represented by a transaction diagram (TrD). The TrD is inspired by the sequence diagram of the Communication Model from Commodious ([Holm, 1996]). Figure 7.18 gives the legend of TrDs. Figure 7.17 gives an example of the transaction diagram for the hotel reservation (for the travel agency), corresponding with (but expanding) the transaction specification in section 5.1.4.1.

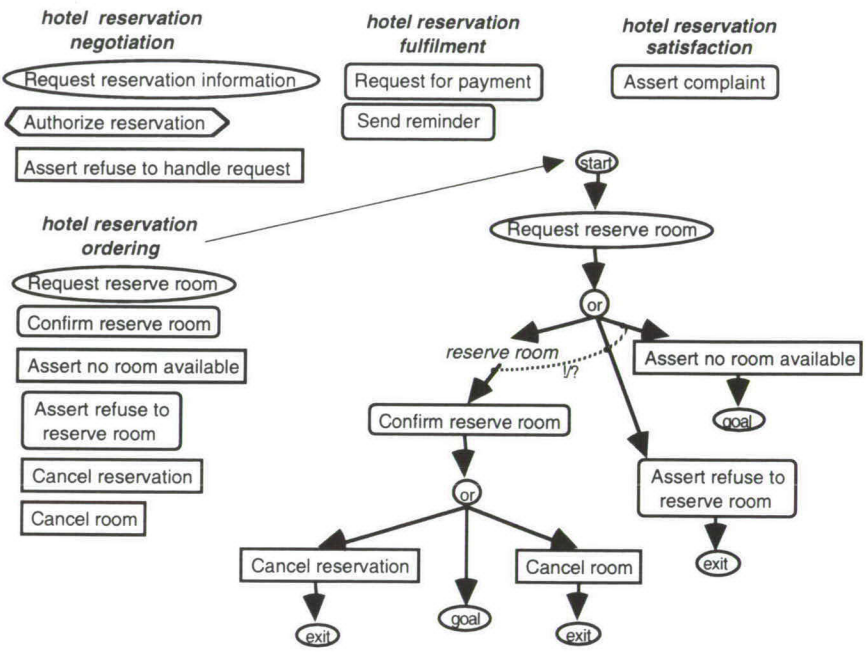


figure 7.17. Example TD for hotel reservation

Input for the transaction modelling phase are the AD, CD and business logic contract. The first step is to decompose the discourse for every phase in the business logic contract, i.e. for every phase one gives the speech acts that make up the transactions that are identified in the AD. In the example the possible speech acts are given for negotiation (request, authorization, and refusal), fulfilment (request for payment, reminder (another request for payment), the private action payment), satisfaction (complaint) and of course the ordering phase. The ordering consists of the speech acts for requesting a reservation, the confirmation or refusal of this, and the private action reserve room. Although not really a part of the transaction (as specified in section 5.1.1) for failure handling we can also include possible cancel messages.

The second step is to construct a sequence diagram in which the speech acts are connected leading to goal or exit states. For the ordering phase such a sequence is represented, starting with the request for reservation. Following this the hotel can assert that no room is available (leading to a goal state of the transaction) or the reservation of a room, that is confirmed to the requesting party. This also ends the transaction in a goal state. Alternatively the hotel can refuse to make a reservation leading to an exit (non-satisfactory) state. The same can happen when the reservation is cancelled (either by the requesting party or the hotel itself). Reaching an exit state usually triggers remedial or sanctional transactions (according to the contract).

In the sequence diagram one also can give temporal and deadline constraints on the connection between speech acts. E.g., in figure 7.17 the temporal constraint is included that after a request to reserve a room always sometime in the future an answer must follow which can be one of the assertion no room is available, the refusal to handle the request or the confirmation of the reservation.

For every speech act its type and authority claim are given. The basic types of messages exchanged between agents in the EoD are based on the illocutionary primitives as described in section 6.3.1 en 6.3.2. A distinction is made between directive, assertive, commissive, declarative, authorization (authorize/retract), permission (permit/forbid) and confirming acts. These are represented by the name of the type, e.g. ***request*** reserve room. Also messages to the actor itself to perform some primitive action or to update its knowledge and beliefs can be included. These are represented as just the action name, or as an assertive message to itself, see section 5.1.4. The authority claim is expressed by the kind of box holding the speech act.

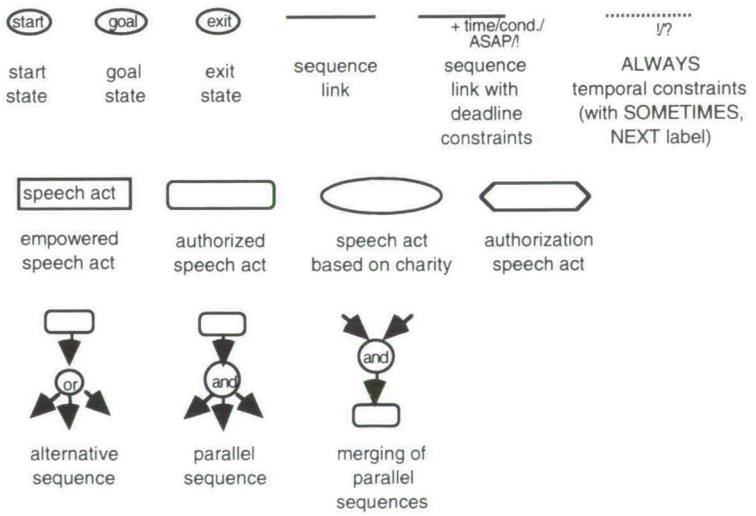


figure 7.18. Legend of the Transaction Diagram

7.4.2. TASK MODELLING

Besides their communicative behaviour, modelled in the communication model, also the tasks of the actors in the organization should be specified. This is especially important if (some of) these tasks are delegated to an automated system. This subsection describes how the tasks of an agent can be represented.

The task model is graphically represented by the Task Diagram (TD). Figure 7.19 gives the TD for the travel agency, corresponding with the task specification examples from section 5.3.1 (and 5.3.1.1). Figure 7.20 gives the legend of the TD.

A task may consist of a number of subtasks, and each subtask is itself a task that again can consist of other subtasks (section 5.3). The elementary subtasks are either transactions (described in the TrD) or private actions (specified in the UoD model, see below) represented by their italicized name, or **skip** in case of a non-vital subtask.

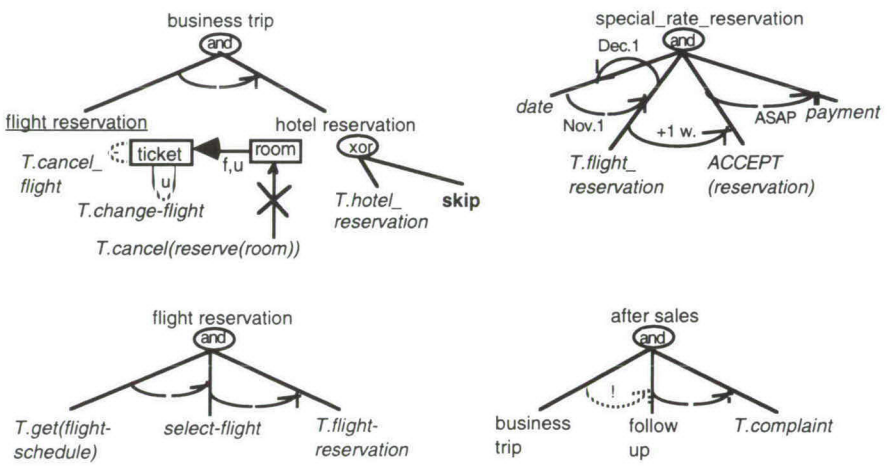


figure 7.19. Travel agency TD

Tasks and subtasks are linked to each other by simple lines, possibly labelled with the preference given to the subtask. The goal of the task is the performance of some set of subtasks. This can be specified by a special node just below the task name labelled AND or XOR (denoting required task/subtask relations and alternative sets respectively). In order not to complicate the graph of one task the subtasks can be modelled with their own graphs. If a subtask is modelled in more detail elsewhere the subtaskname is underlined, e.g., flight reservation in the example. Furthermore, constraints between the subtasks can be specified. These are temporal constraints (including deadline specifications), represented by directed arrows between the links connecting the subtasks to the task, e.g., between flight reservation and hotel reservation, specifying that flight reservation should be performed BEFORE hotel reservation. In case of deadlines these links can be labelled with the time specification or a proposition to be tested, e.g., in the special_rate_reservation task the payment should follow as soon as possible (ASAP)

after ACCEPT(reservation). Complex temporal constraints (such as the ALWAYS, SOMETIMES and NEXT constructs described in section 5.3.1) are represented by a dotted link denoting the ALWAYS constraints, labelled with an '!' denoting ALWAYS(x -> NEXT y) or a '?' denoting ALWAYS(x -> SOMETIMES y), e.g., the ALWAYS constraint between business trip and follow up in the after sales task.

A subtask can be tagged with a result. There are several relationships defined between results (section 5.3.1):

- a *dependency*, represented by an arrow to the result it depends on, labelled with the kind of dependency, be it c (create), f (fail), u (update). E.g., in figure 7.19, the fail and update link between the room (result of the hotel reservation) and the ticket (result of the flight reservation subtask) meaning that if the ticket reservation fails or is updated so must the room (hotel reservation). To not clutter the diagram with long links between dispersed results this can also be represented by a short arrow connecting the result with the name of the depending result.
- an *update* relationship, represented by an arrow from the result to itself, labelled with the transaction that changes the result. For example, the change-flight transaction updating the result (ticket) of the flight reservation.
- an *invalidation* relationship, represented by an arrow from the result to itself, labelled with a '–' and the invalidating transaction. For example, the cancel_flight transaction for the ticket.
- a *revalidation* relationship, represented by an arrow from the result to itself, labelled with a '+' and the transaction that revalidates the result after it has become invalidated.
- a *compensating* relationship, represented by a special arrow to the result connecting the transaction that compensates the result. For example, the cancel(reserve(room)) transaction for the room.

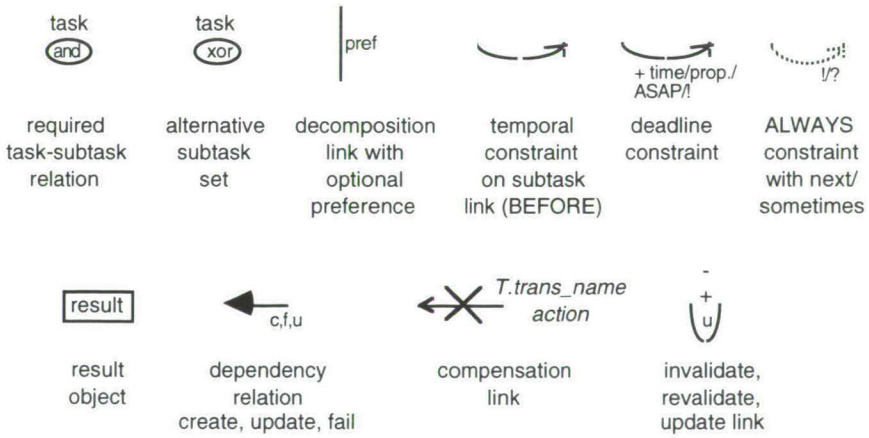


figure 7.20. Legend of the Task Diagram

Again the question is to what level of detail the concepts should be worked out. In case of initially analyzing the tasks of an actor one can suffice with only a description of the subtasks and simple temporal constraints between them. If the model serves as input for the formal specification phase the models can be worked out in more detail since the graphical technique contains representations of all elements of the TaLa specification language (section 5.3.1). A future goal is to build a tool that automates the construction of the formal specification from the TD.

7.4.3. UoD MODELLING

UoD modelling concerns the modelling of the content of the messages exchanged between the actors, and the modelling of the private actions of the actor, to be implemented by an automated system. UoD modelling therefore is more than just information modelling as done in other methods. For both objectives the same modelling technique can be used, because the operations will work on objects recorded in the database(s) of the IS that represent what is communicated about in the discourse.

Depending on the development phase, the result of the UoD modelling is either the *UoD model* (analysis phase) or the *database schema* (design phase). The purpose of the UoD model is to provide an accurate formal description of the UoD, the database schema is the model of the object base. The input however will be different. For the UoD model the input typically consists of the *contents* of the communications modelled in the EoD. The input for the database schema comes from the UoD model.

In section 7.2 reasons were given for the choice of NORM as UoD modelling technique. The next subsection describes NORM in more detail.

7.4.3.1. NORM

NORM (Natural Object Role Model, [De Troyer, 1991]) is an OO modelling technique with a formal semantics ([De Troyer and Meersman, 1995]). The NORM model combines the powerful modelling principles of NIAM (lexical vs. non-lexical objects, is-a subtyping, binary fact types, constraint types and a graphical representation) ([Verheijen and van Bekkum, 1982]) with the principles of OO (behaviour specification, encapsulation, specialization of inherited properties, definition of ADTs, and type constructors).

NORM supports the following concepts:

1) *Object Types* are used to classify objects and to describe their properties, as well as its own properties, the type properties. The structural properties are described by means of *attributes* the behavioural properties by means of *methods*. *Constraints* can be specified to restrict the use of the properties. All attributes are *encapsulated*, this means that they are not visible outside the object type definition.
   a) a distinction is made between *local* and *global* object types. Local object types are defined within the scope of the definition of another object type, called *focus* object type. They are completely encapsulated in the focus, i.e., they can only be used within the scope of the focus and are not visible outside this scope, also their instances are not visible outside the scope. Instances of a local object type

cannot exist without being related with at least one instance of the focus object type. Local object type instances may be shared between focus instances.

Global object types have their own definition schema. In the focus schema only the relationship with a global object type is encapsulated, not the global object type itself. Instances of a global object type are independent of the instances of the focus and can only be created and manipulated by methods of its own type.

The distinction between local and global object types is made to reduce the number of object types and to increase reusability. Only independent meaningful object types should be defined as global object types. Reuse of object types will be easier because all dependent object types are already encapsulated in the object type definition and need not be gathered. In addition, the use of local object types allows to express a much richer structure than traditional attributes.

b) There is also made a distinction between *lexical* and *non-lexical* object types. A lexical object has a lexical representation which is one-to-one with the objects identifier. No two different lexical objects can have the same lexical representation, therefore this lexical representation can be used to identify the lexical object. Examples of lexical object types are INTEGER, CUSTOMER-NAME, examples of non-lexical object types are CUSTOMER, AIRLINE.

As with agents and contracts in the EoD model, all object types are organised into a single subtype hierarchy. The strict IS-A meaning for subtypes is used, meaning that each instance of the subtype is also an instance of the supertype. The ultimate object type is the (pre-defined) object type OBJECTS. It describes all possible objects that will occur in the UoD at their most general level. The object type OBJECTS has two exclusive and total subtypes: NON-LEXICAL-OBJECTS and LEXICAL-OBJECTS which represent the distinction between non-lexical and lexical objects. All other object types are either subtype of NON-LEXICAL-OBJECTS or LEXICAL-OBJECTS.

Exclusion and totality constraints can be specified between subtypes. All properties of the supertype are *inherited* by the subtype. *Overriding* of properties is allowed but only to make the property more specific. The conceptual meaning of the property should be retained. *Multiple inheritance* is supported.

The subtype hierarchy can be extended with user-defined new abstract data that become "first class citizens" as it is called in object-oriented terminology, meaning they are treated as any predefined object type. For modelling complex object types type constructors are provided.

2) *Explicit relationships* are relationships which are not encapsulated in some object type. They are used if there is no *conceptual* reason for considering one object type as attribute of the other object type or vice a versa.

By using explicit relationships we want to avoid the (in)famous ravioli problem and increase reusability. Object types can be modelled as general building blocks or 'general purpose' object types, that only contain the information that is proper to the object type. This has two major advantages:

a) Important UoD relationships between object types are still visible at a global level, e.g. (see figure 7.21) the "issuing, issued by" relationship between

AIRLINE and TRAVEL CONDITIONS in the business trip example is not hidden in either AIRLINE or TRAVEL CONDITIONS or even in both.

b) The general purpose object type can more easily be reused in other UoD environments because they do not contain UoD specific information, e.g. the object type AIRLINE as well as the object type TRAVEL CONDITIONS can be reused in other environments.

3) *Behaviour* is expressed by means of two concepts. The behaviour of the objects of an object type and services to other objects are described by traditional *methods* (including type methods). Methods can be *private* or *public*. Private methods cannot be used outside the object type definition, while public can. Causal relationships between objects are modelled using *triggers*. Triggers are usually, but need not be, connected with explicit relationships. Also methods may be defined for explicit relationships.

The model has an associated graphical representation for the definition of object types, attributes, explicit relationships and a large range of constraints, such as equality, exclusion and subset constraints. Methods also have a graphical representation but there definition is textual. Complex constraints must also be expressed textually.

An example NORM-schema giving explicit relationships for the business trip is given in figure 7.21. Figure 7.22 gives a NORM-schema for ticket.

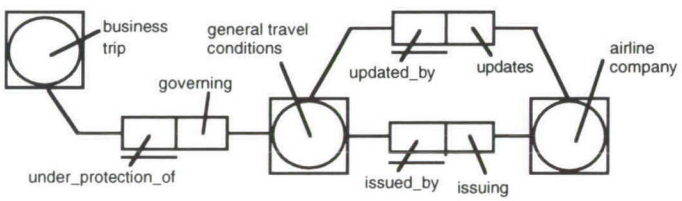


figure 7.21. Example NORM-schema with explicit relationships

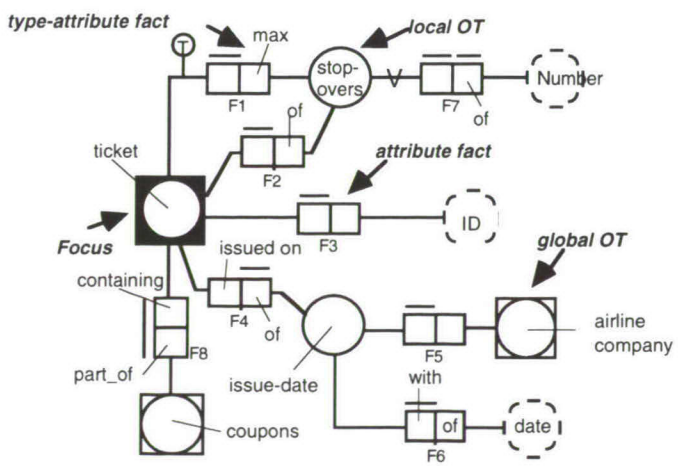


figure 7.22. Example NORM schema for ticket

NORM does not have a graphical representation technique for modelling the operations. Methods are expressed by means of pre- and post-conditions. Optionally a body may be given to specify in a more procedural way how the post-conditions should be realised starting from the pre-conditions. To specify the body of a method an object-oriented language is foreseen including control structures like assignment, sequential order, order unspecified, if-then-else instructions, repeat instructions:

LET - BE -	assignment
;	sequential order
//	order unspecified
IF - THEN - ELSE	choice
REPEAT; - FOR	repeat sequential order
REPEAT// - FOR	repeat order unspecified

Also techniques from object-oriented design can be used to specify the methods. Several primitive methods are pre-defined for every object type. Examples are NEW (to create an object), ADD (to add an attribute value or explicit relationship to an object), IS (to test on the existence of an attribute link for the object), etc. Also pre-defined methods are provided for Type Constructors. Pre-defined methods can be used in the specification of user-defined methods.

The language for the triggers is event driven. Figure 7.23 gives the grammar of the trigger language:

```
(ON | AFTER | BEFORE) <events> DO <messages>

<events> ::= <event>
        ( AND <events> |
        ; <events> |
        // <events> |
        OR <events> )
       | '(' <events> ')'

<event> ::= <message> (AND | WHERE <cond>)
```

figure 7.23. Grammar of trigger language

7.5. COMPARATIVE ANALYSIS OF BUSINESS PROCESS MODELS

With the increasing interest in the Language-Action Perspective a number of methodologies has emerged that take this perspective to modelling business processes and that are aimed at the support of business communication. The most well-known are Action Workflow, DEMO and BAT, already mentioned in chapter 2. For a comparison between the methods see for instance [Goldkuhl, 1996] and [Dietz et al., 1996].

An important role of BAT, DEMO and Action Workflow is that they give structure to certain communication situations. BAT best describes a market situation with a supplier and a customer of services or goods. DEMO, starting from a different transaction concept, describes in more detail the business processes within organizations, and especially the

TPM (transaction process model) describes the process structure of communicative transactions based on the theory of Habermas. Action Workflow on the other hand concentrates on workflow situations and tries to improve on them.

Although we share many ideas and concepts with these methodologies there are some differences, the most notably being the formal foundation which is hardly addressed in the other approaches while it takes a prominent place in the approach presented here. Also the modelling techniques differ. One major difference is the role that authorizations play in the CoLa methodology. The other approaches do not explicate authorization and start mostly from the assumption that the communications are authorized. Another difference is that in the approach presented here in the design phase the focus is shifted to one of the actors for which a supporting system is to be implemented. That is why the CoLa methodology incorporates a Task Model that is missing from the other models which mainly focus on modelling the domain.

This does not mean however that the other approaches are not valuable. Far from it. As the reader probably has noticed many of the models described in the previous section are inspired by models of the other methodologies. To be able to serve as input for the specification of software systems we need a close connection between the models and the underlying formal framework. I have therefore adapted some of the existing models in such a way that they describe all aspects relevant to business communication modelling. If this could not be done I proposed new ones, such as the task model. But, especially in analyzing business domains, one can substitute any of the other models for the ones given here, however I feel this should be done taking into account the situation to describe.

Below some more detailed remarks are made concerning the application of the different methods and models within or differences with the CoLa methodology.

7.5.1. DEMO MODELS

DEMO focuses on the modelling of conversational moves in business communication. It is hypothesized that coherence in business communication is to be found in a restricted number of (initiating and reactive) conversational moves. DEMO specifies the essential model of an organization that is an integrated whole of the partial models: Communication Model, Process Model, Facts Model, and Action Model. The models have an associated graphical representation [Dietz et al., 1996].

The DEMO Communication Model ([van der Rijst and Dietz, 1993], [van Reijswoud, 1996]) contains the identified transaction types and the actors that are involved as initiator and executor of the transaction. It has inspired our Authorization Model. If one adopts the DEMO transaction paradigm (described in section 2.1.3.2) then the DEMO Communication Model can serve as the initial AM. However, one loses the possibility to model the authority relations, the explicit creation of obligations and the explicit representation of granting authorization, which in my view are important aspects of modelling business communication.

The Process Model is graphically described by Process Diagrams. It depicts the relationship in time between the transaction types as identified in the Communication Model. The Process Diagram can be compared to the Transaction Diagram, however, it does not model speech acts but only phases of the DEMO transactions.

The Facts Model is, like the UoD Model presented here, based on NIAM, describing the object world, the objects the actors communicate about. However, it only provides a purely statical description of the object structures. Our UoD Model also describes the dynamics of the objects in the domain and the system behaviour aimed at. Both can be used for modelling the speech act contents.

The Action Model is the specification of the behaviour rules the actors have to follow in performing essential actions. The Action Diagram is a flowchart describing per actor per transaction phase the essential actions, i.e. the conduct of actagenic and factagenic conversations and execution of objective actions. It is similar to the Task Model in that it specifies the order of the communicative and private actions. However no deadlines, other temporal relations between the subtasks, results and dependency relationships between them can be given.

Van Reijswoud recently extended DEMO with a Transaction Process Model ([van Reijswoud, 1996]). The TPM is a business conversation process model presenting the possible communication moves in a business conversation process. It consists of two partial models, one representing the conversation structure of communicative action and the other representing strategic action. The communicative oriented TPM is divided into three layers: the success layer, locating the successful transaction process resulting in the creation of a fact; the discussion-and-failure layer, formed by the discussion about validity claims and unsuccessful transaction processes; and the discourse layer, locating the discourse with the purpose to restore the background conditions for the conversation.

Besides the difference in transaction concept (see below), in the CoLa methodology the transaction process specification is divided between Transaction Model and Contract Model. The Transaction Model models all speech acts from the actors and the Contract Model describes the communication in terms of obligations, authorizations and accomplishments and also outlines the transactions that cause a diversion from the success line (and the transactions how to come back to the success line) together with the (deontic) effects of leaving the success line.

The basis for most differences lies in the different interpretations of the transaction concept. The DEMO transaction concept defines a transaction as a whole that consists of three phases: the inception phase, consisting of an actagenic conversation; the action phase, consisting of the execution of the objective action; and the conclusion phase, consisting of a factagenic conversation. Every established fact, that is recorded in a fact bank, is the result of the successful carrying through of such a transaction. As described in section 5.1 the CoLa transactions are less restrictive in their format than the DEMO transactions, allowing for more freedom in the specification of different kinds of transactions. In DEMO a transaction almost exclusively starts with a *request* of the performance of an action, whereas in the CoLa methodology the transactions can be of many different kinds. Although Dietz describes that some phases of the DEMO transaction can be implicit (if the transaction does not concern a request), thereby

allowing more freedom of the transaction types specified, in my eyes this only complicates the modelling of transactions.

A second difference is that the creation of an agendum, as Dietz calls it, is buried inside the transaction and not modelled explicitly. Again this is obvious if the actagenic conversation concerns an authorized request. However, it is not for other transaction types, and precisely the creations of obligations is one of the most important aspects to be clear about when modelling business communication for the coordination of activities.

A third difference lies in the inclusion of the execution of the objective action within the DEMO transaction. Where the action is the content of a speech act that is part of the transaction (in DEMO, part of the actagenic conversation), in my view the *execution* of the action that is requested or ordered is the concern of the receiving actor only and is not part of the *communicative* behaviour of the actors. For the initiating actor the creation of an obligation for the receiver to perform the action is the most important. The result of the action can take several forms and is not always the creation of a fact that has to be communicated to the initiating actor. This is especially true in an agent setting if the agent has reactive components that can sense a change in its environment (e.g. the delivery of goods). The action itself is described in the Task model (and in case it is a primitive action on one of the agent's knowledge bases it is described in more detail in the UoD model).

The last difference between the two models is the modelling of authorization transactions. In the CoLa methodology these are modelled as special kind of transactions because of the importance that is attached to making the authorization links explicit. In DEMO authorizations are modelled just like other transaction types.

One can argue that DEMO transactions are higher level than CoLa transactions, however the goal is to explicate the authorized communicative behaviour leading to obligations for the communicating actors.

7.5.2. BAT-SIMMs

For modelling purpose according to the general BAT framework ([Goldkuhl, 1995, 1996]) Goldkuhl extended Action Diagrams from the SIMM method for business modelling ([Goldkuhl, 1992]).

SIMM action diagrams have material and information flow orientation only, there is no distinction made in information flow between the contents part (proposition) and the action (performative) part. Goldkuhl admits that to be used within a communicative action framework it is important to be explicit about the (communicative) action character, and therefore has added such descriptional possibilities. In contrast with my approach, he does not use an established classification scheme for action characterization, such as Searle's illocutionary classes. In Goldkuhl's view such an approach is too restricted in semantic expressiveness and is highly dependent on the maturity level of the theoretical basis. In his opinion there is still a good deal of theoretical controversy about communicative action classes to be solved, to use them as good characterization. Instead of a pre-defined approach, the modeller can choose an appropriate (communicative) action denotation.

Goldkuhl admits that there is not a close link between the BAT and Action Diagrams. The BAT framework therefore is not enforced and this is a disadvantage. But the

communicative action framework is important to rely on as a guide and source for inspiration. The BAT framework and its underlying communicative action theory should be used as a basis for this kind of action modelling.

From the approach presented here it should be clear I think along the same line as Goldkuhl. For me the Business as Action game Theory is the most appealing framework to work in. However SIMMs have severe shortcomings in modelling the transactions that make up business communication. Furthermore, in my view, speech act theory and illocutionary logic *can* be used for the design of automated systems supporting the communication and coordination in organizations, under the conditions specified in chapter 2 and section 6.3.1.

7.5.3. ACTION WORKFLOW

Action Workflow is a methodology that incorporates a theory (generic business framework), an analysis and modelling method and a supporting software tool.

The Action Workflow theory can be seen as a generic blueprint for the organization of work. It describes business processes as a loop consisting of four phases (preparation, negotiation, performance and acceptance), between two actors that play the role of customer and performer. However, as [Goldkuhl, 1996] describes, the emphasis is on the customer. As in DEMO it starts with a customer request and the performer's commitments. Work ends up with customer satisfaction. The focus is on communicative actions only, especially commitments, no material actions can be specified. The only model made is the workflow loop where each phase can in itself consist of a new loop.

This method can be compared to the BAT theory, but as Goldkuhl explains in [Goldkuhl, 1996] BAT is more general in its applications to business communication, while Action Workflow is better in modelling workflows. A comparison with models from the CoLa methodology is hard to make. However, it is possible to use the CoLa models in the specification of the different transactions. Instead of the phases identified by the business logic the Action Workflow loop with its phases can be used as a generic blueprint that can help in identifying the transactions.

7.5.4. COMMODIOUS

[Holm, 1996] describes the Commodious (COMmunication MODdeling as an aid to Illustrate the Organizational Use of Software) Method. The method describes how three interrelated models for organizational communication can be developed that can be used as a basis when formulating requirements on a software system. The models are the communication model, system behaviour model and information model (database schema).

The CoLa methodology corresponds closely to the Commodious method. The Transaction Model is inspired by the communication model, describing speech acts and their sequence, only extending this model by the specification of temporal constraints (including deadlines) and the authorization claims on the speech acts. Holm also makes a

distinction between discourses and every discourse can be described by a contracting and an ordering part, similar to the business logic division.

The information model (constituting the database schema) is an ER variant and describes the static structure of the object world (the contents of the speech acts). The dynamics are described in the system behaviour model by specifying registration events and supporting functions. This is similar to the action specification part of the UoD model, and in my view the two can be exchanged. The only difference between the UoD model and the information model is the perspective taken, our UoD model is first used to model the objects in the domain, whether or not they will be part of the software system, and for database schema specification the same model can be used. Holm's information model is directly aimed at specifying how the contents of the speech acts can be stored in the database.

In contrast with the CoLa methodology the Commodious method is supported by an automated tool, that can also be used to give advice on the completeness of the models. This is a future goal for this research.

7.5.5. ELECTRONIC CONTRACTING

None of the methods has a model similar to the Contracts model. This is partly because of the emphasis put here on the authorizations and obligations, following the view of a system as a normative system. However, we are not alone in thinking that Petri Nets make a good modelling mechanism for specifying obligations. In his research on systems for supporting electronic contracting, Lee also uses Petri Nets for specifying contracts ([Lee, 1988a, 1996]). At this moment it is not clear however (to me) how the deontic clauses that make up the places in the Petri Net are specified. In [Lee, 1988a,b], [Ryu and Lee, 1992] also a variant of deontic logic is used as formal framework for specifying contracts, as was described in chapter 6. Furthermore, in Lee's approach the emphasis is on the logical modelling of contracts and the contracting process and not on the business communication (but of which contracts are an important aspect, of course).

CHAPTER 8
EPILOGUE

The research reported on focuses on a new way of designing a class of automated information systems, called Cooperative Information Systems (CIS).

CISs are systems that are used within and between organizations to support cooperation and coordination of organizational activities. Both in organizational theory and computer science it is recognized that this is achieved by communication. Traditional IS development methods do not pay much attention to the design aspects of communication and coordination. When viewing ISs as autonomously communication systems supporting coordination these methods fall short. Therefore, a new perspective is needed that focuses on the use of communication as a coordination mechanism and that can be used to obtain a better understanding of the structure of business communication.

In this dissertation I took a Language-Action Perspective (LAP) on the design of CISs. The LAP describes what actions people and systems *do* while communicating, and what it means to commit to the conduct of some activity. It considers an organization as a network of inter-*acting* agents that create, maintain, and terminate commitments, and aims at understanding the communication structures within and between organizations.

This final chapter summarizes the results and conclusions from this investigation and also gives an agenda for future research.

8.1. RESULTS AND CONCLUSIONS

The main goal of this research (see section 1.2) was to investigate the usefulness of taking a LAP to the design of CISs. In doing so four derived goals were identified:

1. structuring mechanisms for business communication, based on the LAP.
2. the logical formalization of this communication framework.
3. the conceptual modelling techniques and a specification language based on this framework, used for analysis and design of CISs.
4. the use of an (intelligent) agent architecture based on the communication framework for the development of CIS.

Outlined below are the findings related to these goals.

1. Structuring mechanisms for interaction

Interaction between communicating parties is described on two levels: the business logic giving the overall context of the communication, and a communication framework describing the communicative actions of the parties in detail.

a) Business logic framework

For describing business oriented communication a framework is proposed based on Goldkuhl's Business as Action game Theory. In my view the framework not only is applicable in business settings between organizations, but in every situation where one agent (system or person) needs a service from another agent, even if they belong to the same organization or organizational unit, i.e. in every communication situation a supplier and customer role can be identified. In a business process context communicative acts cannot be viewed as isolated entities. They are related to each other, each getting their meaning from the business context: the roles and relations of the parties and the other business actions and the total *action logic* of the business transaction. A business transaction therefore is seen not only as a language game, but as an *action game*, hence the applicability of the Language As Action Perspective.

The framework structures business communication by identifying four phases: proposal and negotiation (setting up conditions), commitment or contracting (coming to a mutual agreement), fulfilment (performance of actions), and completion (or satisfaction).

In contrast to Goldkuhl's framework a contract, as defined here, not only describes the commitment phase, but also (and especially) the succeeding phases: fulfilment and satisfaction. This is in accordance with real-life contracts that describe the role of both supplier and customer regarding the business transaction. Important is that this provides a model of *symmetrical* communication as action between supplier and customer.

For business transaction specification the speech act theory of Searle and the communicative action theory of Habermas are used. Speech act theory provides basic speech act types and conditions for their successful use, but is rather one-sided in its speaker-orientation, and falls short in validation claims for the speech acts. Habermas provides viewpoints on these issues. In the framework therefore relationships between the communicating partners are specified that influence both the type of contract that can be negotiated between them and the effect of the different speech acts performed.

b) Communication framework

Interaction between (autonomous) systems is described by interoperable transactions. The specification of transactions that span many systems and have a long-life duration should be flexible. It is argued that traditional transaction models do not suffice since they are too much centred around central databases, and the standard ACID properties no longer apply. This also means that in interoperable transaction specification special attention should be paid to failure management.

In this thesis a three-level communication framework consisting of (interoperable) transactions, contracts, and tasks is proposed. The main contribution of this framework is the separate specification of failure management, the scheduling and execution management, and communication management of interoperable transactions. This provides a structured approach to specifying interoperable systems.

The framework provides for flexible failure handling. Failures include violations, where one of the agents does not comply to agreements, and cancellations, where one of the agents undoes a previously made commitment. Contracts manage failures due to violations, and tasks manage failures due to cancellation. Also the complex concept of 'compensation' is sorted out into two more focused notions: compensation of the other party, as specified in the contract, and compensation and contingency handling of the task. As result of the separation, compensation of a transaction can be viewed purely from the perspective of achieving the goal. The compensation requirement is relaxed compared to approaches in which the deontic consequences such as liability to claims are part of the compensation procedure. Although exception handling is a complex issue and will remain so, the least we can do is try to manage the complexity. This is the most important goal of the task/contract distinction.

Another advantage of the separation is that task management can be turned into a local issue, rather than a concern of the (global) multidatabase. In this way the global control is kept to a minimum, which makes specification and implementation more flexible. A last advantage is that it is now possible to make use of the most appropriate techniques, such as deontic reasoning in contract management and goal-seeking in task management.

2. The logical formalization of the communication framework

Because messages have effects on the receiver(s), and the sender must be able to reason about these effects, it is important that the messages and their effects are formally described. For this, the notions of obligations and authorizations are made explicit. Also three perspectives describing the relationship between communicating agents are distinguished that influence the effect of the messages: power, authorization, charity. As in human social systems, these perspectives can complement each other in the organization of interoperable computer systems.

For the semantics of communication models *Dynamic Deontic Logic* is used. It provides dynamic concepts for dealing with (communicative) actions and transactions. The fundamental reason for using deontic concepts is that coordination of behaviour always requires some form of mutual commitment. If an agent does not execute an action it committed itself to, it causes a violation of the contract. Violations do not cause logical inconsistency, but can be the trigger for sanctional or repair actions.

Although deontic logic has been applied in the field of ISs before, the dynamics of normative systems have received little attention. In this thesis the way deontic statements are created and adapted in communication processes is explored, and the role they play in the regulation of communication itself. It is shown how authorizations for specific acts can be requested, granted and also retracted, thereby creating a dynamic environment for the establishment and derogation of authorized norms. Also, both the negotiation phase to establish the contract as well as the contract itself can be modelled using this formalism.

Dynamic deontic logic is combined with *Illocutionary logic* to obtain a full logical framework in which the communicative behaviour of CISs can be specified. To describe the semantics of communication processes, the meaning of the information and communication aspects of the messages need to be specified. Illocutionary logic is a logical formalisation of the speech act theory and is used to formally describe the message structure, i.e., the types and effects of the messages.

The combination of dynamic deontic and illocutionary logic gives a formal framework and integrated semantics that provides for the precise description of the concepts used. With the logic formulas the exact effects of each message in a protocol can be inferred. It is also possible to calculate the effects of the complete communication protocol. This makes it easier to react to breakdowns in the communication. Furthermore it can serve as a basis for the development of a formal specification technique.

3. Conceptual modelling techniques

Based on the communication framework, and supported by the logic, the specification language CoLa (Communication and Coordination Language) for specifying transactions, contracts and tasks is developed. The language is complemented with graphical models, that can be used in the analysis and design stages of the development of a CIS.

The need for flexible ISs that support communication and coordination in organizations led to the proposed modelling methodology. It does not give complete cookbook methods with definite steps on how to find all objects and subjects, and build the system; the aim is to produce an abstract model of organizational communication as basis for formulating requirements of a supporting software system. The methodology is strongly communication-driven, and combines models of organizational communication with a model of speech act information content. Although the methodology can be used to guide development of automated IS, but can also be valuable if no automated system is implemented, in that it clearly describes the authorization and communication relations that can be used to improve on the way business is conducted. In this way, business processes are no longer seen as fixed, but as evolving and negotiable structures.

In modelling the domain two perspective on the environment are distinguished: Environment of Discourse (UoD) and Universe of Discourse (EoD). The EoD/UoD distinction separates clearly the activities or operations of an organization from the things which are operated upon. Traditional IS development methods emphasize the modelling of the UoD almost exclusively and do not take the EoD into account.

Based on the separation of EoD and UoD, first the authorizations are modelled at a high level of abstraction. From this the communication (transaction and contract) models can be derived, describing the essential communicative acts. This is complemented with the task model. With the specification of EoD models, the UoD modelling can be relieved, and we proposed an OO representation (NORM) for this. Advantages of this approach are that UoD specifications are easier to understand and more reusable, whereas the EoD models give an accurate insight in the authorizations and communications of the domain.

The main difference with traditional approaches is that the approach presented here gives proper weight to the demanding problems occurring in CIS design. These are: (i) problems of conflicting conceptual frameworks used to specify object, functional and data flow models; (ii) the establishment of authorizations; (iii) the use of standards and application architectures. All models here use concepts from the related business logic and communication frameworks, and are supported with one formal logic. They draw from the restricted vocabulary of the communication terms and their conceptual meaning defined in a Lexicon. Secondly, the Communication Model highlights who is responsible for a certain database, and it also contains primitives for assigning and retracting authorizations dynamically. The deontic language also allows the specification of deviating behaviour (failure handling, sanctions etc.). Being based on speech act theory, our communication language contains higher-level primitives than other approaches based on the concept of communication as data flow, or communication as synchronization. Furthermore, an architecture supporting the communication framework is provided (see below).

4. Intelligent agent architecture

In this thesis an agent architecture for CISs is described. I refer to an autonomous CIS with tasks and contracts as a *Cooperative Information Agent* (CIA).

Work in the agent community mainly focuses on agent theories, formally describing the (mental) components of agents. Also, in AI-based agent theories little attention is paid to the authorization specification so important for CIAs. The agent architecture proposed here is not based on a theory of mental attitudes but instead on a theory of interactions for which the agents are used.

Although the methodology is independent from implementation, an agent-oriented implementation fits well in the approach. I feel that providing an agent architecture gives an implementation that follows the specification closer than a traditional implementation. The specification method provides a link between agent theory and agent implementations and can be applied to the understanding and design of Cooperative Information Agents. An additional advantage of using agent-technology is that agents can be implemented as wrappers for legacy systems, thereby giving them the functionality of a CIA.

What makes the CIAs different from most other cooperating systems described in the literature is that also in the fulfilment phase of communication the CIAs are autonomous. They can at any moment decide not to honour the contract they agreed upon. Furthermore, the communication protocol that other systems use are more limited than the one proposed here, or even fixed. Although some of them give more explicit details about the ways optimal contracts can be formed (using utility functions), once a contract is established those systems cannot violate the contract anymore, thus loosing some of their autonomy.

Adequacy of LAP for CIA design

Finally we can revert to the main goal of this thesis: the application of the language-action perspective on the design of cooperative information systems. The LAP was developed to better describe what *humans* are doing while communicating. With the approach described in this thesis I brought over the theory of communicative action to the

communication of formal *systems*. In order to be applicable and adequate as a foundation for IS design it is important to be aware of the adaptations that are made to the original theory. The adaptations I proposed deal with the formal and explicit description of obligations and authorizations, and the different claims made on basis of the relationship between agents and their effects. Furthermore a way of structuring the communication process and details is presented, with emphasis on the symmetrical communication between communicating agents.

Based on the results and conclusions we can conclude that, *by bringing over the research on how people coordinate their activities by communication to the field of IS engineering, a language-action perspective can be adequately and successfully applied to the design of cooperative information systems for the support of business communication.*

8.2. FUTURE RESEARCH

The agenda for future research concerns research on theory, implementation and practice.

Theory

A topic for further research are the relations between contracts, mentioned in section 5.2.3. One contract can be a specialisation of another contract or an extension of that contract. Also an agent can have related contracts with different parties at the same time. We should be able to prove that these contracts are mutually consistent.

Further research can be done to extend the generality of failure handling, including the re-negotiation of original contracts when failures occur, bringing in a new agent (an outside "facilitator") to deal with the failures or delegate a failed transaction to others.

My main interest at the moment concerns work on the agent theory. The formal framework includes the possibility to describe the agent's intentions and beliefs. The goal is to provide a complete agent theory specifying a set of mental attitudes. [Dignum and van Linder, 1996] is a first attempt at this, and bears similar ideas as presented in this thesis. Work should also be done on representing general pragmatic knowledge, in order to improve the agent's answer capabilities, and ways to model the other agents' knowledge, beliefs and capabilities (second order mental state).

The agents now behave communicatively as opposed to strategically. Many Distributed AI research is focuses on strategic behaviour, and especially utility functions and negotiation protocols. To let the CIAs behave in a strategic way we have to look into the research on decision science and game theory.

Although the logical framework presented here provides a solid basis for the formalization of communicative actions, there are some open ends. These include: the explicit reference to time in dynamic logic, e.g. along the lines of [Dignum and Kuiper, 1997]; the negation of communicative actions; the combination of certain speech acts with conflicting obligations and authorizations, and their logical consequences. Also, in the logic until now all deontic statements were unconditional. In practice, a certain obligation or authorization only obtains under certain conditions: a specific event, a specific time, or more interesting another obligation or authorization. This should be worked out.

Implementation

First of all additional work should be done on the specification language. In section 5.1.4.2 instant services and delayed services were described. One objective is to refine and extend this small set of service types by analyzing more case studies, and the second objective is to implement this in the language. Also the Prolog-like notation for working through a list of alternatives has not yet been implemented.

Research is being undertaken to extend the UoD modelling technique to provide an information model of a basic user interface, along the lines of [van den Boogert, 1996]. The idea is that the communication specification can be used for the generation of intelligent interface agents. Since the agent knows of the different message types and has knowledge of what information to get next, with the help of libraries present in User Interface Management Systems, graphical interface objects can be generated to support the task of the user (and agent).

Another topic for future work concerns the lexicon. In the lexicon we can describe the ontology used by the agent. Although it was not described here, the agreement of the agents about the meaning of the terms that are used in the communication is of prime importance. This can be supported by using standard lexicons for general terms and domain lexicons for specific domain knowledge.

Our short-term goal is to enhance the prototype implementation of the CIA so that more experience can be gained with contingency handling in an "agent-oriented DBMS" to be built as an extra layer around a traditional DBMS. We are currently working on an implementation using Java as implementation platform (see appendix C).

Practice

I proposed the use of CIAs in different application settings, inter-organizational, as in Business Computing, EDI, or intra-organizational, as in Groupware and Workflow Management. An interesting topic for future research is how this view can be applied to the problem of Open EDI. Business today has a much larger component of rapid project-based partnerships that are created and dissolved in time scales too small to permit a full-blown standards process to play out its consensus building. The goal is to sustain ad hoc and short-term trading relationships using simpler legal codes. It is our contention that to realize Open EDI, we must first understand, i.e. know, the essentials of such business processes and communicative action.

Much theoretical research up till now takes a general programming view on transactions, but it is questionable that this is what is needed in practice. At least it seems clear that many (basic) transactions and contracts carry over from one application to another, and the investigation of these reusables is of immediate practical relevance.

APPENDIX A: COLA EBNF GRAMMAR

```
/*      [] = optional;       {} = sequence (1, or more);       | = or (choice)      */
/*      () = grouping, for clarity;       tokens in UPPERCASE or between 'quotes   */

agentprogram :      BEGIN tasks transactions contracts END

/* tasks specification */
tasks :             TASKS [{decl}] {task} END-TASKS
decl:               (INT|CHAR|CHAR*|...) var_name [{',' var_name}] ['=' ident] ';'
var_name :          ident          /* = string */
task :              TASK task_name taskbody END-TASK ';'
task_name :         ident
taskbody :          SUBTASKS ':' {subtask ';'}
                    [CONSTRAINTS ':' {constraint ';'}]
                    [DEADLINE ':' {deadline ';'}]
                    [DEPENDENCIES ':' {dependency ';'}]
                    [CONTINGENCY ':' {contingency ';'}]
                    GOAL '=' goal
                    [EXIT '=' subtask [{XOR subtask}]]
                |   '=' goal
subtask :           subtask_name | result |    transaction | action |
                    ACCEPT '(' trans_name ')'
subtask_name :      ident
transaction :       'T' '.' trans_name ['(' arg [{',' arg}] ')']
action :            'A' '.' action_name ['(' arg [{',' arg}] ')']
trans_name :        ident
action_name :       ident
arg :               ident
constraint :        subtaskspec_list BEFORE subtaskspec_list
              |     ALWAYS '(' subtaskspec '=>' temp_constr ')'
subtaskspec_list :  subtaskspec | '(' subtaskspec {(AND|XOR) subtaskspec} ')'
temp_constr :       (NEXT|SOMETIMES) '(' constr_el ')'
constr_el :         subtaskspec [{(AND|XOR) subtaskspec}] |
                    subtaskspec_list BEFORE subtaskspec_list | temp_constr
subtaskspec :       subtask
deadline :          ['P'][condition '<<'] goal '<<' condition | goal '!' | goal 'ASAP'
condition :         proposition | subtask | time | subtask '+' time | now '+' time
proposition :       string ['(' arg [{',' arg }]')']
dependency :        result DEPENDS-ON result '('[CREATE] FAIL [UPDATE]')'
result :            '"' ident '"'
contingency :       RESULT result
                    CREATED-BY subtask_name
                    CLOSED-BY action_spec
                    UPDATED-BY action_spec
                    INVALIDATED-BY action_spec [{ ',' actionspec }]
                    [REVALIDATED-BY action_spec]
                    COMPENSATED-BY action_spec
                    END-RESULT
action_spec :       transaction | action
goal :              goalspec [{XOR goalspec}][{AND goalspec {XOR goalspec}]}|XOR SKIP]
goalspec :          subtaskspec ['(' pref ')']
pref :              value
/* transactions specification */
transactions :      TRANSACTIONS [{decl}] {transaction_decl} END-TRANSACTIONS
transaction_decl :  TRANSACTION trans_head trans_body END-TRANSACTION ';'
```

```
trans_head :        trans_name ['(' arg ':' type [{',' arg ':' type}] ')']
trans_body :        AGENTS ':' {agent ':' type ';'}
                    {agent CAN SEND MESSAGES ':' {message_spec ';'} }
                    CONSTRAINTS ':' {trans_constr ';'}
                    [DEADLINE ':' {trans_deadline ';'}]
                    GOAL '=' message [{(AND|XOR) message}]
                    [EXIT '=' message [{(AND|XOR) message}] ]
agent :             ident
type :              ident
message_spec:       message TO (agent | self)
message:            REQUEST '(' action ')'                    /* DIRc */
                |   COMMAND '(' action ')'                    /* DIRa */
                |   ORDER '(' action ')'                      /* DIRp */
                |   REQUEST '(' AUTHORIZE '(' action ')' ')'
                |   COMMIT '(' action ')'                     /* COM */
                |   SUGGEST '(' proposition ')'               /* ASSc */
                |   ASSERT '(' proposition ')'                /* ASSa */
                |   CLAIM '(' proposition ')'          /* ASSp */
                |   ASSERT '('REFUSE-TO'('action [','reason]')''')'
                |   ASSERT '(' ACCEPT '(' (action|proposition ) ')' ')'
                |   NOMINATE '(' proposition ')'              /* DECLc */
                |   DECLARE '(' proposition ')'               /* DECLa */
                |   ESTABLISH '(' proposition ')'     /* DECLp */
                |   AUTHORIZE '(' message ')
                |   RETRACT '(' action ')'
                |   FORBID '(' action ')'
                |   PERMIT '(' action ')'
                |   CONFIRM '(' action ')'
                |   action
reason :            string
trans_constr :      message_list BEFORE message_list
                |   ALWAYS '(' message '=>' temp_tr_constr ')'
message_list :      message | '(' message {(AND|XOR) message} ')'
temp_tr_constr :    (NEXT|SOMETIMES) '(' trans_constr_el ')'
trans_constr_el :   message [{(AND|XOR) message}] | message_list BEFORE message_list |
                    temp_tr_constr
trans_deadline :    ['P'] [deadl_cond '<<'] message '<<' deadl_cond | message '!' |
                    message ASAP
deadl_cond :        proposition | message | time | message '+' time | now '+' time

/* contracts specification */
contracts :         CONTRACTS {contract} END-CONTRACTS
contract :          CONTRACT contract_name contract_body END-CONTRACT ';'
contract_name :     ident
contract_body :     AGENTS ':' {agent ':' type ';'}
                    CLAUSES ':' {clause ';'}
clause :            clause_name':'(OBL|AUT '('agent ',' action_spec ')')|ACC'('agent')'
                    IN ':' transaction [{',' transaction}]
                    [DEADLINE ':' obl_deadline]
                    [GOAL':'{action_spec '=>' clause_name [{'&'clause_name}]}]
                    [EXIT ':' {(cancel '(' action ')' | transaction) '=>'
                                clause_name [{'&' clause_name}]}]
                    [MODIFIED-BY {(action_spec|message)}]
                    END clause_name
clause_name :       ident
obl_deadline :      [(action_spec | time | action_spec '+' time | now '+' time) '<<']
                    action_spec '<<' (action_spec|time|action_spec'+'time|now'+'time)
                |   action_spec '!' |   action_spec ASAP
```

APPENDIX B: AGENT ALGORITHMS

In this appendix high level algorithms are given for the several functions of the Contract Manager (as described in section 4.4.1), the Task Manager (section 4.4.2, and section 5.3.2), the Service Execution Manager (section 4.4.3), and the Communication Manager (section 4.4.4). It is by no means a full description of all implemented algorithms. Rather it shows the interaction between the Managers and describes the data structures used by them. The implemented agents will be made available on our web site (see colofon).

With the (task, contract and transaction) specification of an agent, also a domain (UoD) model is made (as described in chapter 7). This specifies both the structure (type) of the domain entitities as well as the instances. Furthermore the elementary actions (and services) of the agent are specified. The models are the input for the agent and stored in the appropriate knowledge bases of the agent.

```
run interpreter (CoLa specification, domain-model, services-model, planning-
function_TM, schedule_TM, replan_TM, contingency-function_TM, contract-function_CM,
violation-function_CM, dissatisfaction-function_CM)
/* fill kb's, db, initiate */
L <- initial-domain-knowledge(domain-model(structure));
B <- initial-beliefs(domain-model(instances));
T <- initial-tasks(CoLa-spec(TaLa));
C <- initial-contracts(CoLa-spec(coLa));
TR <- initial-transactions(CoLa-spec(Trans));
S <- initial-services(services-model);
P <- ∅;                                  /* the plans are initially empty */
CP <- ∅;                                 /* current plans are initially empty */
A <- ∅;                                  /* the agenda is initially empty */
/* MS, (mental) state of the agent contains B, T, C, TR, S, P, CP, A */
for each t ∈ T and p ∈ P
      P <- planning-function(t);         /* generate initial plans */
      if not contains-condition(t.p)     /* only for plans without time or*/
      then CP <- replan(t.p);            /* propositions generate current plan*/
endfor;
A <- schedule(CP);                       /* put plans on agenda */
loop forever                             /* main loop, initiates managers*/
  i = pop(A);                            /* i=1st agenda-item (intention)*/
  case i
  action:
      r = execute(i);                    /* r = result of execution */
      if r ≠ ∅
      then update(MS, r);                /* update kb's */
      else                               /* the execution of action i failed */
            CP <- replan(i.p);           /* re-plan, look for alternatives */
  transaction:
      communication-plan(i);             /* initiate trans (message plan) */
  obl-check:
      r = execute(i);
      if r == ∅
      then violation-function(obl, i.contract);
  endcase;
  TRIG <- triggers;
  for each trig ∈ TRIG
      trig.cp = replan(trig.p);          /* (re)plan on condition satisfied */
  endfor;
  time.cp = replan(time.p);              /* (re)plan because time passed */
  process-incoming-messages;
  send-message;
  A <- schedule(CP);                     /* put plans on agenda */
endloop;
end;
```

```
/* Task Manager functions */
/* planning function */
planning-function(task)
ST <- collect-subtasks(task);
G <- collect-goals(task);
E <- collect-exits(task);
p = make-plan(ST, G, E, task.constraints, task.deadlines);
for each t ∈ ST
      ST <- planning-function(t);
endfor;
return p;
end;

/* make plan, dependency graph construction */
make-plan(ST, G, E, constraints, deadlines)
for each x ∈ constraints, deadlines, ST, G, E
      X <- TA(x);                /* Tableau decomposition function [Ngu et al,1994]*/
endfor;
return compose_graph(X)                  /* graph composition [Kieronska,1991] */
end;

/* decision about which plan to move to agenda */
schedule(CP)
for each cp ∈ CP
  if changed(cp)
  then conflict-resolution(cp);          /* try to add cp to agenda */
endfor;
return order(A, deadline, priority);     /* order actions on agenda */
end;

/* decide which obl gets priority */
conflict-resolution(plan)
for each ai ∈ A, pi ∈ plan
      if unify(pi,ai) ≠ ∅                /* check actions vs. agenda items */
      then /* conflict */
            if pi.obl ∉ C
            then   if ai.obl ∈ C
                   then exit(conflict-resolution)
                   else   if pi.priority < ai.priority
                          then exit(conflict-resolution);
            else   if ai.obl ∈ C
                   then   if pi.priority < ai.priority
                          then exit(conflict-resolution);
endfor;
for each ai ∈ A, pi ∈ plan
      if unify(pi,ai) ≠ ∅                /* delete all conflicting actions on agenda */
      then replan(ai.p);
endfor;
A <- add(plan);
end;

/* contingency plan */
contingency-function(result, task)
if revalidate(result, task) == ∅
then   compensate(result, task);
end;

/* re-plan (sub)task */
replan(plan)
retract(A, plan.cp);                     /* remove cp from agenda */
for each ai.result ∈ plan.cp
      contingency-function(result, ai.t);
endfor;
cp = next(plan);                         /* choose next current plan */
if cp == ∅
then   replan(getparent(cp));            /* backtrack */
end;

/* Contract Manager functions */
/* set up contract negotiation */
setup-contract(agent, action)
strategy = choose-negotiation-strategy(agent, action);
negotiate(agent, strategy, action);
end;

/* negotiation */
negotiate(agent, strategy, action)
case(strategy)
simple:
```

```
  case(action)
  offer:
       if acceptable(offer)
       then   add-message(agent, accept);
              C <- add(offer.contract);
       else   add-message(agent, counter-offer);
  counter-offer:
       if acceptable(counter-offer)
       then   add-message(agent, accept);
              C <- add(counter-offer.contract);
       else   add-message(agent, refuse);
  refuse:
       cp = replan(action.plan);
  accept:
       C <- add(accept.contract);
  default:                                  /* this agent starts negotiation */
       add-message(agent, offer, action);
  endcase;
/* other negotiation strategies can be inserted here */
endcase;
end;

/* adaptation of contracts omitted */

/* violation of obligation */
violation-function(obl, contract)
communication-plan(obl.trans);            /* initiate transaction */
change-state(obl.trans, contract);
end;

dissatisfaction-function(message, contract)
change-state(message.trans, contract);
end;

/* change state */
change-state(trans, contract)
state = new-state(contract, trans);
if exist state.obl
then   for each obl ∈ state
              CP <- add(obl-check(state.obl));
       endfor;
if exist state.auth
then   for each auth ∈ state
              Services.auts <- add(state.auth);
       endfor;
end;

/* Communication Manager functions */
process-incoming-messages                 /* handling of incoming messages */
M <- incoming-messages;
for each m ∈ M and tr ∈ MQ                /* MQ = message-queue */
  case m
  cancel(result):
       contingency-function(result, result.task); /* trigger contingency */
       CP <- replan(result.plan);
       change-state(cancel(result), result.contract);
       retract(MQ, m.tr);                 /* retract tr from MQ */
  exit(tr):
       CP <- replan(tr.plan);             /* fail, replan */
       retract(MQ, tr);
  request(action):
       request-execution(m.sender, action);
  offer:
       negotiate(m.sender, strategy, offer);
  counter-offer:
       negotiate(m.sender, strategy, counter-offer);
  refuse:
       negotiate(m.sender, strategy, refuse);
  accept:
       negotiate(m.sender, strategy, accept);
  default:
       case origin(m.tr)
       task:                                /* m according to task trans */
              update(MS, m);                /* process answer */
              continue-transaction(m, m.tr);
       contract:                            /* m according to contract trans*/
              if m ∈ dissatisfaction-tr
              then dissatisfaction-function(m, tr.contract)
              else   if m ∈ tr.goal
```

```
                    then change-state(tr, tr.contract); /* state change */
                    else continue-transaction(m, m.tr);
                    endif;
        endcase;
  endcase;
endfor;
end;

/* construct MQ (message-queue) */
communication-plan(trans)
MQ <- makeplan(trans);
end;

/* initiate sending messages */
send-message
send(pop(MQ));
end;

/* add message to be send to message queue */
add-message(agent, message, action)
MQ <- add(agent, message, action);
end;

/* Service Execution Manager functions */
/* request-execution */
request-execution(agent, action)
if check-authority(agent, action) == ∅
then add-message(agent, refuse-to(action));     /* agent has no authority */
if contract-initiated(agent, action)            /* check if contract with agent */
then cp = add(action, action.contract);         /* yes, put on agenda */
else                                            /* no contract with agent */
  if check(Services, action) == ∅               /* check if service available */
  then add-message(agent, dontknow(action));    /* no: tell agent*/
  else                                          /* yes */
       if check(Contracts, action) == ∅         /* contract available ? */
       then offer(agent, contract);             /* offer existing contract */
       else setup-contract(agent, action);      /* else set up new contract */
end;

/* execution */
execution(action)
if check(Services, action) == ∅
then   if check(Services, supplier, action) == ∅
       then add-message(all, request(action));
       else   if check(Contracts, supplier) == ∅
              then add-message(supplier, order(action));
              else add-message(supplier, request(action));
else return exec(action);
end.
```

APPENDIX C: IMPLEMENTATION ARCHITECTURE

1. INTRODUCTION

This appendix describes a prototype of a Cooperative Information Agents (CIAs) system, its components and what methods/technology will be used to create them. As a starting point the architecture from chapter 4 is taken. As will become clear from the rest of this document, the general idea is to create the system using (separate) Java applications for the different agents, communicating to each other using KQML on top of the TCP/IP networking protocol and using persistent objects for their knowledge bases. As much as possible available and suitable components have been selected to ease implementation.

In section 2 some remarks about the agent components are made. In section 3 the chosen form of implementation for the system in general and the components will be described and motivated.

My thanks to Sander Bos for implementing my ideas in the prototype.

2. COMPONENT ASPECTS

In chapter 4, a conceptual architecture for Cooperative Information Agents is described. Here some additional comments are made.

2.1 COMMUNICATION

The different autonomous agents must have a means to communicate with each other. In principle the only requirement is that a Communication Manager in one agent can send a message, consisting of a string, to the communication manager of another agent, but several things may be considered:

- **Lower Layer**, the medium over which the messages are send to each other. Examples are telephone lines, direct network connections, e-mail, and program procedure calls. The choice highly depends on the realization of the agent itself.
- **Topology**, the individual agents must be placed in the total system somehow. The agents must be able to identify and communicate with each other.
- **Protocol**, a structured way of communicating. Because existing media may be used and several systems for agents have been developed, it is wise to adopt existing standards (provided those match the architecture described here).

2.2 KNOWLEDGE BASES

The knowledge bases must be stored by each agent individually. The managers of the agent interpreter described above must have access to several of the knowledge bases (several are shared, e.g. the obligations on the agenda contain tasks of the Tasks Knowledge Base). The amount of information will likely be large, the structure of the data is diverse (not relational), and is subject to constant change.

Specifications written in (E)BNF must be converted to the knowledge bases.

3. ARCHITECTURE MAPPING

In this section first a general overview of the proposed implementation methods is given (figure A) which is followed by a description and argumentation of the implementation of the individual components. The detailed part starts with an explanation of the different components, which is followed by sections on communication with other agents and the implementation of the KBs. The basis of implementation is formed by the Java Agent Template, discussed in section 3.2.2.

3.1 TECHNICAL ARCHITECTURE OVERVIEW

In figure A the architecture is depicted with indications of the means by which we propose to implement them.

3.2 TECHNICAL ARCHITECTURE PER COMPONENT

There are several issues that must be taken into account when choosing a form of realization for a particular component:

- It must be feasible to implement the component using the selected technology.
- The selected technical architecture should correspond with the ideas brought forward in chapter 4. For instance, both the autonomy of and communication between agents should be safe-guarded.

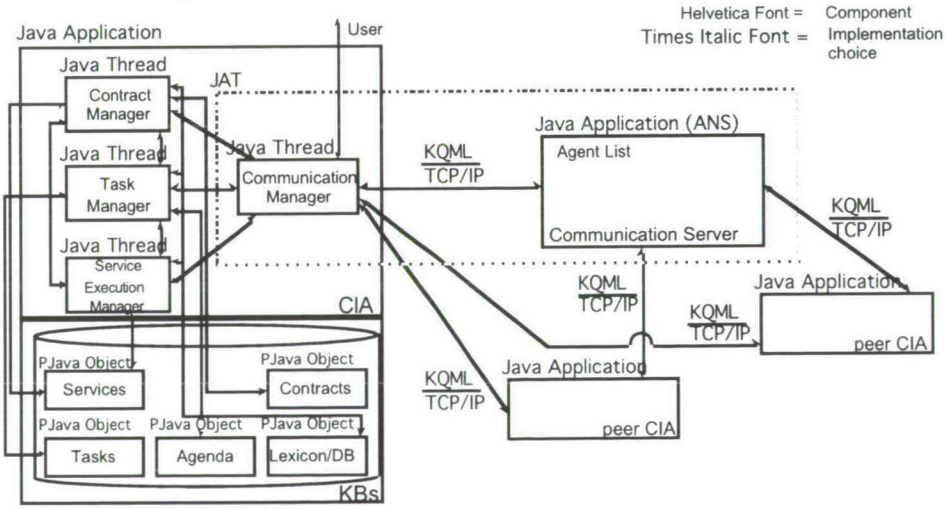


Figure A. Technical Agent Architecture

- A short-term realization with the architecture must be possible. Intermediary results are needed for validation ('are we building the right system').
- Existing standards must be followed. This must be done to comply with existing systems, and might enable us to use existing components.
- Since the system built will be used for further research, it should be created in a structured fashion. Relevant are good documentation and a modular approach (so that created components can be exchanged with another implementation).

The chosen architecture consists of a Java application for each agent. The Java Agent Template, which offers a basic implementation in Java of autonomous communicating agents, will be used as a basis for the program. The knowledge bases will be implemented using persistent objects. These choices are explained in detail below.

3.2.1 AGENT INTERPRETER

How: Every agent will be presented by a separate Java application, meaning that the resulting system will be a number of separate applications.
(As an added bonus, once the prototype is complete, it will be possible to convert the applications into applets, creating an easy way to demonstrate the agents on the World Wide Web. More on this in section 3.2.2).

Why: The choice of implementation language is the most important one to make. Because of the nature of the agents (which are basically reasoning programs), a language which naturally supports AI development (Prolog, Lisp) might seem suitable. However, there are a lot of operational units necessary, and in appendix B the operations are mapped to conventional (sequential) programming structures. For system maintenance it is best to create all of the agent interpreter in one programming system, instead of mixing different systems (Java is not suitable for incorporating different programming languages, if only because that would loose its portability).

Advantages of using Java are:

- Java is an up and coming language which currently has a lot of momentum. It is beneficial for the Infolab to get more experience with this language. Also, because of its current popularity lots of 'add-on' classes are becoming available for general functionality, such as database access.
- Its easy portability to different systems will mean both that the resulting prototype is system independent as well as that one of the goals of the CIA-paradigm, agentifying of legacy systems, should be easy to accomplish.
- Because of its portability Java is used often for implementing mobile agents. Although the agents here are not mobile it might be possible that technology developed for those systems could be applied here (With mobile agents, objects move from machine to machine. The technique used for this, conversion of objects to data-streams, can also be used to store objects).
- Java is a rich language, constructs and structures needed should be relatively easy to accomplish.

The biggest disadvantage is that Java is a sequential language, and therefore not immediately suited for AI development (e.g. there is no built-in mechanism for back-tracking in case a task fails). Most of the system to be created will not require those AI extensions, though.

Another important choice made for the agents is to create a separate application for each of them, instead of creating different 'threads' for each agent within one application. The great advantage is that each agent is really autonomous, and that it will be automatically possible to run the agents on different machines. Of course, this requires inter-application communication, but this should not present a large problem. What is needed for this architecture if multiple agents are to run on one machine, is a platform which allows multi-tasking[1].

[1] Without going into great detail, this might not present a problem for for example a pre-emptive system such as Microsoft Windows 3.1, because the communication calls will provide automatic interleaving.

Contract Manager

How: The Contract Manager, one of the parts of the agent interpreter, will be an object implemented using a thread process. Just like the other managers (except for the Communication Class) described below, the Contract Manager will be an instance of a class which inherits the member-variables and methods of a special Manager super-class. When new contracts must be negotiated, the services database can be queried, or, when the Services knowledge base does not contain information on the required action, requests for the execution of an action can be send to the other agents.
Once a contract has been established the Transaction Management part of the Contract Manager must make sure the Contract is upheld. This functionality is called upon directly by the Task Manager.

Why: The Contract Manager forms a distinct unit within the agent interpreter, which can be called upon by the other units (described below). This indicates it should be an object. Setting up (negotiating) a contract can take quite a while, time in which other tasks may be completed. This suggests to run the contract manager concurrently with the other units, in a separate thread.

Task Manager

How: The Task Manager will also be an object implemented using a thread.
One of the important jobs of the Task Manager is to plan (and replan) the tasks on the agenda. Initially this will be done by using a simple ALAP (As Late As Possible) scheduling mechanism keeping in mind precedence relations between tasks, giving obligations to other agents priority. Every change in the agenda (apart from removing the first item of it) should lead to a renewed ordering of obligations. A graph of all possible execution paths for each task is stored.

Why: This is a copy of the motivation for the Contract Manager. The Task Manager forms a distinct unit within the agent interpreter, which for instance must be able to execute tasks concurrently with other processes. The first point indicates it should be an object, the second that it should run as a separate process (it needs to be able to start actions by itself, not when called).
Since agents will be performing one task at a time, a relatively simple scheduling algorithm can be used. The ALAP approach has the advantage that results of tasks are less inclined to become obsolete because other tasks fail or become invalidated. Also, other obligations that come up later but have a shorter deadline can be executed earlier this way. Using an ASAP (As Soon As Possible) approach would mean that contracts might be handled quicker. This competitive edge does not play a role as long as all agents use the same scheduling technique. As long as there are not too many obligations on the Agenda, they can be completely reordered every time the list of obligations changes significantly. Of course, after the first version of the CIAs is ready more research must be done on improving the planning, by using better rules and reordering the obligations on the Agenda less frequently. The goals of a task might be reached through different combinations of subtasks, actions and transactions. If execution of an element of a task fails, another 'path' to the goal must be selected. To this end a graph is created (through Tableau Decomposition) in which all possible execution paths (which lead to goal or exit states) ar placed. This will allow the Task Manager to quickly select and schedule the next possible plan for a task if the current plan fails.

Service Execution Manager

How: The Service Execution Manager will also be an object. It can be implemented with a thread, just like the two managers mentioned above, but it could also simply be an object instance. The way in which actual tasks will be executed is unknown at the time of writing. Java does support the *exec()* function, so that it can start external applications. The way in which these external applications would be supported depends on the actual application and the system it will run on.

Why: The Service Execution Manager provides services to the other units of the agent, namely the maintenance of (public and private) services lists and the execution of actions. It will not need to act on its own, it will only have to react to requests made by the other units. There are two reasons to run the Service Execution Manager in a separate process in the application:

- Conformity: All other units of the interpreter will run in separate threads.
- Because two of the other units might call upon the Service Execution Manager simultaneously, provisions to handle this event must be made already.

Communication Manager

How: The Communication Manager will provide the communication interface of the agent with the other agents. It must communicate outgoing messages to the correct agents and route incoming messages to the correct Manager (may be more than one), depending on the nature of the message. It will also provide the user with a simple interface to the agent. The implementation will be with an object running in a thread.
More on the communication between the Communication Managers is described in the next section. The Communication Manager will not handle the messages between the different Managers within the agent.

Why: To handle external communication properly, the Communication Manager must deal with interrupts. One of the ways to do this is by running communications as a separate process within the application.
Because the communication between the different managers of the agent directly translate to class-message calls between the different objects there is no need for the Communication Manager to handle this traffic.

The complete Interpreter

The Interpreter will thus consist of four Manager objects in separate processes. These objects will be able to initiate actions if the necessary events take place. They can communicate with each other through message-calls directly to the other objects.

Two important aspects have not been mentioned yet. First of all, the implementation will be done using an existing agent implementation, which will be described next. The reason why this has not been mentioned is because the implementation used should match the architecture described here, and not determine it. The other aspect is the implementation of the Knowledge Bases used by the agents. This will be explicated in the last section.

3.2.2 COMMUNICATION

This section will explain how communication between Agents takes place. This directly defines the external part of the Communication Manager.

Instead of creating a new implementation an existing system for the exchange of messages will be used. This system will be briefly introduced first, after which its usage for the implementation of the CIAs is outlined.

Java Agent Template (JAT)

The JAT provides a fully functional template, written entirely in Java, for constructing software agents which communicate peer-to-peer with a community of other agents distributed over the Internet. Usage is free for academic use, all sources are provided. JAT is created by Robert Frost at Stanford.

The agents use KQML as a message wrapper and include the functionality to dynamically exchange 'resources' (this is the basis of their operation). The JAT includes a user interface (see example is shown in Figure C), which allows one to evaluate the state of the system and send (KQML-) messages to it.

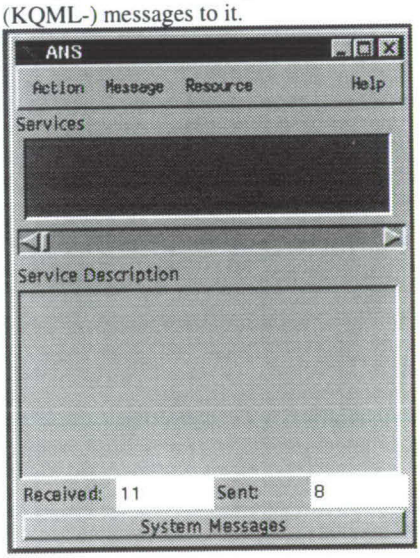


Figure C Example of User Interface of JAT

The basic system consists of a central name server and individual clients. When a client starts it notifies the central server of its presence. When two clients want to communicate, they can ask the server the address of each other after which they communicate directly. The idea is to sub-class these agents to provide more specific services (e.g., the central server could store information on services that the clients offer, and a client might offer a bus-schedule service), not to let the agents function differently. However, since the source-code is available and the agent is built of several objects, this can still be realized.

Usage of JAT for the CIAs

The JAT architecture consists of all the components to build an agent:

- A Communication Interface to communicate with other agents through KQML messages. Connected to this object is the Message Handler object which acts as an interface (buffer) between the actual agent and the Communication Interface.
- An Agent Context which defines the User Interface and handles the input from the user (and provides the necessary output for the user).
- A Resource Manager stores information about Network Class Loaders, Languages, Interpreters, Classes, Files and Addresses.
- The Agent Object itself, which defines how incoming messages (from the user, other agents and the system) are dealt with.

The Communication Interface object will be used together with the Message Handler and the Agent Context of this system. Together these objects provide the functionality of the

Communication Manager of the CIAs. The Resource Manager is somewhat linked to the Services Execution Manager but the way in which the agents of the JAT execute their tasks is different from the way that that is done in the CIAs[2].

To enable the rest of the CIA to work in this template several changes must be made (only a very brief introduction is given here):

- When an message comes in *InterpretMessage* is called which creates an Interpreter object for the message which handles it. This method must be overridden to be able to route the messages to the correct CIA Manager(s).
- When one of the Managers has a message to send it must create a KQML wrapper for it and call the *SendMessage* method of the MessageHandler object.

Finally, here are some of the properties of the communication using the JAT:

1. *TCP/IP* The agents communicate with each other using the TCP/ IP protocol. This protocol is available for most systems, and it allows the agents to run both on the same system (using different ports) as well as on different systems.
2. *Name Server* The agents identify each other through a central server. They know the address of this server, and announce themselves at startup (and unannounce themselves when they quit). The server keeps a list of names with corresponding addresses. When two agents want to communicate with each other, they first inquire the (TCP/ IP) address from the central server, after which they communicate **directly** with each other. The advantage is that agents keep their autonomy, a disadvantage is that they must have knowledge about the usage of addresses[3]. The Name Server class will be subclassed, to enable the broadcasting of messages to multiple agents related to negotiations. The Name Server Agent could be more sophisticated, e.g. by storing information about what services other agents can provide. This would create more efficiency in contract negotiation, but would simultaneously increase the dependency of the agents on the central server, thereby decreasing their autonomy. Special agents providing services such as the services-database described here could be added of course.
3. *KQML* The standard for communication between agents is KQML [Finin et al.,1994]. The content of the message can be easily integrated in a KQML-message, provided a performative is explicitly added. This will take some work though, because the set of performatives of KQML do not match those discussed in chapter 5.

3.2.3 KNOWLEDGE BASES

The Knowledge Bases will be implemented using persistent objects (the PJava references in Figure B are meant to indicate Persistent Java). Each Knowledge Base will have its own object class, in which the information is stored (in the systems memory). Through object persistence, these objects will then be stored on disk regularly. Inheritance will be used to give the Knowledge Base object classes the same basic data manipulation methods, which can then be changed and extended. The Managers of the CIA access the methods of the persistent Knowledge bases directly. There are several advantages of using Persistent Objects:

---

[2] When an agent in the JAT needs to perform a task it cannot perform itself it contacts an agent which **knows** where the service can be **found**, and execution of the task is then done by the agent which requires the action **itself**, after retrieval of the service.

[3] Another disadvantage of direct communication is that the Java applications cannot be turned into Applets any more, because the security systems of WWW-browsers do not allow this.

- From a design perspective they are easier. Structures have to be defined only once for usage in both the program and for storage of data. The Managers of the CIA evaluate/manipulate several of the objects-classes which are stored.
- No separate interface to the database is necessary. The changes for storing data should be minimal (see later).
- The object-orientation will make it possible to store more difficult data structures than can be easily achieved with relational tables (this would also require complex conversion functionality).

There are also some issues to take into account when working with Persistent Objects:

- First of all, there is no standard implementation available at the time of writing, although a lot of work is done on this:
  - Sun, the creators of Java, have released Object Serialization, which enables object instantiations to be written to streams. They are also working (internally[4]) on PJava, which will provide a sophisticated mechanism for persistence in Java. Both these systems require changes to be made to the JVM (Java Virtual Machine), meaning they can not operate on existing Java Platforms. These techniques will be added to new versions of the JDK (Java Development Kit).
  - JRB from O2 Technology offers persistent storage of objects in relational tables, but is limited to specific platforms and specific Data Base Management Systems. The usage license is also limited.
  - Objectstore PSE from ODI offers persistent storage in a non-native fashion, and is freely available. However, the system is based on 'tricks' to implement the object persistence.

  None of these methods seems completely suitable now. Objectstore PSE is available , but a superior (and more standardized) implementation might very well become available in a few months. The required changes for introducing persistence later consist of indicating which objects must be made persistent.
- Unless the persistent object manager is very sophisticated, all of the data will have to be stored in the system memory. This will of course lead to problems when databases get larger. When this happens, an Object Oriented Database will have to replace object persistence.
- As a related issue, if legacy database systems are used (e.g. if a database is 'agentified') object persistence might not be an option for such databases.
- Finally, the file-structure used by the persistent object manager is probably proprietary, which makes it harder for other programs to read this information. However, as long as the class-definitions are copied, other Java applications can analyze (or convert) the Knowledge Bases.

To summarize: Because there are no fully suitable object persistence methods for the Java language at the time of writing, but are expected very soon, the objects will just be stored in memory for now. Once a suitable persistence technique can be selected, it should be easy to integrate this in the prototype. Also, because of the reasons mentioned above, when designing the structure of the Knowledge Base objects the ability to replace the persistent objects with Object Oriented Databases must be kept in mind. This should not place limitations on the design.

[4] Internally is meant to say that no official announcement(s) have been made regarding the product. People involved in development of PJava indicated it would be released soon.

GLOSSARY

A word in SMALL CAPITALS *indicates a cross-reference.*

ACID: Atomicity, Consistency, Isolation, Durability. Properties of classical control mechanisms in databases.

action: non-communicative activity that is performed by an AGENT.

actor: someone or something that can act. Synonym for 'subject' (human as well as computerized), or 'linguistic agent'.

agenda: a set of deontic temporal constraints that defines the normative space (or OBLIGATION and COMMITMENT space) of all things that have to be done. It contains the actions to be performed by the AGENT, instantly or at some designated point in time.

agent: an autonomous computational entity with TASKS and capabilities (actions it can perform), that communicates with other agents by means of MESSAGES, and whose behaviour is not pre-defined but based on COMMITMENTS to other agents.

assertive: SPEECH ACT type used to say how things are. It makes a statement about the state of affairs in the world, and commits the speaker to the truth of the expressed PROPOSITION.

authorize: SPEECH ACT type used to grant AUTHORIZATION to other agents.

authorization: PERMISSION that allows the effective operation of a (communicative) act.

BAT (Business as Action game Theory): a generic business framework describing a BUSINESS LOGIC, developed by Göran Goldkuhl ([Goldkuhl, 1995, 1996]).

BDI (Belief-Desire-Intention): influencial agent theory by Rao and Georgeff ([Rao and Georgeff, 1991a, 1991b, 1993]), describing a logical framework for intelligent AGENTS based on the primitive modalities: BELIEF, desire and INTENTION.

belief: epistemic modality describing the notions and views of the AGENT on the domain.

BPM (Business Process Modelling): The process of modelling BUSINESS PROCESSES.

business communication: the network of discrete recurrent COMMUNICATIVE ACTIONS (or conversational TRANSACTIONS) that form the core of an organization. The focus is on COMMUNICATION processes to understand the business, instead of on the current organisational structures or documental flows.

business logic: generic framework for business TRANSACTIONS between actors, describing the roles of the two actors (a provider, and a customer of goods or services) and their COMMUNICATIVE and material ACTIONS. It distinguishes the phases: proposal, commitment (contractual), fulfilment, and completion (acceptance/claim).

business process: a series of coherent activities that creates a result with some value for an external or internal customer, through solving a problem or task for him. It is a meaningful whole of value adding activities (value chain).

cancellation: the undoing of a COMMITMENT by means of a cancel-message. The CONTRACT between the AGENT cancelling and the agent committed to might specify sanctional actions to be taken.

CIA (Cooperative Information Agent): an autonomous, intelligent COOPERATIVE INFORMATION SYSTEM executing TASKS and COMMUNICATING following CONTRACTS.

CIS (Cooperative Information System): Next generation automated information systems that supports the coordination of activities in and between organizations.

commissive: SPEECH ACT type used to change the situation in which it is uttered to fit the PROPOSITIONAL CONTENT of the speech act. The commissives lay the responsibility of the fit with the speaker. The commissive point is to commit the speaker to a future course of action.

commitment: a promise to a future conduct in order to establish a desired situation.

communication: coordination of activities by means of a process of exchanging MESSAGES. In this thesis, its essence is to commit the partners in communication to a course of action so that one can rely on the other.

Communication Manager: functional component of the AGENT interpreter that is responsible for routing MESSAGES between the agent and other agents and the agent and its responsible human counterpart, using a communication protocol and information sharing language.

communicative action: an "interaction between subjects that engage in a social relationship" ([Habermas, 1984]) with the aim of reaching a mutual understanding of the situation and goals pursued in order to be able to coordinate their actions. The subject's motivation is rational: they respond to requests because they presuppose that these requests can be justified when asked for.

compensation: undoing of achieved results and effectuation of sanctional actions.

Conceptual Model: explicit description of all relevant aspects, objects, relationships, and rules of a specific domain. Constructed for the purpose of problem understanding and as input for automated systems development. Following the LAP the CM consists of two interrelated models: the ENVIRONMENT OF DISCOURSE (EoD) model and the UNIVERSE OF DISCOURSE (UoD) model.

contingency plan: plan describing actions that can be taken in order to reach the goal if unexpected events leading to the failure of a subTASK occur. It specifies how to revalidate the result of the subtask again, or what should happen if it cannot, including COMPENSATION of dependent results. A CIA contingency plan consists of a set of results that have an internal object structure, and associated methods that specify the TRANSACTIONS (or tasks) that can be used to create, close, compensate, invalidate and revalidate the result.

contract: a set of mutually agreed OBLIGATIONS and related AUTHORIZATIONS (permissions and prohibitions) between different parties about services provided to each other, together with rules governing VIOLATION, i.e. consequences and (sanctional) actions to take if one of the agents does not adhere to its obligations and authorizations. It describes the effects of TRANSACTIONS, i.e. the factual or deontic temporal constraints that are created or deleted.
Formally, a contract is a set of tuples, each tuple consisting of a description of the deontic state (authorizations, obligations and accomplishments), in-transitions (transactions), out-transitions (transactions), and violation handling methods.

contract-base: a knowledge base that holds the CONTRACTS between the AGENT and other agents. It also contains old contracts and rules for setting up new contracts (negotiation support).

Contract Manager: the functional component of the AGENT interpreter responsible for managing the CONTRACTS of the agent. It supports the creation of new contracts, and adaption of existing ones, contractual COMMITMENT (including transaction) management, and claim management.

deadline: the time by which an action should be performed, or a particular state should be reached.

declaration: SPEECH ACT type used to bring about some new state of affairs of the world. By making a declaration the world is changed according to it by saying so.

delegation: transfer or assignment of AUTHORIZATION, OBLIGATIONS and control to a subordinate actor.

deliberative agent: "an AGENT that contains an explicitly represented symbolic model of the world, in which decisions (e.g. about what actions to perform) are made via logical (or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulation"([Genesereth and Nilsson, 1987], [Wooldridge and Jennings, 1995]).

DEMO: Dynamic Essential Modelling of Organizations. A BUSINESS PROCESS modelling method based on social theory, grounded in the language philosophy of Searle and Habermas, developed by Dietz ([Dietz, 1992a, 1994a,b]).

deontic logic: the logic to reason about OBLIGATIONS and AUTHORIZATIONS. A branch of modal logic that is concerned with (reasoning about) norms and NORMATIVE versus non-normative behaviour.

directive: SPEECH ACT type used to change the situation in which it is uttered to fit the PROPOSITIONAL CONTENT of the speech act. The directives lay the responsibility of this fit with the addressee. The directive point is to try to get the addressee to do things (carry out a course of action represented by the propositional content). If successful and non-defective, the hearer commits himself to do it.

dynamic deontic logic: the logic used to specify and reason about OBLIGATIONS and AUTHORIZATIONS over actions.

EoD (Environment of Discourse): the projection of the domain concerned with information context. The EoD describes the discourse as a process without going into the contents (what is said, and more in particular the meaning of these terms, is described in the UoD). Typical objects in the EoD are the linguistic agents (human or computerized), the MESSAGE types, and the rules that prescribe and describe the COMMUNICATION (AUTHORIZATIONS and OBLIGATIONS). Also often called 'organizational environment'.

essential communication: a view on COMMUNICATION where one abstracts from technological issues and reproduction of data.

hybrid agent: a neither completely DELIBERATIVE nor completely REACTIVE AGENT.

illocutionary act: the complete SPEECH ACT, consisting of PROPOSITIONAL CONTENTS, ILLOCUTIONARY CONTEXT and ILLOCUTIONARY FORCE.

illocutionary context: the relevant knowledge about the situation in which the SPEECH ACT is made. This knowledge can be factual, about the place the speech act is performed, but also epistemic, about the INTENTIONS and BELIEVES of the participants in the speech act. It also includes the speaker and addressee of the speech act themselves.

illocutionary force: that part of the SPEECH ACT that expresses how the PROPOSITIONAL CONTENT of the speech act is to be taken. It is the expression of the speaker's INTENTIONS, the reasons and the goal of the communication.

illocutionary logic: the logical formalization of the theory of SPEECH ACTS ([Searle and Vanderveken, 1985]).

illocutionary point: indicates the type of effect for which the act is performed (the point or purpose). The illocutionary point describes what it is for the speaker to mean the utterance. The five illocutionary points that Searle distinguished are: ASSERTIVES, DIRECTIVES, COMMISSIVES, DECLARATIONS, and expressives. Other speech act theorists and language philosophers (cf. [Habermas, 1984], [Balmer and Brennenstuhl, 1981], [Chang and Woo, 1994]) have proposed different categorizations of speech acts.

informatics: the interdisciplinary field of science that deals with information and communication and their role in the functioning of dynamic systems.

intention: mental notion used in agent theories that indicate the AGENT's plan to perform an act or to bring about a certain state (goal).

intentional attitudes: mentalistic notions describing properties of an AGENT by which the agent's behaviour can be predicted, and that describe how the mental state of the agent is directed at or about objects and states of affairs in the world.
In agent theory these include the attitudes: knowledge, BELIEF, desire, INTENTION, goals, COMMITMENT, choice, decision, plans, preference, wish, ability, opportunity

LAP (Language-Action Perspective): a framework and foundation for modelling and design of computer support for COMMUNICATIVE ACTION. The LAP emphasizes what people *do* while communicating; how they create a common reality by means of language and how communication brings about coordination of their activities.

lexicon: the system that stores and manages the terminology of a certain domain.

logic: a set of formalisms for representing properties and reasoning over them.

message: a SPEECH ACT that describes the ILLOCUTIONARY FORCE together with its authorization claim (be it power, authority, or charity) and the content (a proposition or action).

negotiation: "a joint decision making process in which the parties verbalise their (possibly contradictory) demands and then move towards agreement by a process of concession or search for new alternatives" [Müller, 1996].

normative system: "A normative system refers to any set of interacting AGENTS whose behaviour can usefully be regarded as governed by norms. Norms prescribe how the agents ought to behave (by OBLIGATIONS), and specify how they are permitted to behave and what their rights are (AUTHORIZATIONS)" [Jones and Sergot, 1993]

objective world: the world of reference that tells how things are ([Habermas, 1984]). It supports the speaker's validity claim to truth in performing a SPEECH ACT, which entails that the speaker contends to represent the factual contents of the speech act as they are.

obligation: something that one must fulfil, something that should be done or brought about. An obligation is the result of a COMMITMENT of the actor, or of a command (given by another actor with power over the actor), or of an request by another actor that is given authority.

permission: the allowance of the COMMITMENT to and/or operation of an act (including COMMUNICATIVE ACTS).

propositional contents: the part of a SPEECH ACT that expresses what the speech act is about.

reactive agent: an AGENT that does not include any kind of central symbolic world model. It is capable of reacting to (known) events in the environment without engaging in complex symbolic reasoning and therefore usually reacts much faster to such events. The best known reactive architecture is the subsumption architecture ([Brooks, 1986, 1991a,b]).

Service Execution Manager: functional component of the AGENT interpreter that is responsible for starting, stopping and monitoring service execution, exception handling, and information (database) management.

services-base: the knowledge base, part of the AGENT architecture, that holds knowledge about the services the agent can provide to others, private actions it can perform, and also knowledge about services other agents can provide to it.

sincerity condition: attitudes or psychological states of the speaker towards the PROPOSITIONAL CONTENTS of a SPEECH ACT ([Searle, 1969]). If the propositional content conforms with the actual psychological state of the speaker, the act is said to be sincere. If a speaker is sincere, he intends to

do a certain predicated act by expressing something will hold in the future, and believes it is possible for him to do such act.

social world: the world of reference that tells how the participants stand towards each other ([Habermas, 1984]). It supports the speaker's validity claim to justice in performing a speech act, which regards the adequacy of the interpersonal relation between the speaker and the hearer

speech act: an uttance or MESSAGE with a performative nature. According to Austin's language-as-action theory: *what one is doing in saying something*. Speech acts are considered to be the basic or minimal units of linguistic communication.

Speech Act Theory: a means to analyze COMMUNICATION in detail at three levels: content (propositions), intention (illocution) and effect (perlocution). Form (syntaxis) of the communication is of less importance than why and what is communicated.
Speech Act Theory is based on the initial work of Austin ([Austin, 1962]) on performative use of language: although conversations are usually thought of as exchanges of information, the communications may not only be informative but also performative in that they change the state of affairs and commitment among the parties involved. The most prominent speech act theory is brought forward by the language philosopher J.R. Searle ([Searle, 1969, 1979]).

strategic action: interaction between subjects that strive after their own goals and plans (as opposed to COMMUNICATIVE ACTION) ([Habermas, 1984]). The subject's motivation is empirical: they try to maximise their own profit and minimise their own losses. Coordination is based on empirical contingencies, especially on a claim to, or use of power.

subjective world: the world of reference that tells how the speaker perceives the world ([Habermas, 1984]). It supports the speaker's validity claim to sincerity in performing a SPEECH ACT, which entails that the speaker is genuine in the performance of it.

task: a meaningful unit of work assigned to an AGENT.
Formally, a task is a tuple consisting of a set of subtasks, a set of constraints on the execution of the subtasks (dependencies), a CONTINGENCY PLAN, a DEADLINE (both possible empty), and goal and exit states expressed by an AND/XOR graph in canonical form. Elementary subtasks are either executed by the agent as an internal procedure (an action) or involves initiating communicative TRANSACTIONS with another agent.

Task Manager: functional component of the AGENT interpreter that is responsible for scheduling and planning TASKS, AGENDA management (adding and removing items, AUTHORIZATION and environment checks, and (OBLIGATION) conflicht resolution), and failure handling (including alternatives and CONTINGENCY PLAN execution).

tasks-base: the part of the AGENT architecture that holds the agent's TASKS.

transaction: logical grouping of (authorized) SPEECH ACTS with temporal constraints between them. Formally a transaction is a tuple consisting of a set of AGENTS, a set of possible MESSAGE types (speech acts), a set of temporal constraints on the synchronisation of the speech acts, a possibly empty DEADLINE, and goal and exit states (identified by message occurences).

transaction-base: the part of the AGENT architecture that holds the TRANSACTIONS the agent can perform.

UoD (Universe of Discourse): the projection of the domain concerned with information content (data, rules *about* the domain). It describes what is communicated, and more in particular the meaning of these terms.

violation: not adhering to OBLIGATIONS. It causes "sanctional" actions described in the CONTRACT to be triggered on the one hand, and rescheduling of the subTASK on the other.

LITERATURE

[Action Technologies, 1993]. Action Workflow Analysis Users Guide, Action Technologies, 1993.

[Agostini et al., 1994]. A. Agostini, G. De Michelis, S. Patriarca, R. Tinini, "Prototype of an integrated coordination support system", in: Computer Supported Cooperative Work, vol. 3, no. 3, pp. 209-238, 1994.

[Agre and Chapman, 1987]. P. Agre and D. Chapman, "PENGI: An implementation of a theory of activity", in: Proc. of 6th Nat.l Conf. on AI (AAAI'87), Seattle, WA., AAAI Press, Menlo Park, CA., pp. 268-272, 1987.

[Allwood, 1977]. J. Allwood, "A critical look at speech act theory", in: *Logic, Pragmatics and Grammar*, Ö. Dahl (ed.), Studentlitteratur, Lund, 1977.

[Allwood, 1980]. J. Allwood, "An analysis of communicative action", in: *The Structure of Action*, M. Brenner (ed.), Blackwell, Cambridge, MA., 1980.

[Alonso et al, 1996] G. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Gunthor, C. Mohan, "Advanced Transaction Models in Workflow Contexts", in: Proc. of the 12th Int.l Conf. on Data Engineering, New Orleans, Lousiana, March 1996.

[Alty et al., 1994]. L. Alty, D. Griffiths, N.R. Jennings, E.H. Mamdani, A. Struthers, M.E. Wiegand, "ADEPT - Advanced Decision Environment for Process Tasks: Overview & Architecture", in: Proc. BCS Expert Systems '94 Conf. (Appl. Track), Cambridge, UK, pp. 359-371, 1994.

[Anderson, 1958]. A.R. Anderson, "A reduction of deontic logic to alethic modal logic", in: Mind 67, pp. 100-103, 1958.

[Åqvist, 1984]. L. Åqvist, "Deontic logic", in: *Handbook of Philosophical Logic II*, D.M. Gabbay and F. Guenthner (eds.), Reidel, Dordrecht, pp. 605--714, 1984.

[Assenova and Johannesson, 1996]. P. Assenova and P. Johannesson, "First Order Action Logic: an approach for modelling the communication process between agents", in: [Dignum et al., 1996c]

[Aumann, 1976]. R. Aumann, "Agreeing to disagree", in: Annuals of Statistics, 4, pp. 1236-1239, 1976.

[Auramäki, 1988]. E. Auramäki, "Speech act based model for analysing cooperative work in office information systems", in: Proc. of 6th EFISS Symposium, Atlanta, Georgia, 1988.

[Auramäki et al., 1988]. E. Auramäki, E. Lehtinen, and K. Lyytinen, "A Speech Act Based Office Modeling Approach", in: ACM Transactions on Office Information Systems (TOIS), Vol. 6, No 2, pp. 126-152, 1988.

[Auramäki et al., 1992a]. E. Auramäki, R. Hirscheim, and K. Lyytinen, "Modelling offices through discourse analysis: The SAMPO approach", in: The Computer Journal, vol. 35, no. 4, 1992.

[Auramäki et al., 1992b]. E. Auramäki, R. Hirscheim, and K. Lyytinen, "Modelling offices through discourse analysis: A comparison and evaluation of SAMPO and OSSAD and ICN", in: The Computer Journal, vol. 35, no. 5, 1992.

[Auramäki and Lyytinen, 1996]. E. Auramäki and K. Lyytinen, "On the success of speech acts and negotiating commitments", in: [Dignum et al., 1996c]

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

[Austin, 1962b]. J.L. Austin, Philosophical Papers, Clarendon Press, Oxford, 1962.

[Baeten and Weijland, 1990]. J.C.M. Baeten and W.P. Weijland, *Process Algebra*, Cambridge University Press, Cambridge, MA., 1990.

[Bates, 1994]. J. Bates, "The role of emotion in believable agents", in: Communications of the ACM (CACM), vol. 37, no. 7, pp. 122-125, 1994.

[Bates et al., 1992]. J. Bates, A. Bryan Loyall, W. Scott Reilly, "An architecture for action, emotion, and social behaviour", Technical Report CMU-CS-92-144, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA., 1992.

[de Bakker et al., 1986]. J.W. de Bakker, J.N Kok, J.-J.Ch. Meyer, E.-R. Olderog, and J.I. Zucker, "Contrasting themes in the semantics of imperative concurrency", in: *Current Trends in Concurrency: Overviews and Tutorials*, J.W. de Bakker, W.P. de Roever, and G. Rozenberg (eds.), LNCS 224, Springer-Verlag, Berlin, 1986.

[Ballmer and Brennenstuhl, 1981]. T. Ballmer and W. Brennenstuhl, *Speech Act Classification: A Study in the Lexical Analysis of English Speech Activity Verbs*, Springer-Verlag, Berlin, 1981.

[Beek and Jager, 1993]. A. Beek and J.J. Jager, *Hoofdlijnen informatiekunde*, Wolters-Noordhoff, Groningen, 1993. (in Dutch)

[Belnap and Perloff, 1989]. N. Belnap and M. Perloff, "Seeing to it that: a canonical form for agentives", in: Theoria, 54, pp. 175-199, 1988.

[Bennett, 1991]. J. Bennett, "How do gestures succeed?", in: *John Searle and his critics*, E. Lepore and R. van Gulick (eds.), Blackwell, Cambridge MA, pp. 3-16, 1991.

[Bertino and Weigand, 1994]. E. Bertino, H. Weigand, "An approach to authorization modelling in object-oriented database systems", in: Data and Knowledge Engineering 12, pp.1-29, 1994.

[Biagioli et al., 1987]. C. Biagioli, P. Mariani, D. Tiscornia, "ESPLEX: a rule and conceptual based model for representing statutes", in: Proc. of the 1st Int.l Conf. on Artificial Intelligence and Law, ACM Press, New York, NY., pp. 240-251, 1987.

[Blyth et al., 1992]. A.J.C. Blyth, J. Chudge, J.E. Dobson and M.R. Strens, "The ORDIT approach to requirements identification", in: Proc. of Compsac '92, 1992.

[Bond and Gasser, 1992]. A.H. Bond and L. Gasser, "A subject-indexed bibliography of distributed artificial intelligence", in: IEEE Transactions SMC (Systems, Man & Cybernetics), 22 (6), pp. 1260-1281, 1992.

[van den Boogert, 1996]. P. van den Boogert, "Lexitrons", MSc Thesis, Infolab, Tilburg University, august, 1996.

[Boutilier, 1994]. C. Boutilier, "Toward a logic for qualitative decision theory", in: Proc. of 4th Int.l Conf. on Principles of Knowledge Representation and Reasoning (KR&R'94), J. Doyle, E. Sandewall, P. Torasso (eds.), Morgan Kaufmann, San Mateo, CA. pp. 75-86, 1994.

[Bowers, 1993]. J. Bowers, "COSMOS, AMIGO Advanced and MacAll II", in: Computational mechanisms of interaction for CSCW, C. Simone and K. Schmidt (eds.), COMIC Deliverable 3.1, ESPRIT BRA 6225, Lancaster University, Lancaster, 1993.

[Bowers and Churcher, 1988]. J. Bowers and J. Churcher, "Local and global structuring of computer mediated communication: Developing linguistic perspectives on CSCW in COSMOS", in: Proc. of 2nd Conf. on Computer Supported Cooperative Work (CSCW'88), ACM SIGCHI and SIGOIS, ACM Press, New York, NY., pp. 291-302, 1988. (also in Office: Technology and People, vol. 4, no. 3)

[Bratman et al., 1988]. M.E. Bratman, D.J. Israel and M.E. Pollack, "Plans and resource-bounded practical reasoning", in: Computational Intelligence, no. 4, pp. 349-355, 1988.

[Bratman, 1990]. M.E. Bratman, "What is intention ?", in: *Intentions in Communication*, P.R. Cohen, J.L. Morgan, M.E. Pollack (eds.), MIT Press, Cambridge, MA., pp. 15-32, 1990.

[Brazier et al., 1996]. F.M.T. Brazier, B. Dunin-Keplicz, N.R. Jennings and J. Treur, "DESIRE: Modelling multi-agent systems in a compositional formal framework", in: International Journal of Cooperative Information Systems, M. Huhns, M. Singh (eds.), special issue on Formal Methods in Cooperative Information Systems, 1996.

[Brooks, 1986]. R.A. Brooks, "A robust layered control system for a mobile robot", in: IEEE Journal of Robotics and Automation, vol. 2, no. 1, pp. 14-23, 1986.

[Brooks, 1991a]. R.A. Brooks, "Intelligence without reason", in: Proc. of 12th Int.l Joint Conf. on AI (IJCAI'91), Sydney, Australia, pp. 569-595, 1991.

[Brooks, 1991b]. R.A. Brooks, "Intelligence without representation", in: Artificial Intelligence, no. 47, pp. 139-159, 1991.

[Broy, 1986]. M. Broy, "A theory for nondeterminism, parallelism, communication and concurrency", in: Theoretical Computer Science, vol.45, pp. 1-62, 1986.

[Buchmann et al., 1992] A. Buchmann, M. Tamer Özsu, M. Hornick, D. Georgakopoulos, F. Manola, "A Transaction Model for Active Distributed Object Systems", in: *Database Transaction Models for advanced applications*, A. Elmagarmid (ed), Morgan Kaufman, San Mateo, CA. 1992.

[Burg and van de Riet, 1995]. J.F.M. Burg and R.P. van de Riet, "COLOR-X: Linguistically-based event modelling - a general approach to dynamic modelling", in: Proc. of Advanced Information Systems Engineering (CAISE'95), J. Iivari, K. Lyytinen, M. Rossi (eds.), LNCS-932, Springer-Verlag, Berlin, pp. 26-39, 1995.

[Burmeister and Sundermeyer, 1992]. B. Burmeister and K. Sundermeyer, "Cooperative problem solving guided by intentions and perception", in: Decentralized AI 3 - Proc. of the 3rd European WS on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW'91), E. Werner and Y. Demazeau (eds.), Elsevier Science Publ., North Holland, pp. 77-92, 1992.

[Castaneda, 1981]. H.N. Castaneda, "The paradoxes of deontic logic", in: *New Studies in Deontic Logic*, R. Hilpinen (ed.), Reidel, Dordrecht, pp. 37-85, 1981.

[Castaneda, 1982]. H.N. Castaneda, "The logical structure of legal systems: a new perspective", in: *Deontic Logic, Computational Linguistics and Legal Information Systems, vol. 2*, A.A. Martino (ed.), North-Holland, 1982.

[Castaneda, 1989]. H.N. Castaneda, "The content of legal speech acts and legal deontic logic", in Ned. Tijdschrift voor Rechtsfilosofie, 18 (2), pp. 108-129, 1989.

[Castelfranchi, 1995]. C. Castelfranchi, "Guarantees for autonomy in cognitive agent architecture", in: *Intelligent Agents: Theories, Architectures, and Languages*, M. Wooldridge and N.R. Jennings (eds.), LNAI 890, Springer-Verlag, Heidelberg, pp. 56-70, 1995.

[Chang and Woo, 1994]. M.K. Chang and C.C. Woo, "A Speech Act Based Negotiation Protocol: Design, Implementation and Test Use", in: ACM Transactions on Information Systems (TOIS), vol.12, no.4, pp. 360-382, 1994.

[Chen, 1976]. P.P. Chen, "The Entity-Relationship model - toward a unified view of data", in: ACM Trans. on Database Systems (TODS), vol. 1, no. 1, pp. 9-38, 1976.

[Codd, 1970]. E.F. Codd, "A relational model of data for large shared data banks", in: Communications of the ACM (CACM), vol. 13, no. 6, pp. 377-387, 1970.

[Cohen and Levesque, 1990a]. P.R. Cohen and H.J. Levesque, "Intention is choice with commitment", in: Artificial Intelligence 42, pp. 213-261, 1990.

[Cohen and Levesque, 1990b]. P.R. Cohen and H.J. Levesque, "Rational interaction as the basis for communication", in: *Intentions in Communication*, P.R. Cohen, J. Morgan and M.E. Pollack (eds.), MIT Press, Cambridge, MA., pp. 221-256, 1990.

[Coleman et al, 1993]. D. Coleman et al, *Object-Oriented Development - The Fusion Method*, Prentice-Hall, 1993.

[Cooke, 1994]. M. Cooke, *Language and Reason: A Study of Habermas's Pragmatics*, MIT Press, Cambridge, Mass., 1994.

[CoopIS, 1994]. Proceedings of the Second International Conference on Cooperative Information Systems (CoopIS-94), Toronto, Canada, May 17-20, 1994.

[Covington, 1996]. M.A. Covington, "Toward a new type of language for electronic commerce", in: Proc. of the 29th Annual Hawaii Int.l. Conf. on System Sciences (HICSS), IEEE Computer Society Press, pp. 329-336, 1996.

[CSCW, 1995]. Continuation of the debate on the Language/Action Perspective, in: Computer Supported Cooperative Work (CSCW), Vol. 3, no. 2, pp. 29-95, 1995.

[Date, 1990]. C. Date, *An Introduction to Database Systems, vol. 1, 5th Ed.*, Addison-Wesley, 1990.

[Davenport, 1993]. T.H. Davenport, *Process Innovation*, Harvard Business School Press, Boston, MA., 1993.

[Davenport, 1995]. T.H. Davenport, "Business Process Reengineering: Where it's been, where it's going", in: *Business Process Change: Reengineering Concepts, Methods and Technologies*, V. Grover and W.J. Kettinger (eds.), Idea Group Publishing, Harrisburg PA., pp. 1-13, 1995.

[Davenport and Stoddard, 1994]. T.H. Davenport and D.B. Stoddard, "Reengineering: Business change of mythic proportions ?", in: MIS Quarterly, June, pp. 121-127, 1994.

[Davies and Edwards, 1994]. W.H.E. Davies and P. Edwards, "Agent-K: an integration of AOP and KQML", available as URL: http://www.csd.abdn.ac.uk/~wdavies/Publications/CIKM94/agentk.html

[Davis and Olson, 1984]. G.B. Davis and M.H. Olson, *Management Information Systems: Conceptual foundations, structure and development*, McGraw-Hill, New York, 1984.

[De Cindio et al., 1986]. F. De Cindio, G. De Michelis, C. Simone, R. Vassalo and A. Zanaboni, "CHAOS as a coordinating technology", in: Proc. of the 1st Conf. on Computer Supported Cooperative Work (CSCW'86), 1986.

[De Michelis and Grasso, 1994]. G. De Michelis and M.A. Grasso, "Situating Conversations within the Language/Action Perspective: The Milan Conversation Model," in: Proc. of the 5th Conf. on Computer Supported Cooperative Work (CSCW'94), ACM Press, New York, pp. 89-100, 1994.

[De Michelis et al., 1997]. G. De Michelis, E. Dubois, M. Jarke, F. Matthes, J. Mylopoulos, K. Pohl, J. Schmidt, C. Woo and E. Yu, "Cooperative Information Systems: A Manifesto", in: *Cooperative Information Systems: Trends & Directions*, M.P.Papazoglou, G.Schlageter (eds.), Academic-Press, Sept., 1997. To appear.

[Dennett, 1981]. D.C. Dennett, *Brainstorms*, Harvester Press, 1981.

[Dennett, 1987]. D.C. Dennett, *The Intentional Stance*, MIT Press, Cambridge, MA., 1987.

[Denning and Medina-Mora, 1995]. P.J. Denning and R. Medina-Mora, "Completing the loops", in: Interfaces, vol. 25, no. 3, pp. 42-57, 1995.

[De Troyer, 1991]. O.M.F. De Troyer, "The OO-Binary Relationship Model: A Truly Object-Oriented Conceptual Model", in: Proc. of Advanced Information Systems Engineering (CAISE'91), R. Andersen, J.A. Bubenko jr. A. Sølvberg (eds.), LNCS 498, Springer-Verlag, Berlin, 1991.

[De Troyer and Meersman, 1995]. O.M.F. De Troyer and R. Meersman, "A logic framework for a semantics of object-oriented data modelling", in: Proc. of 14th Object-Oriented and Entity-Relationship Modelling (OOER'95), Gold Coast, Australia, LNCS 1021, Springer Verlag, Berlin, pp. 238-249, 1995.

[De Troyer et al., 1993]. O.M.F. De Troyer, E. Verharen, H. Weigand, "Modelling information systems dynamics", in: Proc. of the Int.l ISCORE'93 WS on Inf. Syst.- Correctness and Reusability, Hannover, sept.'93, U. Lipeck and G. Koschorreck (eds.), Informatik-Berichte 01/93, Univ. Hannover, Hannover, 1993.

[Dewitz, 1991]. S.D. Dewitz, "Contracting on a performative network: using information technology as a legal intermediary", in: Proc. of Collaborative Work, Social Communications and Information Systems, R. Stamper et al (eds.), Elsevier Science Publ., North-Holland, 1991.

[Dietz, 1990a]. J.L.G. Dietz, "A communication-oriented approach to conceptual modelling of information systems", in: Proc. of Advanced Information Systems Engineering (CAISE'90), B. Steinholtz, A. Sølvberg and L. Bergman (eds.), LNCS 436, Springer-Verlag, Berlin, 1990.

[Dietz, 1990b]. J.L.G. Dietz, "A communication-oriented approach to conceptual systems modeling" in: Proc. of the 1990 IFIP Working Conference on Dynamic Modelling of Information Systems, Noordwijkerhout, pp. 37-60, 1990.

[Dietz, 1992a]. J.L.G. Dietz, "Modelling Communication in Organizations", in: Proc. of Linguistic Instruments in Knowledge Engineering, R.P. Van de Riet and R. Meersman (eds.), North-Holland, 1992.

[Dietz, 1992b]. J.L.G. Dietz, *Leerboek Informatiekundige Analyse*, Kluwer Bedrijfswetenschappen, Deventer, 1992. (in Dutch)

[Dietz, 1994a]. J.L.G. Dietz, "Business Modelling for Business Redesign", in: Proc. of 27th Hawaii Int.l. conf. on System Sciences (HICSS), IEEE Computer Society Press, pp. 723-732, 1994.

[Dietz, 1994b]. J.L.G. Dietz, "Modeling Business Processes for the Purpose of Redesign", in: Proc. of IFIP TC8 Open Conference on Business Process Redesign, B.C. Glasson, I.T. Hawryszkiewycs, B.A. Underwood, R.A. Weber (eds.), Elsevier Science Publ., North-Holland, pp. 249-258, 1994.

[Dietz and Mulder, 1996]. J.L.G. Dietz and H.B.F. Mulder, "Realising strategic management reengineering objectives with DEMO", in: [Dignum et al., 1996c].

[Dietz and Widdershoven, 1991]. J.L.G. Dietz and G.A.M. Widdershoven, "Speech acts or communicative action?", in: Proc. of 2nd European Conf. on Computer Supported Cooperative Work (ECSCW'91), L. Bannon, M. Robinson, K. Schmidt (eds.), Kluwer, Dordrecht, 1991.

[Dietz and Widdershoven, 1992]. J.L.G. Dietz and G.A.M. Widdershoven, "A comparison of the linguistic theories of Searle and Habermas as a basis for communication supporting systems", in: Proc. of Linguistic Instruments in Knowledge Engineering, R.P. van de Riet and R.A. Meersman (eds.), North-Holland, 1992.

[Dietz et al., 1996]. J.L.G. Dietz, N.B.J. van der Rijst, F.L.H. Stollman, "The specification and implementation of a DEMO supporting CASE-tool", in: [Dignum et al., 1996c].

[Dignum, 1989]. F.P.M. Dignum, *A language for modelling knowledge bases*, Ph.D. Thesis, Vrije Universiteit (Free University), Amsterdam, 1989.

[Dignum, 1996]. F.P.M. Dignum, "Autonomous agents and social norms", in: Proc. of the Int.l. Conf. on Multi-Agent Systems (ICMAS'96) Workshop on Norms, Obligations and Conventions, Tokyo, Japan, dec., 1996 (to be published).

[Dignum, 1997]. F.P.M. Dignum, "Social interactions of autonomous agents; private and global views on communication", in: Proc. of 3rd workshop of the ModelAge Project, P.-Y. Schobbens (ed.), Sienna, Italy, January, 1997 (to be published).

[Dignum and Kuiper, 1997]. F.P.M. Dignum and R. Kuiper, "Combining Dynamic Deontic Logic and Temporal Logic for the Specification of Deadlines", Proc. of 30th Hawaii Int.l. conf. on System Sciences (HICSS), IEEE Computer Society Press Hawaii, January 6-10 1997. (to be published)

[Dignum and Meyer, 1990]. F. Dignum and J.-J.Ch. Meyer., "Negations of transactions and their use in the specification of dynamic and deontic integrity constraints", in: *Semantics for Concurrency*, M. Kwiatkowska, M.W. Shields, and R.M. Thomas (eds.), Springer-Verlag, Berlin, pp. 61-80, 1990.

[Dignum and Weigand, 1995a]. F. Dignum and H. Weigand, "Communication and Deontic Logic", in: Information Systems, Correctness and Reusability, Proc. of ISCORE-94 Workshop, R. Wieringa and R. Feenstra (eds.), World Scientific, Singapore, pp. 242-260, 1995.

[Dignum and Weigand, 1995b]. F. Dignum and H. Weigand, "Modelling Communication between Cooperative Systems", in: Proc. of Advanced Information Systems Engineering (CAISE'95), J. Iivari, K. Lyytinen, M. Rossi (eds.), LNCS-932, Springer-Verlag, Berlin, pp. 140-153, 1995.

[Dignum and van Linder, 1996]. F.Dignum and B. van Linder, "Modelling Rational Agents in a Dynamic Environment: Putting Humpty Dumpty Together Again", in: Proc. of 2nd workshop of the ModelAge Project, J.L. Fiadeiro and P.-Y. Schobbens (eds.), Sesimbra, Portugal, January 1996, pp. 81-92, 1996.

[Dignum et al., 1994]. F.P.M. Dignum, J.-J.Ch. Meyer and R. Wieringa, "Contextual permission: a solution to the free choice paradox", in: Proc. of 2nd Int.l WS on Deontic Logic in Computer Science, A. Jones and M. Sergot (eds.), Tano A.S., Oslo, pp. 107-135, 1994.

[Dignum et al., 1996a]. F. Dignum, H. Weigand and E. Verharen, "Meeting the deadline: on the formal specification of temporal deontic constraints", in: Proc. of the Int.l. Symposium on Methodologies for Intelligent Systems (ISMIS'96): Foundations of Intelligent Systems, Z.W. Ras and M. Michalewicz (eds.), LNAI 1079, Springer-Verlag, Berlin, pp. 243-252, 1996.

[Dignum et al., 1996b]. F. Dignum, J.-J.Ch. Meyer, R. Wieringa and R. Kuiper, "A Modal approach to intentions commitments and obligations: Intention plus commitment yields obligation", in: *Deontic Logic, Agency and Normative Systems*, M. Brown and J. Carmo (eds.), Springer-Verlag, Berlin, pp. 80-97, 1996.

[Dignum et al., 1996c]. F. Dignum, J. Dietz, E. Verharen, H. Weigand (eds.), Proc. of 1st Int.l Workshop on Communication Modelling - The language/action perspective 1996, Electronic Workshop in Computing Series, Springer-Verlag, Berlin, 1996. (also pre-proceedings, EIT Report 96-01, Tilburg University, 1996).

[Dik, 1978]. S.C. Dik, *Functional Grammar,* North-Holland, Amsterdam, 1978.

[Dik, 1989]. S.C. Dik, *Theory of Functional Grammar*, Foris, Dordrecht, 1989.

[Dobson et al., 1991]. J.E. Dobson, M.J. Martin, C.W. Olphert and S.E. Powrie, "Determining requirements for CSCW: The ORDIT Approach", in: proc. of Collaborative Work, Social Communications and Information Systems, R. Stamper, P. Kerola, R. Lee and K. Lyytinen (eds.), IFIP, Elsevier Science Publ., North-Holland, 1991.

[Doyle, 1988]. R. Doyle, "Artificial intelligence and rational self-government", Technical Report CMU-CS-88-124, Comp. Science Dep., Carnegie-Mellon University, Pittsburgh, PA., 1988.

[Durfee et al., 1989]. E.H. Durfee, V.R. Lesser, D.D Corkill, "Trends in cooperative distributed problem solving", in: IEEE Transactions on Knowledge and Data Engineering, vol. 1, no. 1, p. 63-68, 1989.

[van Eck, 1982]. J.A. van Eck, "A system of temporally relative modal and deontic predicate logic and its philosophical applications", in: Logique et Analyse 100, pp. 249-381, 1982.

[Edmonds, 1992]. E. Edmonds, *The separable user interface*, Academic Press, London, 1992.

[Elgesem, 1993]. D. Elgesem, *Action Theory and Modal Logic*, Ph.D. Thesis, Institute for Philosophy, University of Oslo, Oslo, Norway, 1993.

[Elmagarmid et al., 1990]. A.K. Elmagarmid, Y. Leu, W. Litwin, and M. Rusinkiewicz, "A Multidatabase Transaction Model for InterBase", in: Proc. of 16th Int.l Conf. on Very Large Databases (VLDB'90), Brisbane, Morgan Kaufmann, Los Altos, CA., pp. 507-518, 1990.

[Elmagarmid,1992] A. Elmagarmid (ed.), *Database Transaction Models for Advanced Applications*, Morgan Kaufman, 1992.

[Emerson, 1989]. E.A. Emerson, "Temporal and Modal Logic", in: *Handbook of Theoretical Computer Science*, J. van Leeuwen (ed.), North-Holland, Amsterdam, pp. 995-1072, 1989.

[Eriksson, 1996]. O. Eriksson, "A communicative action analysis of information systems: a sales support system and its effects", in: [Dignum et al., 1996c].

[Etzioni et al., 1994]. O. Etzioni, N. Lesh and R. Segal, "Building softbots for UNIX", in: Software agents - Papers from the 1994 Spring Symposium (Technical Report SS-94-03), O. Etzioni (ed.), AAAI Press, Menlo Park, CA., pp. 9-16, 1994.

[Ferguson, 1992a]. I.A. Ferguson, *TouringMachines: An architecture for dynamic, rational, mobile agents*, Ph.D. Thesis, Clare Hall, University of Cambridge, UK, 1992. (also Technical Report No. 273, University of Cambridge Computer Laboratory)

[Ferguson, 1992b]. I.A. Ferguson, "Towards an architecture for adaptive, rational, mobile agents", in: Decentralized AI 3 - Proc. of the 3rd European WS on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW'91), E. Werner and Y. Demazeau (eds.), Elsevier Science Publ., North Holland, pp. 249-262, 1992.

[Fiadeiro and Maibaum, 1991]. J. Fiadeiro and T. Maibaum, "Temporal Reasoning over Deontic Specification", in: Journal of Logic and Computation 1 (3), 1991.

[Finin et al., 1993]. T. Finin, D. McKay, R. Fritzson, R. McEntire, "KQML: an information and knowledge exhange protocol", in: Proc. of Int.l Conf. on Building and Sharing of Very Large-Scale Knowledge Bases, dec., 1993. (Also available as: "KQML: an information and knowledge exhange protocol", in: *Knowledge building and knowledge sharing*, K. Fuchi and T. Yokoi (eds.), Ohmsha and IOS Press, 1994; and as URL http://www.cs.umbc.edu/kqml/papers/kbks.ps)

[Finin et al., 1994]. T. Finin, R. Fritzson, D. McKay, R. McEntire, "KQML: a language and protocol for knowledge and information exchange", in: Distributed AI - Papers from the 13th Int.l. WS, M. Klein and K. Sharma (eds.), AAAI Press, Menlo Park, CA., pp. 93-103, 1994. (Technical Report WS-94-02)

[Flach, 1995]. P.A. Flach, *Conjectures, an inquiry concerning the logic of induction*, PhD Thesis, Tilburg University, 1995.

[Flores and Ludlow, 1980]. F. Flores and J.J. Ludlow, "Doing and Speaking in the Office", in: *Decision Support Systems: Issues and Challenges*, G. Fick, H. Spraque Jr. (Eds.)., Pergamon Press, New York, pp. 95-118, 1980.

[Flores et al., 1988]. F. Flores, M.Graves, B.Hartfield,and T.Winograd, "Computer Systems and the Design of Organizational Interaction", in: ACM Trans. on Information Systems (TOIS), Vol.6, No.2, 1988.

[Galbraith, 1973]. J.R. Galbraith, *Designing Complex Organizations*, Addison-Wesley, Reading MA, 1973.

[Galbraith, 1977]. J.R. Galbraith, *Organization Design*, Addison-Wesley, Reading MA, 1977.

[Galliers, 1988]. J.R. Galliers, *A theoretical framework for computer models of cooperative dialogue, acknowledging multi-agent conflict*, Ph.D. Thesis, Open University, UK, 1988.

[Garcia-Molina et al., 1990]. Garcia-Molina, H., D Gawlick, J. Klein, K. Kleissner, K. Salem, "Coordinating multi-transaction activities", Technical Report CS-TR-247-90, Princeton University, Dept of Computer Science, Feb 1990.

[Gasparotti and Simone, 1990]. P. Gasparotti and C. Simone, "A user defined environment for handling conversations", in: *Multi-User Interfaces and Applications*, S. Gibbs and A.A. Verrijn-Stuart (eds.), IFIP, Elsevier Science Publ., North-Holland, pp. 271-289, 1990.

[Geanakoplos, 1988]. J. Geanakoplos, "Common knowledge, Bayesian learning, and market speculation with bounded rationality", Memo, Yale Universit, New Haven, CT., 1988.

[Genesereth and Ketchpel, 1994]. M.R. Genesereth and S.P. Ketchpel, "Software Agents", in: Communications of the ACM (CACM), vol. 37, no. 7, pp. 48-53, 1994.

[Genesereth and Fikes, 1992]. M.R. Genesereth and R. Fikes, "Knowledge Interchange Format, v. 3.0, Reference Manual", Technical Report Logic-92-1, Computer Science Department, Stanford University, June, 1992.

[Genesereth and Nilsson, 1987]. M.R. Genesereth and N. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann Publ., San Mateo, CA., 1987.

[Georgakopoulos and Hornick, 1994]. D. Georgakopoulos and M.F. Hornick, "A framework for enforcable specification of extended transaction models and transactional workflows", in: Int.l. Journal of Intelligent and Cooperative Information Systems, 3 (3), pp. 225-253, 1994.

[Georgakopoulos et al., 1994]. D. Georgakopoulos et al., "Specification and management of extended transactions in a programmable transaction environment", in: Proc. of the 10th Data Engineering Conf., Houston, TX., feb., 1994.

[Georgeff and Lansky, 1987]. M.P. Georgeff and A.L. Lansky, "Reactive reasoning and planning", in: Proc. of 6th Nat.l Conf. on AI (AAAI'87), Seattle, WA., AAAI Press, Menlo Park, CA., pp. 677-682, 1987.

[Glasgow et al., 1989]. J. Glasgow, G. MacEwen, P. Panangaden, "Security by permission in databases", in Database Security II: Status and Prospects, Results of the IFIP WG 11.3 WS on Database Security, C.E. Landwehr (ed.), Kingston, Ontario, Canada, 1989.

[Gmytrasiewicz and Durfee, 1993]. P.J. Gmytrasiewicz and E.H. Durfee, "Reasoning about other agents: philosophy, theory and implementation", in: Proc. of the 12th Int.l. Workshop on Distributed AI, AAAI-Press, 1993.

[Goldkuhl, 1992]. G. Goldkuhl, "Contextual activity modelling of information systems", in: Proc. of 3rd Int.l WC on Dynamic Modelling of Information Systems, H.G. Sol (ed.), IFIP, Delft University of Technology, Delft, 1992.

[Goldkuhl, 1993]. G. Goldkuhl, "Verksamhetsutveckla Datasystem", Intention, Linköping, 1993. (in Swedish)

[Goldkuhl, 1995]. G. Goldkuhl, "Information as Action and Communication", in: The Infological Equation, *Essays in honour of B. Langefors,* B. Dahlbohm (ed.), Gothenburg Studies in Information Systems, Gothenburg Univ., 1995. (Also: Lingkoping Univ. Report LiTH-IDA-R-95-09)

[Goldkuhl, 1996]. G. Goldkuhl, "Generic business frameworks and action modelling", in: [Dignum et al., 1996c].

[Goldkuhl and Röstlinger, 1988]. G. Goldkuhl and A. Röstlinger, *Förändringsanalys,* Studentlitteratur, Lund, 1988.

[Goodwin, 1993]. R. Goodwin, "Formalizing Properties of Agetns", Report CMU-CS-93-159, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, may 1993.

[Gray, 1981]. J.N. Gray, "The transaction concept: Virtues and limitations", in: Proc. of 7th Int.l Conf. on Very Large Data Bases (VLDB'81), Cannes, France, pp. 144-154, 1981.

[Grossman and Ege, 1987]. M. Grossman and R. Ege, "Logical composition of object-oriented interfaces", in: Proc. of OOPSLA'87, ACM Press, New York, pp.295-306, 1987.

[Guarino, 1992]. N. Guarino, "Concepts, Attributes, and Arbitrary Relations: Some Linguistic and Ontological Criteria for Structuring Knowledge Bases", in: Proc. of Linguistic Instruments in Knowledge Engineering, R. v.d. Riet and R.A. Meersman (eds), North-Holland, Amsterdam, pp.195-211, 1992.

[Habermas, 1981]. J. Habermas, *Theorie des kommunikativen Handelns*, Erster Band, Suhrkamp Verlag, Frankfurt am Main, 1981. (in German)

[Habermas, 1984]. J. Habermas, *The Theory of Communicative Action: Reason and the Rationalization of Society, Volume One*, Beacon Press, Boston, 1984.

[Habermas, 1987]. J. Habermas, *The Theory of Communicative Action: Lifeworld and System: A critique of functionalist reason*, Beacon Press, Boston, 1987.

[Habermas, 1991]. J. Habermas, "Comments on John Searle: Meaning, Communication and Representation", in: *John Searle and his Critics*, E. Lepore, R. Van Gulick (Eds.), Blackwell, Cambridge MA, pp. 17-31, 1991. (Searle's reply to Habermas, pp. 89-96)

[Haddadi, 1995]. A. Haddadi, *Communication and Cooperation in Agent Systems: A pragmatic theory*, LNCS 1056, Springer-Verlag, Berlin, 1995.

[Haerder and Reuter, 1983]. T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery", in ACM Computing Surveys, 15 (4), pp. 287-317, 1983.

[Hammer, 1990]. M. Hammer, "Reengineering work: don't automate, obliterate" in: Harvard Business Review, july-august 1990, pp. 104-112, 1990.

[Hammer and Champy, 1993]. M. Hammer and J.A. Champy, *Reengineering the corporation: A manifesto for business revolution*, Nicholas Brealy, London, 1993.

[Harel, 1979]. D. Harel, "First Order Dynamic Logic, in: LNCS 68, Springer-Verlag, Berlin, 1979.

[Harel, 1984]. D. Harel, "Dynamic Logic", in: *Handbook of Philosophical Logic, Vol. II - Extensions of Classical Logic*, D. Gabbay and F. Guenther (eds.), Synthese Lib. Vol. 164, Reidel, Dordrecht, pp. 497-604, 1984.

[Hartson and Dix, 1989]. H.R. Hartson and D. Hix, "Human-Computer Interface Development", in: ACM Computing Surveys 21, 1, p.5-92, 1989.

[Helander, 1988]. M. Helander, *Handbook of human-computer interaction*, North-Holland, 1988.

[Helm et al., 1990]. R. Helm, I.M. Holland and D. Gangopadhyay, "Contracts: Specifying Behavioral Compositions in Object-Oriented Systems", in: proc. of OOPSLA/ECOOP'90, Conf. on Object-Oriented Programming: Systems, Languages, and Applications/ European Conference on Object-Oriented Programming, N. Meyrowitz (ed.), ACM Sigplan Notices vol.25, no.10, oct. 1990, ACM Press, New York, NY. 1990.

[Hengeveld, 1988]. K. Hengeveld, "Illocution, mood, and modality in a Functional Grammar of Spanish", in: Journal of Semantics, vol. 6, pp 227-269, 1988.

[Hengeveld, 1990]. K. Hengeveld, "The hierarchical structure of utterances", in: *Layers and Levels of Representation in Language Theory: a functional view*, J. Nuyts, A.M. Bolkenstein, C. Vet (eds.), John Benjamins, Amsterdam/Philadelphia, 1990.

[Herrestad and Krogh, 1995]. H. Herrestad and C. Krogh, "Deontic logic relativised to bearers and counterparties", in: *25 Years Anniversary Anthology NRCCL*, J. Bing and O. Torvund (eds.), Tano Publ., CompLex Series, Oslo, 1995.

[Hirscheim, 1985]. R.A. Hirscheim, *Office Automation: A social and organizational perspective*, John Wiley and Sons, Chichester, 1985.

[Holm, 1994]. P. Holm, "The COMMODIOUS Method - COMmunication MODelling as an Aid to Illustrate the Organizationsl Use of Software", in: Proc. of 6th Int.l Conf. on Software Engineering and Knowledge Engineering, Jurmala, Latvia, 1994.

[Holm, 1996]. P. Holm, *On the design and usage of information technology and the structuring of communication and work*, Ph.D. Thesis, Stockholm University, april 1996.

[Hooff, 1995]. B. van den Hooff, "For what it's worth: de waarde van communicatietechnologieën voor organisaties", in: Informatie en Informatiebeleid, vol. 13, no.2, pp. 34-43, 1995. (in Dutch)

[Horty, 1996]. J.F. Horty, "Combining agency and obligation", in: Deontic Logic, Agency and Normative Systems, M. Brown and J. Carmo (eds.), Springer-Verlag, Berlin, pp. 98-122, 1996.

[Huhns et al., 1992]. M.N. Huhns, N. Jacobs, T. Ksiezyk, W.M. Shen, M.P. Singh and P.E. Cannata, "Integrating enterprise information models in Carnot", in: Proc of Int.l Conf on Intelligent and Cooperative Information Systems (CoopIS'92), Rotterdam, The Netherlands, pp. 32-42, 1992.

[ISO, 1982]. J.J. v. Griethuysen (ed.), Concepts and terminology for the conceptual schema and the information base, Report ISO/TC97/SC5-N695, ISO, 1982.

[ISO, 1984]. J.J.van Griethuysen (ed.), Concepts and Terminology for the Conceptual Schema and the Information Base, ISO/TC97/SC21 N197, ISO, 1984.

[ISO, 1989]. ISO/DP 10026-1,2,3. *Information Processing Systems*, Open Systems Interconnection, Distributed Transaction Processing, 1989.

[Jacobson, 1987]. I. Jacobson, "Object-Oriented development in an industrial environment", in: proc. of OOPSLA'87, Conference on Object-Oriented Programming: Systems, Languages, and Applications, N. Meyrowitz (ed.), ACM Sigplan Notices vol.22, no.12, dec. 1987, ACM Press, New York, NY., pp. 183 - 191, 1987.

[Jacobson et al., 1992]. I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard, *Object-Oriented Software Engineering: a use-case driven approach*, ACM Press, Addison-Wesley, 1992.

[Janson and Woo, 1992]. M.A. Janson and C.C. Woo, "Investigating information and knowledge gathering methods: A speech act lexicon perspective", in: *Information Systems Concepts: Improving the understanding*, E.D. Falkenberg, C. Rolland and E.N. El-Sayed (eds.), Elsevier Science Publ., North-Holland, 1992.

[Janson and Woo, 1995]. M.A. Janson and C.C. Woo, "Comparing IS Development Tools and Methods: Using the Speech Act Theory", in: Information and Management. Vol. 28, pp. 1-12, 1995.

[Jennings, 1993a]. N.R. Jennings, "Commitments and conventions: The foundation of coordination in multi-agent systems", in: Knowledge Engineering Review, vol. 8, no. 3, pp. 223-250, 1993.

[Jennings, 1993b]. N.R. Jennings, "Specification and implementation of a belief desire joint-intention architecture for collaborative problem solving", in: Journal of Intelligent and Cooperative Information Systems, vol. 2, no. 3, pp. 289-318, 1993.

[Jennings, 1995]. N.R. Jennings, "Controlling cooperative problem solving in industrial multi-agent systems using joint intentions", in: Artificial Intelligence 75 (2), pp. 195-240, 1995.

[Jennings, 1996]. N.R. Jennings, "Coordination techniques for Distributed Artificial Intelligence", in: *Foundations of Distributed Artificial Intelligence*, G. O'Hare and N. Jennings (eds.), John Wiley and Sons, New York, pp. 187-210, 1996.

[Jennings et al., 1992]. N.R. Jennings, E.H. Mamdani, I. Laresgoiti, J. Perez, J. Corera, "GRATE: A General Framework for Cooperative Problem Solving", in: IEE-BCS Journal of Intelligent Systems Engineering, 1 (2), pp. 102-114, 1992.

[Jennings et al., 1996a]. N.R. Jennings, P. Faratin, M.J. Johnson, P. O'Brien, M.E. Wiegand, "Using intelligent agents to manage business processes", in: Proc of 1st Int.l Conf. on the Practical Application of intelligent Agents and Multi-agent technology (PAAM'96), B. Crabtree and N.R. Jennings (eds.), The Practical Application Company, Lancashire, pp. 345-360, 1996.

[Jennings et al, 1996b]. N.R. Jennings, J. Corera, I. Laresgoiti, E.H. Mamdani, F. Perriolat, P. Skarek, L.Z. Varga, "Using ARCHON to develop real-world DAI applications for electricity transportation management and particle accelerator control", in: IEEE Expert,december, 1996.

[Jensen, 1987]. K. Jensen, "Coloured Petri nets", in: Petri nets: central models and their properties (LNCS 188), G. Rosenberg (ed.), Springer-Verlag, Berlin, pp. 248-299, 1987.

[Johannesson, 1995]. P. Johannesson, "Representation and communication - a speech act based approach to information systems design", in: Information Systems, vol.20, no.4, pp. 291-303, 1995.

[Jones and Sergot, 1993]. A.J.I. Jones and M. Sergot, "On the characterization of law and computer systems: the normative systems perspective", in: *Deontic Logic in Computer Science*, J.-J.Ch. Meyer and R. J. Wieringa (eds.), John Wiley and Sons Ltd., Chichester, 1993.

[Jones et al., 1979]. S. Jones, P. Mason, R. Stamper, "LEGOL 2.0: a relational specification language for complex rules", in: Information Systems 4, pp. 157-169, 1979.

[Kaelbling, 1991]. L.P. Kaelbling, "A situated automata approach to the design of embedded agents", in: SIGART Bulletin, vol. 2, no. 4, pp. 85-88, 1991.

[Kaelbling and Rosenschein, 1990]. L.P. Kaelbling and S.J. Rosenschein, "Action and planning in embedded agents", in: *Designing autonomous agents*, P. Maes (ed.), MIT Press, Cambridge, MA., pp. 35-48, 1990.

[Kaplan et al., 1992]. S. Kaplan, W. Tolone, D. Bogia, C. Bignoli, "Flexible, active support for collaborative work with ConversationBuilder", in: Proc. of 4th Conf. on Computer Supported Cooperative Work (CSCW'92), ACM Press, New York, NY., pp. 378-385, 1992.

[Keen, 1991]. P.G.W. Keen, *Shaping the future: Business design through information technology*, Harvard Business School Press, Boston, MA., 1991.

[Kensing and Winograd, 1991]. F. Kensing and T. Winograd, "The language/action approach to design of computer-suppor for cooperative work: A preliminary study in work mapping", in: *Collaborative Work, Social Communications and Information Systems*, R.K. Stamper, P. Kerola and K. Lyytinen (eds.), Elsevier Science Publ., North-Holland, 1991.

[Khosla, 1988]. S. Khosla, *System Specification: A deontic approach*, Ph.D. Thesis, Imperial College, London, 1988.

[Khosla and Maibaum, 1987]. S. Khosla and T.S.E. Maibaum, "The prescription and description of state based systems", in: *Temporal logic in specification*, B. Banieqbal, H. Barringer, A. Pnueli (eds.), LNCS-398, Springer-Verlag, Berlin, 1987.

[Kieronska, 1991]. D.H. Kieronska, *A system for the synthesis of concurrent programs: an algorithmic approach to state graph constructions and transformations,* Technical Report, PhD Thesis, Department of Computer Science, University of Western Australia, 1991.

[Kimbrough and Lee, 1996]. S.O. Kimbrough and R.M. Lee, "On formal aspects of Electronic (or Digital) Commerce: examples of research issues and challenges", in: Proc. of the 29th Annual Hawaii Int.l Conf. on System Sciences (HICSS), IEEE Computer Society Press, pp. 319-328, 1996.

[Kimbrough et al., 1984]. S.O. Kimbrough, R.M. Lee and D. Ness, "Performative, informative, and emotive systems: The first piece of the PIE", in: Proc. of 5th Int.l Conf. on Information Systems (ICIS'84), L. Maggi, J.L. King, K.L. Kraenens (eds.), pp. 141-148, 1984.

[King and Novak, 1989]. R. King and M. Novak, "FaceKit: a database interface design toolkit", in: Proc. of Very Large Databases (VLDB'89), Morgan Kaufmann, Los Altos, CA., 1989.

[Kinny et al., 1992]. D. Kinny, M. Ljungberg, A.S. Rao, E. Sonenberg, G. Tidhar, E. Werner, "Planned team activity", in: Artificial Social Systems - Selected papers from the 4th European WS on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW'92), C. Castelfranchi and E. Werner (eds.), LNAI 830, Springer-Verlag, Heidelberg, pp. 226-256, 1992.

[Kiss, 1992]. G. Kiss, "Variable coupling of agents to their environment: Combining situated and symbolic automata", in: Decentralized AI - 3, E. Werner and Y. Demazeau (eds.), Elsevier Science Publ., North-Holland, pp. 231-248, 1992.

[Klusch, 1994]. M. Klusch, "Using a cooperative agent system for a context-based recognition of interdatabase dependencies", in: Proc. CIKM-94 WS on Intelligent Information Agents, Gaithersburg, 1994.

[Klusch, 1995]. M. Klusch, "Cooperative recognition of interdatabase dependencies", ACM SIGMOD Proc. 2. Int. WS on Advances in Databases and Information Systems, Moscow, ACM Press, New York, NY., 1995.

[Klusch, 1996]. M. Klusch, "Utilitarian coalition formation between information agents for a cooperative discovery of interdatabase dependencies", in: *Cooperative Knowledge Processing*, S. Kirn, G. O'Hare (eds.), Springer Verlag, London, 1996.

[Klusch and Shehory, 1996]. M. Klusch and O. Shehory, "Coalition formation among rational information agents", in: Proc. of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World MAAMAW-96, Eindhoven (Netherlands), W. van de Velde, J. Perram (eds.), LNAI 1038, pp204-217, Springer Verlag, Berlin 1996.

also available as:URL: http://www.informatik.uni-kiel.de/~mkl/papers/f-maamaw96.ps

[Kowalski, 1995]. R.A. Kowalski, "Using meta-logic to reconcile reactive with rational agents", in: *Meta-Logics and Logic Programming*, MIT Press, 1995.

[Kraus, 1993]. S. Kraus, "Agents contracting tasks in non-collaborative environments", in: Proc of Nat.l Conf. on AI (AAAI'93), Washington, DC., AAAI Press, Menlo Park, CA., 1993.

[Kühn et al., 1992]. E. Kühn, F. Puntigam, A.K. Elmagarmid, "Multidatabase transaction and query processing in logic", in: *Database Transaction Models for advanced applications*, A. Elmagarmid (ed), Morgan Kaufman, San Mateo, CA., 1992.

[Kyng, 1995]. M. Kyng, "Making representations work", in: Communications of the ACM (CACM), vol. 38, no. 9, pp. 46- , 1995.

[Langefors, 1966]. B. Langefors, *Theoretical analysis of information systems*, Studentlitteratur, Lund, 1966.

[Langefors, 1993]. B. Langefors, *Essays on infology*, Dep. of Information Systems, University of Göteborg, Göteborg, 1993.

[Langefors and Samuelson, 1976]. B. Langefors and K. Samuelson, *Information and Data in Systems*, Petrocelli/Charter, New York, 1976.

[Lee, 1985]. R.M. Lee, "Bureaucracies as artificial intelligence", in: Knowledge Representation for Decision Support Systems, Proc. of IFIP WG 8.3 Working Conference, L.B. Methlie, R.H. Sprague jr. (eds.), North-Holland, Amsterdam, 1985.

[Lee, 1988a]. R.M. Lee, "Bureaucracies as deontic systems", ACM Transactions on Office Information Systems (TOIS), 6 (2), p.87-108, 1988.

[Lee, 1988b]. R.M. Lee, "A logic model for electronic contracting", in: Decision Support Systems 4, pp. 27-44, 1988.

[Lee, 1996]. R.M. Lee, *Contract Grammars: Computable contract procedures for electronic contracting*, Euridis Report no. WP 96.06.01, Erasmus University, Rotterdam, 1996.

[Lehtinen and Lyytinen, 1986]. E. Lehtinen, K. Lyytinen, "Action Based Model of Information Systems", in: Information Systems, vol. 11, no. 4, pp. 299-317, 1986.

[Lepore and Van Gulick, 1991]. E. Lepore and R. Van Gulick (eds.), *John Searle and his critics*, Basil Blackwell, Cambridge, Mass., 1991.

[Levesque et al., 1990]. H.J. Levesque, P.R. Cohen and J.H.T. Numes, "On acting together", in: Proc. of 8th Nat.l Conf. on AI (AAAI'90), Boston, MA., AAAI Press, Menlo Park, CA., pp. 94-99, 1990.

[Levinson, 1983]. S.C. Levinson, *Pragmatics*, Cambridge University Press, Cambridge, MA., 1983.

[van Linder, 1996]. B. van Linder, *Modal Logics for Rational Agents*, Ph.D. Thesis, Utrecht University, Utrecht, 1996.

[Ljungberg and Holm, 1996]. J. Ljungberg and P. Holm, "Speech acts on trial", in: *Computers in Context*, L. Mattiassen and M. Kyng (eds.), MIT Press, Cambridge, MA., 1996.

[Luhmann, 1985]. N. Luhmann, *The sociology of law*, Routledge, London, 1985.

[Lundeberg et al., 1981]. M. Lundeberg, G. Goldkuhl, A. Nilsson, *Information systems development - A systematic approach*, Prentice-Hall, Englewood Cliffs, NJ., 1981.

[Lynch, 1995]. M. Lynch, "On making explicit", in: Computer Supported Cooperative Work, vol. 3, no. 1, pp. 65-68, 1995.

[Lyytinen, 1985]. K. Lyytinen, "Implications of theories of language for information systems", in: MIS Quarterly, March, pp. 61-74, 1985.

[Lyytinen and Hirschheim, 1988]. K. Lyytinen and R. Hirschheim, "Information Systems as Rational Discourse: an Application of Habermas' Theory of Communicative Action", in: Scandinavian Journal of Management, Bd 4, Nr.1/2, pp.19-30, 1988.

[Maes, 1989]. P. Maes, "The dynamics of action selection", in: Proc. of 11th Int.l Joint Conf. on AI (IJCAI'89), Detroit, MI., pp. 991-997, 1989.

[Maes, 1990]. P. Maes, "Situated agents can have goals", in: *Designing Autonomous Agents*, P. Maes (ed.), MIT Press, Cambridge, MA., pp. 49-70, 1990.

[Maes, 1991]. P. Maes, "The agent network architecture (ANA)", SIGART Bulletin, vol. 2, no. 4, pp. 115-120, 1991.

[Maibaum, 1993]. T. Maibaum, "Temporal reasoning over deontic specifications", in: *Deontic Logic in Computer Science, Normative System Specification*, J.-J.Ch. Meyer and R.J. Wieringa (eds.), John Wiley and Sons Ltd., Chichester, 1993.

[Mally, 1926]. E. Mally, *Grundgesetze des Sollens, Elemente der Logik des Willens*, Leuschner and Lubensky, Graz, 1926.

[Martin, 1989]. J. Martin, *Information Engineering - Introduction*, Prentice-Hall, Englewood Cliffs, NJ., 1989.

[McCarthy, 1979]. J. McCarthy, Ascribing mental qualities to machines, Technical Report Memo 326, Stanford University AI Lab, Stanford, CA., 1979.

[McCarty and Monk, 1994]. J.C. McCarty and A. Monk, "Channels, conversation, cooperation and relevance: all you wanted to know about communication but were afraid to ask", in: Collaborative Computing, vol. 1, no. 1, 1994.

[Medina-Mora et al., 1992]. R. Medina-Mora, T. Winograd, R. Flores and F. Flores, "The Action-Workflow Approach to Workflow Management Technology", in: Proc. of 4th Conf. on Computer Supported Cooperative Work (CSCW'92), J. Turner, R. Kraut (eds.), ACM Press, New York, NY., pp. 281-288, 1992.

[Meyer, 1988]. J.-J.Ch. Meyer, "A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic", in: Notre Dame Journal of Formal Logic 29(1), pp. 109-136, 1988.

[Meyer and Wieringa, 1993]. J.-J.Ch. Meyer and R.J. Wieringa, "Deontic Logic: A concise overview", in: *Deontic Logic in Computer Science, Normative System Specification*, J.-J.Ch. Meyer and R.J. Wieringa (eds.), John Wiley and Sons Ltd., Chichester, 1993.

[Minsky, 1986]. M. Minsky, *The Society of Mind*, Simon and Schuster, New York, 1986.

[Minsky and Lockman, 1985]. N.H. Minsky and A.D. Lockman, "Ensuring integrity by adding obligations to privileges", in: 8th IEEE Int.l Conf. on Software Engineering, pp. 92-102, 1985.

[Mintzberg, 1989]. H. Mintzberg, *Mintzberg on Management: inside our strange world of organizations*, The Free Press, New York, 1989.

[Mizoguchi, 1993]. R. Mizoguchi, "Knowledge Acquisition and ontology", in: Proc. of Conf. on Building and Sharing of very large-scale knowledge bases (KB&KS'93), JIPDEC, Tokyo, 1993.

[Moore, 1980]. R.C. Moore, "Reasoning about knowledge and action", Technical Report 191, SRI International, 1980.

[Moore, 1985]. R.C. Moore, "A formal theory of knowledge and action", in: *Formal theories of the commonsense world*, J.R. Hobbs and R.C. Moore (eds.), Ablex, Norwood, pp. 319-358, 1985.

[Moore, 1990]. R.C. Moore, "A formal theory of knowledge and action", in: *Readings in Planning*, J.F. Allen, J. Hendler, A. Tate (eds.), Morgan Kaufmann Publ., San Mateo, pp. 480-519, 1990.

[Morgan, 1986]. G. Morgan, *Images of Organization*, Sage, Newbury Park, 1986.

[Moulin and Chaib-draa, 1996]. B. Moulin and B. Chaib-draa, "An Overview of Distributed Artificial Intelligence", in: *Foundations of Distributed Artificial Intelligence*, G. O'Hare and N. Jennings (eds.), John Wiley and Sons, New York, pp. 3-55, 1996.

[Moutaouakil, 1991]. A. Moutaouakil, "On representing implicated illocutionary force: grammar or logic", in: Working papers in Functional Grammar, WPFG, no. 40, 1991.

[Müller, 1994]. J.P. Müller, "A conceptual model for agent interaction", in: Proc. of 2nd Int.l WC on Cooperative Knowledge Based Systems (CKBS'94), S.M. Deen (ed.), DAKE Centre, University of Keele, UK, pp. 213-234, 1994.

[Müller, 1996]. H.J. Müller, "Negotiation Principles", in: *Foundations of Distributed Artificial Intelligence*, G. O'Hare and N. Jennings (eds.), John Wiley & Sons, NY., pp. 211-229, 1996.

[Müller and Pischel, 1994]. J.P. Müller and M. Pischel, "Modelling interacting agent in dynamic environments", in: Proc. of 11th European Conf. on AI (ECAI'94), Amsterdam, The Netherlands, pp. 709-713, 1994.

[Müller et al., 1995]. J.P. Müller, M. Pischel and M. Thiel, "Modelling reactive behaviour in vertically layered agent architectures", in: *Intelligent Agents: Theories, Architectures and Languages*, M. Wooldridge and N.R. Jennings (eds.), LNAI 890, Springer Verlag, Heidelberg, pp. 261-276, 1995.

[Murata, 1989]. T. Murata, "Petri Nets: Properties, Analysis and Applications", in: Proc. of IEEE 77 (4), 1989.

[Narazaki et al., 1995]. S. Narazaki, H. Yamamura and N. Yoshida, "Strategies for selecting communication structures in cooperative search", in: Int.l Journal of Cooperative Information Systems (IJCIS), M.P. Papazoglou, G. Schlageter (eds.), vol. 4, no. 4, pp. 405-422, 1995.

[Nash, 1950]. J.F. Nash, "The bargaining problem", in: Econometrica 28, pp. 155-162, 1950.

[Nierstrasz and Papathomas, 1990]. O. Nierstrasz and M. Papathomas, "Viewing objects as patterns of communicating agents", in: Proc. of OOPSLA/ECOOP'90, 1990.

[Nodine et al., 1994]. M.H. Nodine, N. Nakos and S. Zdonik, "Specifying Flexible Tasks in a Multidatabase", in: ACM SIGOIS Bulletin 16 (1), p. 13-17, 1994. (also in Proc. CoopIS-94)

[Ngu, 1990]. A.H.H. Ngu, "Specification and verification of temporal relationships in transaction modelling", in: Information Systems, 15 (2), pp. 257-267, 1990.

[Ngu et al., 1994]. A.H.H. Ngu, R.A. Meersman and H. Weigand, "Specification and verification of communication for interoperable transactions", in: Int.l. Journal of Intelligent and Cooperative Information Systems, vol. 3, no. 1, pp.47-65, 1994.

[Nijssen and Halpin, 1989]. G.M. Nijssen and T.A. Halpin, *Conceptual Schema and Relational Database Design: A Fact-Oriented Approach*, Prentice-Hall, Sidney, Australia, 1989.

[ODP, 1992]. Open Distributed Processing group documents, ISO 10746-1, 10746-2, 10747-3, ISO, Nov. 1992.

[O'Hare and Jennings, 1996]. G.M.P. O'Hare and N.R. Jennings (eds.), Foundations of Distributed Artificial Intelligence, John Wiley and Sons, New York, 1996.

[Oliver, 1996]. J.R. Oliver, "On artificial agents for negotiation in electronic commerce", in: Proc. of the 29th Annual Hawaii Int.l. Conf. on System Sciences (HICSS), IEEE Computer Society Press, pp. 337-345, 1996.

[Olle et al., 1988]. T.W. Olle, J. Hagelstein, I.G. MacDonald, C. Rolland, H.G. Sol, F.J.M. van Assche, A.A. Verrijn-Stuart, *Information Systems Methodologies: A framework for understanding*, Addison-Wesley, Wokingham, 1988.

[Özsu and Valduriez, 1991]. M.T. Özsu and P. Valduriez, *Principles of Distributed Database Systems*, Prentice-Hall, Englewood Cliffs, NJ., 1991.

[Papazoglou et al., 1992]. M.P. Papazoglou, S.C. Laufman and T.K. Sellis, "An organizational framework for cooperating intelligent information systems", in: Journal of Intelligent and Cooperative Information Systems, vol. 1, no. 1, pp. 169-202, 1992.

[Patil et al., 1992]. R.S. Patil, R.E. Fikes, P.F. Patel-Schneider, D. McKay, T. Finin, T. Gruber, R. Neches, "The DARPA knowledge sharing effort: progress report", in: Proc. of Knowledge Representation and Reasoning (KR'92), C. Rich, W. Swartout, B. Nebel (eds.), pp. 777-788, 1992.

[Porter, 1980]. M.E. Porter, *Competitive strategy: Techniques for analyzing industries and competitors*, The Free Press, New York, NY., 1980.

[Porter, 1985]. M.E. Porter, *Competitive advantage: Creating and sustaining superior performance*, The Free Press, New York, NY., 1985].

[Powell, 1988]. M. Powell, "An input/output primitive for object-oriented systems", in: Information and Software Technology 30, no.1, jan./feb.'88. pp.44-56, 1988.

[Power, 1993]. R. Power, *Cooperation among organizations: the potential of computer supported cooperative work,* Springer-Verlag, Berlin, 1993.

[Raiffa, 1982]. H. Raiffa, *The Art and Science of Negotiation*, Harvard University Press, Boston, MA., 1982.

[Rao and Georgeff, 1991a]. A.S. Rao and M.P. Georgeff, "Asymmetry thesis and side-effect problems in linear time and branching time intentional logics", in: Proc. of the 12th Int.l Joint Conf. on AI (IJCAI'91), Sydney, Australia, pp.498-504, 1991.

[Rao and Georgeff, 1991b]. A.S. Rao and M.P. Georgeff, "Modeling rational agents within a BDI-architecture", in: Proc. of Knowledge Representation and Reasoning (KR&R-91), R. Fikes and E. Sandewall (eds.), Morgan Kaufmann Publishers, San Mateo, CA. pp. 473-484, 1991.

[Rao and Georgeff, 1992]. A.S. Rao and M.P. Georgeff, "Social plans: preliminary report", in: Proc. of Decentralized AI 3 - 3th European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-91), E. Werner and Y. Demazeau (eds.), Elsevier Science Publ., North-Holland, 1991.

[Rao and Georgeff, 1993]. A.S. Rao and M.P. Georgeff, "A model-theoretic approach to the verification of situated reasoning systems", in: Proc. of the 13th Int.l Joint Conf. on AI (IJCAI'93), Chambery, France, pp. 318-324, 1993.

[Reuter, 1989]. A. Reuter, "ConTracts: A means for extending control beyond transaction boundaries", in: Proc. of the 3rd Int.l Workshop on High Performance Transaction Systems, Asilomar, Sept., 1989.

[Reynolds, 1988]. G.W. Reynolds, *Information systems for managers,* West, St. Paul, MN, 1988.

[van Reijswoud, 1996]. V.E. van Reijswoud, *The structure of business communication: theory, model and application*, Ph.D. Thesis, Delft University of Technology, 1996.

[van Reijswoud and van der Rijst, 1995]. V.E. van Reijswoud and B.J. van der Rijst, "Modelling business communication as a foundation for business process redesign: a case of production logistics", in: Proc. of the 28th Hawaii Int.l. Conf. on Systems Sciences, IEEE Computer Society Press, Los Alamitos, pp 841-850, 1995.

[Rice, 1987]. R.E. Rice, "Computer mediated communication and organizational innovation", in: Journal of Communication, vol. 37, no. 4, pp. 65-94, 1987.

[Rice and Shook, 1990]. R.E. Rice and D.E. Shook, "Relationships of job categories and organizational levels to use of communication channels, including electronic mail: a meta-analysis and extension", in: Journal of Management Studies, vol. 27, no. 2, pp. 195-229, 1990.

[Robinson, 1991]. M. Robinson, "Computer supported cooperative work: Case and concepts", in: *Groupware 1991: The potential of team and organizational computing*, P.R. Hendriks (ed.), SERC, Utrecht, pp. 59-75, 1991.

[Rosenschein, 1985]. J.S. Rosenschein, "Formal theories of knowledge in AI and robotics", in: New Generation Computing, pp. 345-357, 1985.

[Rosenschein and Genesereth, 1985]. J.S. Rosenschein and M.R. Genesereth, "Deals among rational agents", in: Proc. of 9th Int.l Joint Conf. on AI (IJCAI'85), pp. 91-99, 1985.

Also published in *Readings in Distributed Artificial Intelligence*, A. Bond and L. Gasser (eds.), Morgan Kaufmann, pp. 227-234, 1988.

[Rosenschein and Kaelbling, 1986]. J.S. Rosenschein and L.P. Kaelbling, "The synthesis of digital machines with provable epistemic properties", in: Proc. of the 1986 Conf. on Theoretical Aspects of Reasoning about Knowledge (TARK'86), J.Y. Halpern (ed.), Morgan Kaufmann, San Mateo, CA., 1986.

[Rosenschein and Zlotkin, 1994]. J.S. Rosenschein and G. Zlotkin, "Designing conventions for automated negotiation", in: AI Magazine, Fall, pp. 29-46, 1994.

[Ross, 1973]. S. Ross, "The economic theory of agency", in: American Economic Review, 63, pp. 134-139, 1973.

[Royakkers and Dignum, 1994]. L. Royakkers and F. Dignum, "Deontic inconsistencies and authorities", in: Proc. of ECAI WS on Artificial Normative Reasoning, Amsterdam, Aug., 1994.

[Rusinkiewicz and Sheth, 1994]. M. Rusinkiewicz ad A. Sheth, "Specification and execution of transactional workflows", in: *The object model, interoperability, and beyond*, W. Kim (ed.), Addison-Wesley, 1994.

[van der Rijst and Dietz, 1993]. N.B.J. van der Rijst and J.L.G. Dietz, "Modelling the essential activities in Job Shop planning and scheduling for decision support system development", in: Proc of the 1993 European Simulation Symposium (ESS'93), Society for Computer Simulation, San Diego, CA., pp. 198-203, 1993.

[Ryu and Lee, 1992]. Young U. Ryu and Ronald M. Lee, *A formal representation of normative systems; A defeasible deontic reasoning approach,* EURIDIS Research Monograph no. RM-1992-08-1, Erasmus University, Rotterdam, 1992.

[Sachs, 1995]. P. Sachs, "Transforming work: Collaboration, learning and design", in: Communication of the ACM (CACM), vol. 38, no. 9, pp. 36-45, 1995.

[Santos and Carmo, 1996]. F. Santos and J. Carmo, "Indirect action, influence and responsibility in deontic logic, agency and normative systems", in: Proc. of DEON'96, Springer-Verlag, Berlin, 1996.

[Schäl, 1995]. T. Schäl, *Workflow management technology for process organizations*, Ph.D. Thesis Rheinisch-Westfälischen Technischen Hochschule, Aachen, 1995.

[Schäl and Zeller, 1993a]. T. Schäl and B. Zeller, "Workflow management sytems for financial services", in: Proc. of the Conf. on Organizational Computing Systems (COOCS'93), ACM Press, New York, NY., pp. 142-153, 1993.

[Schäl and Zeller, 1993b]. T. Schäl and B. Zeller, "Supporting cooperative processes with workflow management technology", Tutorial Proc., 3th European Conf. on Computer Supported Cooperative Work (ECSCW'93), Milano, Italy, 1993.

[Schiffrin, 1994]. D. Schiffrin, *Approaches to Discourse*, Blackwell, Cambridge, MA., 1994.

[Schmidt, 1993]. K. Schmidt, "Modes and mechanisms of interaction in cooperative work", in: Computational mechanisms of interaction for CSCW, C. Simone and K. Schmidt (eds.), COMIC Deliverable 3.1, ESPRIT BRA 6225, Lancaster University, Lancaster, 1993.

[Scott Morton, 1991]. M.S. Scott Morton, *The corporation of the 1990's: Information technology and organizational transformations*, Sloan School of Management, Oxford University Press, New York, 1991.

[Searle, 1969]. J.R. Searle, *Speech Acts,: An essay in the philosophy of language,* Cambridge University Press, 1969.

[Searle, 1971]. J.R. Searle (ed.), *The Philosophy of Language*, Oxford Univ. Press, London, 1971.

[Searle, 1979]. J.R. Searle, *Expression and Meaning: studies in the theory of speech acts*, Cambridge University Press, Cambridge, Mass., 1979.

[Searle, 1983]. J.R. Searle, *Intentionality: An essay in the philosophy of mind*, Cambridge University Press, Cambridge, Mass., 1983.

[Searle and Vanderveken, 1985]. J.R. Searle and D. Vanderveken, *Foundations of illocutionary logic*, Cambridge University Press, 1985.

[Senn, 1978]. J.A. Senn, *Information systems in management*, Wadsworth, Belmont CA, 1978.

[Sergot, 1990]. M. Sergot, "The representation of law in computer programs: A survey and comparison", in: *Knowledge based systems in the law*, T.J.M. Bench-Capon (ed.), Academic Press, 1990.

[Shannon and Weaver, 1949]. C.E. Shannon and W. Weaver, *The Mathematical Theory of Communications*, MIT Press, Cambridge, MA., 1949.

[Shehory and Kraus, 1993]. O. Shehory and S. Kraus, "Coalition formation among autonomous agents: Strategies and complexity", in: Proc. of the 5th European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW'93), Neuchâtel, Springer-Verlag, Berlin, 1993.

[Shepherd et al., 1990]. A. Shepherd, N. Mayer and A. Kuchinsky, "Strudel: an extensible electronic conversation toolkit", in: Proc. of the 2nd Conf. on Computer Supported Cooperative Work (CSCW'90), ACM Press, New York, 1990.

[Sheth et al., 1990] A. Sheth, M. Rusinkiewicz, G. Karabatis., "Using polytransactions to manage interdependent data", in: *Database Transaction Models for advanced applications*, A. Elmagarmid (ed), Morgan Kaufman, San Mateo, CA. 1992.

[Siewierska, 1991]. A. Siewierska, *Functional Grammar*, Routledge, London, 1991.

[Singh, 1994]. M.P. Singh, *Multiagent systems: A theoretical framework for intentions, know-how, and communication*, LNCS 799, Springer-Verlag, Berlin, 1994.

[Shoham, 1993]. Y. Shoham, "Agent-oriented programming", in: Artificial Intelligence 60, pp. 51-92, 1993.

[Shoham and Cousins, 1994]. Y. Shoham and S.B. Cousins, "Logics of mental attitudes in AI", in: *Foundations of Knowledge Representation and Reasoning*, G. Lakemeyer and B. Nebel (eds.), LNAI, Springer-Verlag, Berlin, 1994.

[Shoham and Tennenholtz, 1992]. Y. Shoham and M. Tennenholtz, "On the synthesis of useful social laws for artificial agent societies", in: Proc. of the 10th Nat.l Conf. on AI (AAAI'92), San Jose, CA., AAAI Press, Menlo Park, CA., pp. 276-281, 1992.

[Smith, 1980]. R.G. Smith, "The contract net protocol: high level communication and control in a distributed problem solver", in: IEEE Transactions on Computing, 29 (12), pp. 1104-1113, 1980. (reprinted in *Readings in Distributed Artificial Intelligence*, A.H. Bond and L. Gasser (eds.), Morgan Kaufmann, San Mateo, CA., 1988)

[Smith and Davis, 1980]. R.G. Smith and R. Davis, "Frameworks for cooperation in distributed problem solvers", in: IEEE Transactions SMC (Systems, Man & Cybernetics), 11 (1), pp. 61-70, 1980. (reprinted in *Readings in Distributed Artificial Intelligence*, A.H. Bond and L. Gasser (eds.), Morgan Kaufmann, San Mateo, CA., 1988).

[Sowa, 1984]. J.F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, Mass. 1984.

[Spewak and Hill, 1993]. Steven H. Spewak and Steven C. Hill, *Enterprise Architecture Planning: Developing a blueprint for data, applications and technology*, QED Publ.Group, Wellesley, MA, USA, 1993.

[Steuten and van Reijswoud, 1996]. A. Steuten and V.E. van Reijswoud, "The interpretation of business communication", in: [Dignum et al., 1996c].

[Suchman, 1987]. L. Suchman, *Plans and Situated Actions*, Cambridge University Press, Cambridge, MA., 1987.

[Suchman, 1994]. L. Suchman, "Do Categories have Politics? The Language/Action Perspective Reconsidered", in: Computer Supported Cooperative Work (CSCW), Vol. 2, no. 3, pp. 177-190, 1994. (also: Proc of 3rd ECSCW Conf, G. De Michelis, C. Simone, K. Schmidt (eds.), 1993.)

[Suchman, 1995]. L. Suchman, "Making work visible", in: Communications of the ACM (CACM), vol. 38, no. 9, pp. 56-70, 1995.

[Sycara, 1990]. K. Sycara, "Persuasive argumentation in negotiation", in: Theory and Decision, 28, pp. 203-242, 1990.

[Szekely and Myers, 1988]. P. Szekely and B. Myers, "A user-interface toolkit based on graphical objects and constraints", in: Proc. of OOPSLA'88, pp. 36-45, 1988.

[Tan, 1993]. M. Tan, "Using communication theory for systems design: A model for eliciting information requirements", in: *Human, Organizational and Social Dimensions of Information Systems Development*, D. Avison, J.E. Kendal, J.I. DeGross (eds.), Elsevier Science Publ., North-Holland, pp. 241-262, 1993.

[Tan and Thoen, 1996]. Y.-H. Tan and W. Thoen, "Modelling the dynamics of transferable obligations in business procedures", in: [Dignum et al., 1996c].

[Taylor, 1990]. D.A. Taylor, *Object-Oriented Technology: A Manager's Guide*, Servio-Logic, 1990.

[Taylor, 1993]. J.R. Taylor, *Rethinking the Theory of Organizational Communication; How to read an organization*, Ablex Publishing, Norwood, NJ, 1993.

[Teng et al., 1992]. J.T.C. Teng, W.J. Kettinger, and S. Guha, "Business Process Redesign and Information Architecture: Establishing the Missing Links", in: Proc. of 13th Int.l Conf. on Information Systems (ICIS), JI. DeGross, J.D Becker and J.J. Elam (eds.), pp. 81-90, 1992.

[Teufel and Teufel, 1995]. S. Teufel and B. Teufel, "Bridging Information Technology and Business - Some Modelling Aspects", in: ACM SIGOIS Bulletin 16 (1), pp. 13-17, 1995.

[Thomas et al, 1991]. S.R. Thomas, Y. Shoham, A. Schwartz, S. Kraus, "Preliminary thoughts on an agent description language", in:International Journal of Intelligent Systems, vol. 5, no.6, pp. 497-508, 1991.

[Thomas, 1993]. S.R. Thomas, *PLACA, an agent oriented programming language*, Ph.D. Thesis, Stanford University, Report No. STAN-CS-93-1487, Stanford, CA., 1993.

[Thomason, 1981]. R.H. Thomason, "Deontic logic as founded on tense logic", in: *New studies in deontic logic*, R. Hilpinen (ed.), Reidel, Dordrecht, pp. 165-176, 1981.

[Vanderveken, 1990]. D. Vanderveken, *Meaning and Speech Acts*, Cambridge University Press, Cambridge, MA., 1990.

[Verharen et al., 1994]. E. Verharen, H. Weigand, and O. De Troyer, "Agent-oriented information system design", in: Working Papers. of the Int.l. Workshop on Information Systems, Correctness and Reusability (ISCORE'94), R. Wieringa and R. Feenstra, (eds), Vrije Universiteit Report IR-357, Amsterdam, pp. 378-392, 1994.

[Verharen and Weigand, 1994]. E. Verharen and H. Weigand, "Agent Oriented Information Systems Design", in: Poster Proc. of the Int.l. Symposium on Methodologies for Intelligent Systems (ISMIS'94), Z. Ras and M. Zemankova (eds.), Oak Ridge Nat.l. Lab., 1994.

[Verharen et al., 1996]. E. Verharen, F. Dignum, H. Weigand, "A language/action perspective on cooperative information agents", in: [Dignum et al., 1996c].

[Verharen and Dignum, 1997]. E. Verharen and F. Dignum, "Cooperative Information Agents and Communication", in: Proc. of the First Int.l. Workshop on Cooperative Information Agents, M. Klusch (ed.), 27-28 Februari 1997, Kiel, Germany, LNAI, Springer-Verlag, Berlin, 1997. (To appear)

[Verheijen and van Bekkum, 1982]. G. Verheijen and J. van Bekkum, "NIAM: an information analysis method", in: Proc. IFIP TC-8 Conf. on Comparative review of information systems methodologies (CRIS-1), A.A. Verrijn-Stuart, T.W. Olle, H. Sol (eds.), North-Holland, pp. 537-589, 1992.

[Vet, 1990]. C. Vet, "Asymmetries in the use of tense and modality", in: *Layers and Levels of Representation in Language Theory: a functional view*, J. Nuyts, A.M. Bolkenstein, C. Vet (eds.), John Benjamins, Amsterdam/Philadelphia, 1990.

[Wächter and Reuter,1992]. H. Wächter and A. Reuter, "The ConTract Model", in: *Database Transaction Models for advanced applications*, A. Elmagarmid (ed), Morgan Kaufman, San Mateo, CA., 1992.

[Wagner, 1996]. G. Wagner, "Vivid agens: How they deliberate, how they react, how they are verified", report Institut für Informatik, Universität Leipzig, Germany, 1996. (also versions have been published in proc. of ModelAge'96, and proc. of MAAMAW'96).

[Wasserman and Shewmake, 1985]. A.I. Wasserman and D.T. Shewmake, "The role of prototypes in the user software engineering methodology", in: *Advances in Human-Computer Interaction, vol. 1*, H. Rex Hartson (ed.), Ablex, 1985.

[Weber, 1956]. M. Weber, *Economy and Society*, Bedminster Press, New York, NY., 1956.

[van de Weg, 1995]. R.L.W. van de Weg, Analysis and Design of Information Systems based on an Object-Oriented Framework, Ph.D. Thesis, Universiteit Twente, Enschede, 1993.

[Weigand, 1989]. E. Weigand, *Sprache als Dialog : Sprechakttaxonomie und kommunikative Grammatik*, Niemeyer Verlag, Tubingen, 1989.

[Weigand, 1990]. H. Weigand, *Linguistically Motivated Principles of Knowledge Based Systems*, Foris, Dordrecht, 1990.

[Weigand, 1991a]. H. Weigand, "The linguistic turn in information systems", in: Proc. of Collaborative Work, Social Communications and Information Systems, R. Stamper, P. Kerola, R. Lee and K. Lyytinen (eds.), IFIP, Elsevier Scientific Publ., North-Holland, 1991.

[Weigand, 1991b]. H. Weigand, "An object-oriented approach in a multimedia database project", in: Proc. of IFIP DS-4: Object-Oriented Databases: Analysis, Design & Construction, R.A. Meersman, W. Kent, S. Khosla (eds), IFIP, Elsevier Science Publ., North-Holland, 1991.

[Weigand, 1992a]. H. Weigand, "Assessing Functional Grammar for Knowledge Representation", in: Data and Knowledge Engineering 8 (1992), pp. 191-203, 1992.

[Weigand, 1992b]. H. Weigand, "Towards a design methodology for interoperable databases", in: Proc. IFIP WG 2.5 Conf. Semantics of Interoperable Database Systems (DS-5), Lorne Australia, D. Hsiao, E.J. Neuhold, R. Sacks-Davis (eds.), 1992.

[Weigand, 1993]. H. Weigand, "Deontic aspects of communication", in: *Deontic Logic in Computer Science*, J.-J.Ch. Meyer and R. Wieringa (eds.), John Wiley and Sons Ltd., Chichester, 1993.

[Weigand et al., 1995]. H. Weigand, E. Verharen, and F. Dignum, "Integrated Semantics for Information and Communication Systems", in: Proc. of IFIP WG 2.5 Conf. on Database Application Semantics (DS-6), R. Meersman, L. Mark (eds), Stone-Mountain, Georgia, 1995.

[Weigand et al., 1996]. H. Weigand, E. Verharen and F. Dignum, "Interoperable transactions - a structured approach", in: Proc. of 8th Int.l. Conf. on Advanced Information Systems Engineering (CAISE'96), P. Constantopoulos, J. Mylopoulos and Y. Vassiliou (eds.), LNCS 1080, Springer-Verlag, Berlin, 1996.

[Weikum, 1986]. G.A. Weikum, "Theoretical Foundations of Multi-level Concurrency Control" in: Proc. of conf. Principles of Distributed Systems (PODS'86), 1986.

[Werner, 1989]. E. Werner, "Cooperating agents: A unified theory of communication and social structure", in: *Distributed AI, vol. II*, L. Gasser and M. Huhns (eds.), Pitman Publ., London and Morgan Kaufmann, San Mateo, CA., pp. 3-36, 1989.

[Werner, 1991]. E. Werner, "A unified view of information, intention and ability", in: Decentralized AI - 2, Proc. of 2nd European WS on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW'90), Y. Demazeau and J.P. Müller (eds.), Elsevier Science Publ., North-Holland, pp. 109-126, 1991.

[White, 1994]. J.E. White, *Telescript Technology: The foundation for the electronic marketplace*, White Paper, General Magic, Mountain View, CA., 1994.

[Wieringa et al., 1989]. R. J. Wieringa, J.-J.Ch. Meyer and H. Weigand, "Specifying dynamic and deontic integrity constraints", in: Data and Knowledge Engineering 4, pp. 157-189, 1989.

[Wieringa et al., 1991]. R.J. Wieringa, H. Weigand, J.-J.Ch. Meyer and F. Dignum, "The Inheritance of Dynamic and Deontic Integrity Constraints", in: Annals of Mathematics and Artificial Intelligence, no. 3, pp.393-428, 1991.

[Wieringa, 1993]. R. Wieringa, "A method for building and evaluating formal specifications of object-oriented conceptual models of database systems", Rapport IR-340, Free University, Amsterdam, dec. 1993.

[Wieringa and Meyer, 1993]. R.J. Wieringa and J.-J.Ch. Meyer, "Applications of Deontic Logic in Computer Science: a concise overview", in: *Deontic Logic in Computer Science*, J.-J.Ch. Meyer and R. Wieringa (eds.), John Wiley and Sons Ltd., Chichester, 1993.

[Winograd, 1988]. T. Winograd, "A Language/Action Perspective on the Design of Cooperative Work", in: *Computer Supported Cooperative Work: A Book of Readings*, I. Greif (ed.), Morgan Kaufmann, San Mateo, CA., 1988. (also Human computer Interaction 3 (1), 1987/88, p. 3-30)

[Winograd, 1994]. T. Winograd, "Categories, Disciplines, and Social Coordination", in: Computer Supported Cooperative Work (CSCW), Vol. 2, no. 3, pp. 191-197, 1994.

[Winograd and Flores, 1986]. T. Winograd and F. Flores, *Understanding Computers and Cognition: A New Foundation for Design,* Ablex, Norwood NJ, 1986. (Available from Addison-Wesley)

[Wittgenstein, 1968]. L. Wittgenstein, *Philosophical Investigations*, Basil Blackwell & Mott. Ltd., Oxford, 1968.

[Wolper, 1981]. P. Wolper, "Specification and synthesis of communicating processes using an extended temporal logic", in: Proc. of 9th Annual ACM Symposium on Principles of Programming Languages (PoP'81), ACM Press, New York, NY., pp. 20-33, 1981.

[Wooldridge, 1994]. M. Wooldridge, "Coherent social action", in: Proc. of 11th European Conf. on AI (ECAI'94), Amsterdam, The Netherlands, pp. 279-283, 1994.

[Wooldridge, 1995]. M. Wooldridge, "This is MYWORLD: The logic of an agent-oriented testbed for DAI", in: *Intelligent Agents: Theories, Architectures and Languages*, M. Wooldridge and N.R. Jennings (eds.), LNAI 890, Springer-Verlag, Heidelberg, pp. 160-178, 1995.

[Wooldridge and Jennings, 1994]. M. Wooldridge and N.R.,Jennings, "Formalizing the cooperative problem solving process", in: Proc. of 13th Int.l WS on Distributed AI (IWDAI'94), Lake Quinalt, WA., pp. 403-417, 1994.

[Wooldridge and Jennings, 1995]. M. Wooldridge and N.R. Jennings, "Intelligent Agents: Theory and Practice", Knowledge Engineering Review 10(2), 1995.

[von Wright, 1951]. G.H. von Wright, "Deontic logic", in: Mind 60, pp. 1-15, 1951.

[von Wright, 1963]. G.H. von Wright, *Norm and Action: A logical enquiry*, Routledge and Kegan Paul, London, 1963.

[von Wright, 1964]. G.H. von Wright, "A new system of deontic logic", in: *Danish Yearbook of Philosophy 1*, pp. 173-182, 1964.

[von Wright, 1965]. G.H. von Wright, "A correction to a new system of deontic logic", in: *Danish Yearbook of Philosophy 2*, pp. 103-107, 1965.

[von Wright, 1968]. G.H. von Wright, "An Essay in Deontic Logic and the General Theory of Action", in: Acta Philosophica Fennica, Fasc. 21, North-Holland, 1968.

[Von Wright, 1980]. G.H. von Wright, "Problems and prospects of deontic logic: a survey", in: *Modern Logic - A Survey: Historical, Philosophical and Mathematical Aspects of Modern Logic and Its Applications*, E. Agazzi (ed.), Reidel, Dordrecht, pp. 399-423, 1980.

[Zlotkin and Rosenschein, 1991]. G. Zlotkin and J.S. Rosenschein, "Cooperation and conflict resolution via negotiation amon autonomous agents in noncooperative domains", in: IEEE Transactions SMC, 21(6), pp. 1317-1324, 1991.

SUMMARY

The research reported on in this dissertation focuses on a new way of designing a class of automated information systems, called Cooperative Information Systems.

Cooperative Information Systems (CIS) are seen as the next step in the evolution of information systems (IS) from central data repositories to systems that are used within and between organizations to support cooperation and the coordination of organizational activities. This coordination is achieved by communication. Traditional IS development methods do not pay much attention to design aspects of communication and coordination within and across organizations. Therefore, a new perspective is needed that focuses on the use of communication as a coordination mechanism and that can be used to obtain a better understanding of the structure of business communication.

In this dissertation a Language-Action Perspective (LAP) to the design of CISs is taken. The LAP describes what people and systems *do* while communicating, how they create a common reality by means of language, and what it means to commit to the conduct of some business activity. It considers an organization as a network of inter-*acting* agents that create, maintain and terminate commitments. The LAP has proven to be a new basic paradigm for a new generation of IS design methods and business process models. However, there is not much convergence yet on the formal and logical underpinnings of communication models and the LAP. Here, a formal framework and specification language are proposed that can be applied for the design of CISs.

In this dissertation CISs are considered to be intelligent communicating agents. Communication, in the definition used here, is not just information exchange, its essence is *to commit the partners in communication (called subjects or agents) to a course of action so that one can rely on the other*. It is a constant organizational effort to explicate commitments (both intra- and inter-organizational) in order to improve the cooperation and support the coordination of activities. ISs are instrumental in this effort, both by taking over certain routine tasks, and by explicating the rules of the communicative actions, the meaning of the terms and other kinds of mutual knowledge. Important aspects are the authorizations and obligations of the communicating agents. In order to structure the communication both a business logic framework describing the overall interaction process, and a communication framework describing the communicative actions of the agents, are developed.

For ISs to be able to cooperate they must have an intelligent interface that can cope with all types of requests for information from users or other systems. An IS actively maintains its information; it communicates with other systems and reasons about the information it contains. It might decide to search for information it needs by inquiring for it from other ISs, preferably in ways it negotiates (and lays down in contracts) with those other systems. Therefore the IS should contain a task module that plans the tasks it has to fulfil. CISs are not integrated but rather function autonomously. The behavior of the system cannot be entirely prescribed from start to finish, but they have to be responsive to changes in the environment and pro-actively take opportunities when they arise. For this the notion of *agents* is introduced. I refer to an autonomous CIS with tasks and contracts as a *Cooperative Information Agent* (CIA).

Recent years have shown a growing interest in intelligent agents. Although there is no consensus on what an agent is precisely, in this thesis an agent is defined as: *an autonomous computational entity and its behaviour is not predefined but based on commitments to other agents.*

Intelligent agents seem to be suitable candidates for implementing CISs, and in this dissertation an agent architecture for CISs is described.

What follows is a brief overview of the different chapters of this dissertation.

Chapter 1 gives the motivation for this research, and justification for the choices of taking a LAP, the use of dynamic deontic and illocutionary logic, and agent-technology for the design of CISs.

Chapter 2 describes background theory and related work. Most of the work in the LAP is based on Searle's speech act theory. A valuable addition is given by Habermas' theory of communicative action. Both theories were developed to describe the way people communicate. In this thesis, however, I am interested in how these theories can be applied to the design of automated systems. Therefore some adaptations were made to the theories, which are described in this chapter. Furthermore an overview is given of related research in the LAP community. The chapter also contains an overview of the research on agent technology, mainly from the AI discipline. Again, similar approaches are described.

The kernel of this dissertation consists of the chapters 3-6. Chapter 3 describes a framework for business oriented communication. It gives a generic business logic for modelling business communication between a supplier of services or goods, and a customer needing the service. The framework not only is applicable in business settings between organizations, but in every situation where one agent (system or person) needs a service from another agent, even if they belong to the same organization, i.e., in every communication a supplier and customer role can be distinguished. The business logic framework emphasizes the symmetrical roles and relations of the agents and the total *action logic* of the business transaction. It structures business communication by identifying four phases: proposal and negotiation (setting up conditions), commitment and contracting (coming to a mutual agreement), fulfilment (performance of the actions), and completion (or satisfaction).

Chapter 4 first introduces the three-level communication framework for describing the interaction between two agents, consisting of (interoperable) transactions, contracts, and tasks. After this an agent architecture is given. It consists of a number of knowledge bases and functional components working on them. The knowledge bases that are identified are: Transactions, Contracts, Tasks, Services, the Lexicon, the Database, and the Agenda. The main functional components are: the Communication Manager, the Contract Manager, the Task Manager, and the Service Execution Manager. The working of these components is described.

Chapter 5 describes the three components of the communication framework in more detail. Interoperable *transactions* are specified as a set of messages together with (temporal) relations between them, and a goal identified by particular message occurrences. Following the business logic, communication behaviour between two agents is described by *contracts*. The contract outlines the obligations and authorizations of the communicating systems using interoperable transactions, together with deontic constraints, and rules for appropriate action when they are violated. A *task* is a meaningful unit of work assigned to an agent, specified by a goal and consisting of a number of subtasks, and relationships and dependencies between them. The task's specification and updates thereof do concern the agent in question only, whereas changes in the transactions can only be made by consent of other agents involved. For that reason the distinction between task and contract is made, where the contract corresponds to the agreements between the agents and the task draws on this potential for fulfilling an agent's goal.

Besides the characteristics of the components also general features like deadlines and failure management are described. Special attention is paid to the constructing of contracts, the relationships between contracts, and the construction of the agenda. Furthermore, the specification language CoLa (Communication and Coordination Language) is developed in which the components can be described, and that can be used for the design of CIAs. In the chapter examples of transaction, contract and task specifications are given.

Chapter 6 describes the formal framework. For the semantics of communication models a combination of *Dynamic Deontic Logic* and *Illocutionary logic* is used. It provides dynamic concepts for dealing with (communicative) actions and transactions. Deontic concepts make it possible to describe the course of action in the case the communication protocol fails. Also the way deontic statements are created and adapted in communication processes, and the role they play in the regulation of communication itself is explored. It is shown how authorizations can be requested, granted and retracted, thereby creating a dynamic environment for the establishment and derogation of authorized norms. Illocutionary logic is a logical formalisation of the speech act theory and is used to formally describe the message structure itself, i.e., the types and effects of the messages. The combination of dynamic deontic logic and illocutionary logic gives a formal framework and integrated semantics that provides for the precise description of (the structure and effects of) communicative actions. Furthermore, it serves as a basis for the formal specification language.

In chapter 7 a modelling methodology for CIS development is described. The aim of the methodology is to produce abstract models of organizational communication as basis for formulating requirements of a supporting software system. The methodology is strongly communication-driven and influenced by the separation between Environment of Discourse (organizational environment) and Universe of Discourse (information (data, rules) about the domain). It describes stages of organizational analysis and (re)design, communication modelling and UoD modelling, and system specification. For modelling purposes graphical models are provided. The methodology can be used to guide development of automated ISs, but the first steps are also valuable if no automated system is implemented, in that it clearly describes the authorization and communication relations that can be used to improve on the way business is conducted.

Finally, chapter 8 gives the results and conclusions of the research and an agenda for further research. From the conclusions can be derived that a language-action perspective can be adequately and successfully applied to the design of cooperative information systems for the support of business communication.

The results from this research can be applied in different application settings, inter-organizational, as in Business Computing, EDI, or intra-organizational, as in Groupware and Workflow Management.

The three main topics discussed in this thesis: communication, intelligent agents and CIS design, make for an interesting combination that in my view describes one of the future paths IS development is taking. Hence the title of the thesis:

"a language-action perspective
on the design of
cooperative information agents".

SAMENVATTING

EEN TAALHANDELING-PERSPECTIEF OP HET ONTWERPEN VAN COÖPERATIEVE INFORMATIE-AGENTEN[1]

Dit proefschrift beschrijft het onderzoek naar een nieuwe manier van het ontwerpen van geautomatiseerde Coöperatieve Informatiesystemen.

Coöperatieve Informatiesystemen (CIS) worden gezien als de volgende stap in de informatiesysteem-evolutie van centraal data-opslagsysteem naar systemen die in en tussen organisaties gebruikt worden voor de ondersteuning van samenwerking en de coördinatie van bedrijfsactiviteiten. Dit wordt bewerkstelligd door communicatie. Traditionele IS-ontwerpmethoden besteden geen of weinig aandacht aan de ontwerpaspecten van communicatie en coördinatie. Daarom is een nieuw perspectief nodig dat zich richt op het gebruik van communicatie als coördinatiemechanisme en dat gebruikt kan worden om een beter begrip te verkrijgen van de struktuur van zakelijke communicatie. In dit proefschrift volg ik het taalhandelings-perspectief voor het ontwerpen van CIS. Het taalhandelings-perspectief beschrijft wat mensen en systemen *doen* tijdens communicatie, hoe zij een gemeenschappelijk beeld van de werkelijkheid creëren door het gebruik van taal, en wat het betekent om zich te verplichten tot het uitvoeren van een bepaalde bedrijfsactiviteit. Vanuit dit perspectief wordt een organisatie gezien als een netwerk van inter*acterende* agenten die verplichtingen creëren, onderhouden, wijzigen, rapporteren en opheffen.

Het taalhandelings-perspectief heeft zich ontwikkeld tot een paradigma voor een nieuwe generatie IS-ontwerpmethoden en bedrijfsactiviteitmodellen. Er is echter nog niet veel overeenstemming over de formele en logische onderbouwing ervan. In dit proefschrift zijn een formeel raamwerk en specificatietaal ontwikkeld die dienen als basis voor formalisatie van het perspectief en de communicatie modellen, en die kunnen worden gebruikt bij CIS ontwikkeling.

In dit proefschrift wordt een CIS beschouwd als een intelligente, communicerende agent. Volgens de hier gebruikte definitie, is *communicatie* niet alleen maar informatie uitwisseling, maar de essentie is dat ***de communicerende partijen zich verplichten tot een verloop van activiteiten, zodat zij op elkaar kunnen vertrouwen en hun activiteiten kunnen coördineren.*** Het is voor de organisatie van belang om alle verplichtingen (zowel intra- als interorganisationeel) boven tafel te krijgen, zodat samenwerking en coördinatie kan worden verbeterd. IS zijn hierbij van onmiskenbaar belang, zowel door het overnemen van routinetaken, als het expliciet maken en vastleggen van de regels voor communicatieve actie, de betekenis van de gebruikte termen en andere gemeenschappelijke kennis. Belangrijke begrippen hierbij zijn 'autorisatie' en 'verplichting'. Om de communicatie te beschrijven en te structureren worden een bedrijfslogica raamwerk dat het gehele interactie process beschrijft, en een communicatieraamwerk dat de communicatieve acties van de agenten beschrijft ontwikkeld.

Voor samenwerking tussen systemen is een intelligente interface nodig die overweg kan met allerlei verzoeken voor informatie van gebruikers of andere systemen. Het IS beheert zijn informatie

---

[1] Nederlandstalige samenvatting van: E.M. Verharen, A Language-Action Perspective on the Design of Cooperative Information Agents, Proefschrift, Katholieke Universiteit Brabant, Tilburg, 1997. ISBN 90-9010286-8.

actief; het redeneert en communiceert hierover met andere systemen. Het kan besluiten andere systemen om informatie te vragen als het deze nodig heeft en zelf niet bevat, bij voorkeur op een manier die het onderhandeld heeft (en heeft vastgelegd in een contract) met de andere systemen. Het IS heeft daarom een taakmodule nodig, die de taken plant die het systeem moet en wil uitvoeren. De CIS kunnen vanwege organisatorische en technische redenen vaak niet geïntegreerd worden, maar functioneren *autonoom*. De werking van zo'n systeem kan niet volledig beschreven worden van te voren. In plaats daarvan moeten ze kunnen reageren op wijzigingen in de omgeving en pro-actief kunnen handelen als dat uitkomt. Hiervoor wordt het begrip '*agent*' geïntroduceerd. Een autonome CIS met taken en contracten wordt een *Coöperatieve Informatie Agent* (CIA) genoemd.

De laatste jaren is er een groeiende interesse voor intelligente agenten. Hoewel er nog geen overeenstemming is over wat een agent precies is, wordt een agent in dit proefschrift gedefinieerd als *een autonome computationele entiteit waarvan het gedrag niet vooraf wordt vastgelegd maar wordt bepaald door de verplichtingen die de agent aangaat tegenover andere agenten*.

Intelligente agenten lijken geschikte kandidaten om CIS te implementeren en in het proefschrift wordt een intelligente agent architectuur voor CISs beschreven.

Hierna volgt een overzicht van de hoofdstukken van dit proefschrift.

Hoofdstuk 1 geeft de motivatie voor dit onderzoek, en een verantwoording voor de verschillende keuzes, zoals die voor het taalhandelings-perspectief, het gebruik van dynamische deontische en illocutionaire logica, en agent-technologie voor het ontwerp van CIS.

Hoofdstuk 2 beschrijft achtergrond theorie en gerelateerd onderzoek. Het meeste werk in het taalhandelings-perspectief is gebasserd op Searle's taalhandelingstheorie. Een waardevolle aanvulling hierop is Habermas' theorie van communicatief handelen. Beide theorieën beschrijven de manier waarop mensen communiceren. In dit proefschrift ben ik echter geïnteresseerd in de toepassing van deze theorieën bij het ontwerpen van geautomatiseerde systemen en hun onderlinge communicatie. Daarom zijn enige aanpassingen gemaakt, die beschreven staan in dit hoofdstuk. Hiernaast wordt ook een overzicht gegeven van gerelateerd onderzoek naar het taalhandelings-perspectief. Dit hoofdstuk bevat tevens een overzicht van onderzoek naar agent technologie, vooral vanuit de Kunstmatige Intelligentie. Ook het gerelateerde onderzoek op dit gebied wordt beschreven.

De kern van het proefschrift bestaat uit de hoofdstukken 3 tot en met 6. Hoofdstuk 3 beschrijft een raamwerk voor het zakelijke communicatieproces. Het geeft een generieke handelingslogica voor het modelleren van zakelijke communicatie tussen een leverancier van diensten of goederen en een klant die hieraan behoefte heeft. Het raamwerk is niet alleen nuttig in zakelijke omgevingen maar kan toegepast worden in iedere situatie waar een agent (persoon of systeem) een dienst van een andere nodig heeft, zelfs als deze tot dezelfde organisatie behoort, d.i. in iedere communicatie situatie kan een leverancier- en klantrol onderscheiden worden. Het raamwerk benadrukt de symmetrische relatie tussen de agenten en de totale *actielogica* van de zakelijke transacties. Het structureert zakelijke communicatie en onderscheidt vier fasen: de voorstel- en onderhandelingsfase (voor het opzetten van handelscondities), de verplichtingen- en contractenfase (waarbij tot een gemeenschappelijke overeenkomst gekomen wordt), de uitvoeringsfase, en de voltooiings- en voldoeningsfase.

Hoofdstuk 4 introduceert allereerst een drie-lagen communicatieraamwerk bestaande uit (interoperabele) transacties, contracten en taken, dat gebruikt wordt om de interactie tussen systemen in detail te kunnen beschrijven. Hierna wordt een agent architectuur gegeven. Deze bestaat uit een aantal kennisbanken en functionele componenten die daar op werken. De kennisbanken die

onderscheiden worden, zijn: Transacties, Contracten, Taken, Diensten, het Lexicon, de database, en de Agenda. De functionele componenten zijn: de communicatie manager, de contract manager, de taak manager en de dienstenuitvoerings manager.

Hoofdstuk 5 beschrijft de drie lagen van het communicatieraamwerk in meer detail. Interoperabele *transacties* worden gespecificeerd als een verzameling boodschappen samen met hun (temporele) relaties. Het communicatieve gedrag van de agenten wordt, de handelingslogica volgend, beschreven door contracten. Een *contract* specificeert de verplichtingen en autorisaties van de agenten, gebruikmakend van de interoperabele transacties, en regels die specificeren welke acties ondernomen moeten worden als er een overtreding van het contract plaatsvindt. Een *taak* is een betekenisvolle eenheid werk dat is toegewezen aan een agent, bestaande uit een aantal subtaken en relaties en afhankelijkheden tussen de subtaken. De taakspecificatie en wijzigingen daarvan gaan alleen de agent in kwestie aan, maar veranderingen in de transacties kunnen alleen gemaakt worden met toestemming van de andere agenten. Daarom is er onderscheid gemaakt tussen taken en contracten; de contracten corresponderen met de afspraken die gemaakt zijn tussen agenten, en de taken gebruiken dit potentieel om de doelen van de agent te bereiken.

Naast de eigenschappen van deze componenten worden ook algemene kenmerken zoals tijdslimieten en foutafhandeling beschreven. Speciale aandacht wordt geschonken aan contract-constructie en relaties tussen contracten, en aan de opbouw van de agenda. Tevens wordt de specificatietaal CoLa (communicatie en coördinatie taal) gedefinieerd, die gebruikt kan worden voor het ontwerp van CIAs. Van transactie-, contract- en taakspecificatie worden voorbeelden gegeven.

Hoofdstuk 6 beschrijft het formele raamwerk. Voor de semantiek van communicatiemodellen wordt een combinatie van *dynamische deontische logica* en *illocutionaire logica* gebruikt. Dynamische concepten worden gebruikt voor de beschrijving van (communicatieve) acties en transacties. Deontische concepten maken het mogelijk om te beschrijven wat er moet gebeuren in het geval het communicatie-protocol faalt. Tevens wordt onderzocht op welke manier deontische verklaringen worden gecreëerd en veranderd tijdens een communicatieproces, en de rol die ze spelen in de regulering van de communicatie zelf. Er wordt aangetoond hoe autorisaties worden aangevraagd, toegewezen en ingetrokken, hiermee een dynamische omgeving creëerend voor het vaststellen en opheffen van normen. Illocutionaire logica is de formalisatie van Searle 's taalhandelingstheorie en wordt gebruikt om de boodschapstruktuur zelf, d.i. de typen en effecten van de boodschappen, formeel te beschrijven. De combinatie van dynamische deontische en illocutionaire logica levert een formeel raamwerk en semantiek op voor de precieze beschrijving van (de structuur en effecten van) communicatieve acties. Verder dient het als basis voor de formele specificatietaal.

In hoofdstuk 7 wordt een modelleringsmethodologie voor CIS-ontwerp beschreven. Het doel van de methodologie is om abstracte modellen van zakelijke communicatie te produceren als basis voor het formuleren van eisen aan een ondersteunend software systeem. De methodologie is sterk communicatie-georiënteerd en gaat uit van de scheiding tussen 'Environment of Discourse' (de organisatie-omgeving) en 'Universe of Discourse' (de domein-informatie (data, regels)). Het beschrijft organisatie analyse en (her)ontwerp, communicatie en informatie modellering, en systeem specificatie. Voor modelleringsdoeleinden worden grafische modellen aangereikt. Hoewel de methodologie gebruikt kan worden bij het ontwerp van geautomatiseerde systemen, kunnen de eerste stappen ook waardevol zijn als er geen geautomatiseerd systeem geïmplementeerd wordt. Zij beschrijven namelijk duidelijk de autorisatie- en communicatie-relaties die gebruikt kunnen worden om de manier waarop wordt gehandeld te verbeteren.

Tenslotte worden in hoofdstuk 8 de resultaten en conclusies van het onderzoek samengevat en plannen voor vervolgonderzoek beschreven. Het proefschrift brengt de theorie van communicatieve actie over naar de communicatie tussen formele systemen met het doel om CIS te ontwerpen. De aanpasssingen die gemaakt zijn bestaan uit: de formele en expliciete beschrijving van autorisaties en obligaties, de verschillende autorisatierelaties die tussen agenten bestaan en hun effecten. Tevens is er een handelingslogica, waar nadruk wordt gelegd op de symmetrie van de communicatie, en communicatie raamwerk ontwikkeld. De combinatie van dynamische deontische en illocutionaire logica resulteert in een formeel raamwerk en geïntegreerde semantiek voor communicatie-modellering. Het maakt het mogelijk om een beter inzicht te verschaffen in de fundamentele concepten die ten grondslag liggen aan het taalhandelings-perspectief. Een modellerings-methodologie en specificatietaal zijn ontwikkeld om het ontwerp van CIA te ondersteunen.

Uit de conclusies kunnen we afleiden dat een taalhandelings-perspectief adequaat en succesvol kan worden toegepast bij het ontwerpen van coöperatieve informatie systemen voor het ondersteunen van zakelijke communicatie.

De resultaten van het onderzoek kunnen worden toegepast in verschillende applicatie omgevingen, zoals bijvoorbeeld EDI, bedrijfsprocesmodellering, en groupware en workflow management.

De drie onderwerpen die in het proefschrift worden besproken: communicatie, intelligente agenten, en CIS ontwerp, vormen in mijn ogen een interessante combinatie die een toekomstig pad in informatiesysteemontwikkeling beschrijft. Vandaar de titel van dit proefschrift:

"een taalhandelings-perspectief op
het ontwerpen van
coöperatieve informatie agenten"

Curriculum Vitae

Egon Verharen was born in Abcoude (Utrecht) on June 7, 1965. After graduating from the St. Nicolaaslyceum (VWO) in Amsterdam in June 1983, he enrolled as a student of Computer Science at the Technische Hogeschool Twente in Enschede. He received his ir.-title (MSc.) from the by then renamed Universiteit Twente in June 1988. His master's thesis, written under supervision of dr. Peter Braspenning and prof. dr. Peter Apers, was on Deductive Semantic Database Systems.

In December 1988 he joint the Institute for Language Technology and Artificial Intelligence (ITK) at the Katholieke Universiteit Brabant, as a research assistant. Here he worked on several language-technology related projects, before being responsible for the design of the LEDA (LEgislative Design and Advisory) System for the Dutch Department of Justice.

After the unfortunate demise of the institute he was appointed as assistant professor at the department of Informatics and Accountancy (BIKA) of the Faculty of Economics at the Katholieke Universiteit Brabant in January 1996. He is teaching courses on Information Technology, Programming and Software Engineering. He is also teaching courses on Interactive Systems in the Hogeschool West-Brabant's MSc. Development and Application of Information Systems programme.

His research interests concern the application of linguistics and language technology in knowledge engineering and information system development, human-computer interfaces, and computer supported cooperative work.

A LANGUAGE-ACTION PERSPECTIVE ON THE DESIGN OF COOPERATIVE INFORMATION AGENTS

Designed by the author using Microsoft Word® for Apple Macintosch®.
Set in Times and AppleGaramond Bk, formulas in Amsterdam, Script MT Bold.
Cover design by Edgar Grimbergen, using Adobe Photoshop® and Quark XPress®.
Composed on a Hewlett Packard Laserjet 4SiMx
Film by GVK, Alblasserdam
Printed by Offsetdrukkerij Ridderprint B.V., Ridderkerk.
Bound by Boekbinderij van Strien, Dordrecht.

Published and distributed by:

Egon M. Verharen
Klein Brabant 132
5262 RR Vught
The Netherlands
phone: +31 (0)73 6561984

Infolab/Tilburg University
POBox 90153
5000 LE Tilburg
The Netherlands
phone: + 31 13 4662767 / ...3020; fax: +31 13 4663069; email: E.M.Verharen@kub.nl
URL: http://infolabwww.kub.nl:2080/infolab/people/egon/

Information about the prototype (including source code) can be found at WWW page:
http://infolabwww.kub.nl:2080/infolab/people/egon/CIA/

Cover justification:

When my good friend Egon asked me to design the cover of his dissertation, I had to tackle the problem of comprehending the complex and to me unknown subject matter, and develop an image that gave form to my (and Egon's) understanding of it. Several sessions with Egon gave me a better understanding and a basic design could be set up. Besides the constraint on book size (16.5x24 cm) there were constraints like: it should fit in the tradition of dissertation covers, yet break with this tradition to attract attention; the title, author, back and spine text were set. Furthermore, Egon clearly indicated that the three main topics of the dissertation: language-action perspective (or communication), intelligent agents (both human and automated), and design should be represented, preferably interrelated, since that is the message of this thesis. For the agent part I took "human and automated" as the central theme and created an image (lower left corner) consisting of a thinking man and a computer printboard (also, see the spine, where another human/computer morph is featured). This idea was carried over to the communication metaphor (right side). Here, agents with their own environment (represented by their aura) communicate based on some shared idea/concepts (touching auras). The design aspect finally, is taken very literally. I scanned one of Egon's design sketches (the one he explained me the thesis with), and laid it under the title (top). For the basic color, green was chosen, both because it is a smoothing color, and it hasn't been used much for dissertation covers (to my knowledge) thereby attracting attention. (All colors are made up of Pantone 300 and 584). The several areas (fields) are austere, because this expresses serenity yet seriousness. (It is also in line with Egon's personal taste for modern and symmetrical design, like applied in his home interior, in contrast to his office :-)). All titles and texts were designed in concordence with this. In the title furthermore the separation between theory (language/action) and practice (design of computer systems) is represented.

Edgar Grimbergen

Stellingen

behorend bij het proefschrift

A Language-Action Perspective on the Design of Cooperative Information Agents

Egon M. Verharen

1. Het taalhandelingsperspectief (LAP) biedt niet alleen een rijk paradigma om coördinatie en communicatie in en tussen organisaties te beschrijven, maar is tevens een adequaat raamwerk voor het ontwerp van coöperatieve informatiesystemen. (Hoofdstuk 2)

2. Het opsplitsen van de activiteitenbeschrijving van agenten in transacties, contracten en taken draagt bij aan de beheersbaarheid van het communicatieproces. (Hoofdstuk 5)

3. De combinatie van dynamische deontische logica en illocutionaire logica maakt het mogelijk om communicatie formeel te beschrijven en is daarmee een geschikte kandidaat voor de formele onderbouwing van het taalhandelingsperspectief (LAP). (Hoofdstuk 6)

4. Daar het een illusie is om *alle* uitzonderingen in een communicatie-situatie te kunnen beschrijven (en formaliseren), is het nuttig om ze algemeen als 'overtreding' (violation) van iets dat gedaan had moeten worden te kunnen bestempelen. Deontische logica biedt ons dit handvat. (Hoofdstuk 6)

5. Architecturen en ontwerpmethoden voor intelligente software agenten dienen meer rekening te houden met het taalhandelingsperspectief (LAP). (Hoofdstukken 4 en 7)

6. De stelling van Weigand "een informatiesysteem is niet alleen betrokken op een domein (Universe of Discourse), maar ook op een omgeving (Environment of Discourse)" wordt nog steeds niet op waarde geschat. (hoofdstuk 7)

   Lit.: H. Weigand, *Linguistically Motivated Principles of Knowledge Based Systems*, Proefschrift Vrije Universiteit, Amsterdam, 1989.

7. Wil KQML als volwaardige agent-communicatietaal gebruikt worden dan behoeft het semantiek, hetgeen tot nu toe vermeden is.

8. In deze tijd van begripsvervaging is er ruimte in het spectrum tussen 'notie' (besef) en 'definitie' (formele beschrijving) voor de nieuwe woorden:
defi´noteren <ov.ww.; definoteerde, h. gedefinoteerd> **0.1** het niet geheel eenduidig omschrijven.
defi´notie <de ~ (v.); -s> **0.1** semi-formele omschrijving van de kenmerken, de betekenis van een begrip of woord ⇒ *begripsbepaling* **0.2** vage omschrijving van een nieuw ingevoerde term met behulp van andere (soms even vage) termen **0.3** het definoteren.

9. Daar geautomatiseerde processen door de complexiteit van de gebruikte informatietechnologie nauwelijks meer worden begrepen door degenen die ze uitvoeren, zou de term 'automatisch' vervangen moeten worden door 'automagisch'.

10. Veel, om marketingtechnische redenen, 'multimedia' gedoopte producten verdienen niet meer dan het predicaat 'muddy media' (slecht gestructureerde informatie van verschillende typen).

11. Het versimpelen van gebruikersinterfaces heeft een averechts effect op het gemiddelde digibetisme (computer-analfabetisme) van de gebruikers.

12. Promovendi die hun proefschrift niet afkrijgen zijn niet noodzakelijk lui, maar lijden waarschijnlijk aan het tot nu toe onbekende "taak-uitvoerings-ijdele-hoop-syndroom". (University of Sussex research on laziness project)

A Language-Action Perspective on the Design of Cooperative Information Agents investigates the application of communication modelling according to the Language-Action paradigm, intelligent agent architectures, and the combination of dynamic deontic logic and illocutionary logic to the design of a new generation of information systems. The results of this research can be used in different application settings, such as business process modelling and electronic commerce.

Egon M. Verharen is an assistant professor at the Faculty of Economics of Tilburg University (the Netherlands).

ISBN 90-9010286-8